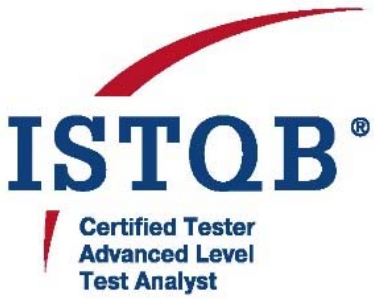
Lehrplan

Certified Tester Advanced Level
Test Analyst Syllabus

Version 4.0



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

Übersetzung des englischsprachigen Lehrplans des International Software Testing Qualifications Board (ISTQB®, Originaltitel: Certified Tester Advanced Level, Test Analyst, Version 4.0.

Urheberschutzvermerk

Änderungsübersicht der deutschsprachigen Ausgabe

Version	Datum	Bemerkungen
v4.0 DE	TT.MM.2025	Deutschsprachige Fassung der englischen Version v4.0 freigegeben
V3.1.2 DE	30.04.2022	Errata: Marc-Florian Wendland, Matthias Hamburg, Klaudia Dussa-Zieger
v3.1.1 DE	20.05.2021	Errata: Matthias Hamburg
v3.1.0 DE	03.05.2021	Geringfügiges Update mit Neuformulierung von Abschnitt 3.2.3 und verschiedenen Verbesserungen im Wortlaut
V2019 DE	05.04.2020	Deutschsprachige Fassung freigegeben

Inhaltsverzeichnis

Danksagungen

Die Generalversammlung des ISTQB® hat das englische Dokument am 2. Mai 2025 offiziell freigegeben.

Es wurde von einem Team des International Software Testing Qualifications Board erstellt: Armin Born, Filipe Carlos, Wim Decoutere, István Forgács, Matthias Hamburg (Product Owner), Attila Kovács, Sandy Liu, François Martin, Stuart Reid, Adam Roman, Jan Sabak, Murian Song, Tanja Tremmel, Marc-Florian Wendland, Tao Xian Feng.

Das Team dankt Julia Sabatine für ihr Korrekturlesen, Gary Mogyorodi für sein Technisches Review, Daniel Pol'an für seine ständige technische Unterstützung sowie dem Reviewteam und den nationalen Mitgliedboards für ihre Anregungen und Beiträge.

Die folgenden Personen waren am Review, der Kommentierung und der Abstimmung über diesen Lehrplan beteiligt:

Aktuelle Ausgabe (v4.0): Gergely Ágnecz, Laura Albert, Dani Almog, Somying Atiporntham, Andre Baumann, Lars Bjørstrup, Ralf Bongard, Earl Burba, Filipe Carlos, Renzo Cerquozzi, Kanitta Chantaramanon, Alessandro Collino, Zheng Dandan, Nicola De Rosa, Aneta Derkova, Dingguofu, Karol Frühauf, Dinu Gamage, Chen Geng, Sabine Gschwandtner, Ole Chr. Hansen, Zsolt Hargitai, Tharushi Hettiarachchi, Ágota Horváth, Arnika Hryszko, Caroline Julien, Beata Karpinska, Ramit Manohar Kaul, Mattijs Kemmink, László Kvintovics, Thomas Letzkus, Marek Majerník, Donald Marcotte, Dénes Medzihradszky, Imre Mészáros, Krisztián Miskó, Gary Mogyorodi, Ebbe Munk, Maysinee Nakmanee, Ingvar Nordström, Tal Pe'er, Kunlanan Peetijade, Sahani Pinidiya, Lukáš Piška, Nishan Portoyan, Meile Posthuma, Yasassri Ratnayake, Randall Rice, Adam Roman, Marc Rutz, Jan Sabak, Vimukthi Saranga, Sumuduni Sasikala, Gil Shekel, Radoslaw Smilgin, Péter Sótér, Helder Sousa, Richard Taylor, Benjamin Timmermans, Giancarlo Tomasig, Robert Treffny, Shun Tsunoda, Stephanie Ulrich, François Vaillancourt, Linda Vreeswijk, Carsten Weise, Marc-Florian Wendland, Paul Weymouth, Elżbieta Wiśniewska, John Young, Claude Zhang.

Ausgabe 2021-2022 (v3.1): Gery Ágnecz, 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.

In den Errata 3.1.1 und 3.1.2 wurden grammatikalische und wörtliche Verbesserungen aufgrund der Vorschläge von Tal Pe'er, Stuart Reid, Marc-Florian Wendland und Matthias Hamburg veröffentlicht.

Ausgabe 2019 (v3.0): 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 Pe'er, 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.

Ausgabe 2012 (v2.0): 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, Chris van Bael, Jurian

van der Laar, Stephanie van Dijk, Erik van Veenendaal, Hans Weiberg, Paul Weymouth, Wenqiang Zheng, Debi Zylbermann.

0 Einführung in diesen Lehrplan

0.1 Zweck dieses Lehrplans

Dieser Lehrplan bildet die Grundlage für die internationale Softwaretestqualifikation für den ISTQB® Advanced Level Test Analyst (CTAL-TA). Das German Testing Board e.V. (im Folgenden GTB genannt) hat diesen Lehrplan in Zusammenarbeit mit dem Swiss Testing Board (STB) in die deutsche Sprache übersetzt. Das ISTQB® stellt den Lehrplan folgenden Adressaten zur Verfügung:

1. Nationalen Mitgliedboards, die den Lehrplan in ihre Sprache(n) übersetzen und Schulungsanbieter akkreditieren dürfen. Die nationalen Mitgliedboards dürfen den Lehrplan an die Anforderungen ihrer nationalen Sprache anpassen und Referenzen hinsichtlich lokaler Veröffentlichungen berücksichtigen.
2. Zertifizierungsstellen zur Ableitung von Prüfungsfragen in ihrer nationalen Sprache, die an die Lernziele dieses Lehrplans angepasst sind.
3. Schulungsanbietern zur Erstellung von Lehrmaterialien und zur Bestimmung angemessener Lehrmethoden.
4. Zertifizierungskandidaten zur Vorbereitung auf die Zertifizierungsprüfung (entweder als Teil einer Schulung oder unabhängig davon).
5. Der internationalen Software- und Systementwicklings-Community zur Förderung des Berufsbildes des Software- und Systemtests und als Grundlage für Bücher und Fachartikel.

0.2 Der ISTQB® Advanced Level Test Analyst (CTAL-TA) im Softwaretest

Die ISTQB®-Zertifizierung Advanced Level Test Analyst (CTAL-TA) vermittelt die erforderliche Qualifikation, um strukturiertes und gründliches Softwaretesten über den gesamten Softwareentwicklungslebenszyklus hinweg durchzuführen. Sie beschreibt detailliert die Rolle und die Verantwortlichkeiten der Testanalysten in jedem Schritt eines Standard-Testprozesses und geht auf wichtige Testverfahren näher ein. Die Zertifizierung zum ISTQB® Advanced Level Test Analyst (CTAL-TA) richtet sich an Personen, die bereits ein ISTQB®-Zertifikat Certified Tester Foundation Level besitzen und ihr Fachwissen im Bereich Testanalyse und Testverfahren weiter ausbauen möchten.

In diesem Lehrplan wird Testanalyst als eine Rolle verstanden, die

- sich mehr auf die geschäftlichen Bedürfnisse des Kunden als auf technische Aspekte des Testens konzentriert,
- hauptsächlich funktionale Tests durchführt, sich aber auch an benutzerorientierten, nichtfunktionalen Tests beteiligt, wie Gebrauchstauglichkeits-, Anpassbarkeits-, Installierbarkeits- oder Interoperabilitätstests,
- Black-Box-Testverfahren und erfahrungsbasierten Test vorrangig vor White-Box-Testverfahren verwendet und
- die Effektivität des Testens durch Fehlerpräventionsverfahren verbessert.

0.3 Karriereweg für Tester

Das ISTQB®-Programm unterstützt Testexperten in allen Phasen ihrer Laufbahn. Personen, die die ISTQB®-Zertifizierung Advanced Level Test Analyst (CTAL-TA) erlangen, sind möglicherweise auch an den anderen Core Advanced Levels (Technical Test Analyst und Testmanagement) und danach an den Expert Levels (Testmanagement oder Verbesserung des Testprozesses) interessiert. Für alle, die ihre Fähigkeiten im Bereich des Testens in einer agilen Umgebung ausbauen möchten, kommen die Zertifizierungen ISTQB® Agile Technical Tester oder ISTQB® Agile Test Leadership at Scale in Frage. Darüber hinaus bieten Spezialistenmodule Zertifizierungsprodukte an, die sich auf bestimmte Testtechnologien und -ansätze, bestimmte Qualitätsmerkmale und Teststufen oder das Testen in bestimmten Industriebereichen konzentrieren. Aktuelle Informationen über das ISTQB®-Certified-Tester-Schema finden Sie unter www.istqb.org, unter dem Berufsbild Testen des GTB unter www.gtb.de/der-certified-tester/berufsbild/ oder auf den Seiten der nationalen Boards, wie z. B. www.gtb.de (Deutschland), www.swisstestingboard.org (Schweiz) oder www.austriantestingboard.at (Österreich).

0.4 Geschäftlicher Nutzen

In diesem Abschnitt werden die geschäftlichen Nutzen (Business Outcomes, BO) aufgeführt, die von Kandidaten erwartet werden, die die Zertifizierung zum ISTQB® Advanced Level Test Analyst (CTAL-TA) erlangt haben.

Ein zertifizierter ISTQB® Advanced Level Test Analyst (CTAL-TA) kann Folgendes:

ID	Beschreibung
TA-BO1	Für den angewendeten Softwareentwicklungslebenszyklus angemessenes Testen unterstützen und durchführen.
TA-BO2	Die Grundsätze des risikobasierten Tests anwenden.
TA-BO3	Geeignete Testverfahren auswählen und anwenden, um die Erreichung der Testziele zu unterstützen.
TA-BO4	Dokumentation mit angemessenem Detaillierungsgrad und geeigneter Qualität bereitstellen.
TA-BO5	Die geeigneten funktionalen Testarten bestimmen, die durchgeführt werden sollen.
TA-BO6	Zu nicht-funktionalen Tests beitragen.
TA-BO7	Zur Fehlerprävention beitragen.
TA-BO8	Die Effizienz des Testprozesses durch den Einsatz von Werkzeugen verbessern.
TA-BO9	Die Anforderungen an Testumgebungen und Testdaten spezifizieren. .

0.5 Lernziele und kognitive Wissensstufe

Die Lernziele (Learning Objectives, LO) unterstützen den geschäftlichen Nutzen und dienen zur Ausarbeitung der Prüfungen zum ISTQB® Advanced Level Test Analyst (CTAL-TA).

Im Allgemeinen sind alle Inhalte dieses Lehrplans prüfbar, mit Ausnahme der Einleitung und der Anhänge. Die Prüfungsfragen bestätigen die Kenntnis der Schlüsselbegriffe auf der Stufe K1 (sich erinnern, siehe unten) oder der Lernziele auf der jeweiligen Wissensstufe.

Die spezifischen Lernziele und ihre Wissensstufen sind zu Beginn jedes Kapitels aufgeführt und wie folgt klassifiziert:

- K2: Verstehen
- K3: Anwenden
- K4: Analysieren

Für alle Begriffe, die als Schlüsselbegriffe direkt unter den Kapitelüberschriften aufgelistet sind, müssen die korrekte Bezeichnung und Definition aus dem ISTQB®-Glossar bekannt sein (K1), auch wenn sie nicht ausdrücklich in den Lernzielen erwähnt werden.

Weitere Einzelheiten und Beispiele für Lernziele finden Sie in *Abschnitt 7*.

0.6 Die Zertifizierungsprüfung zum ISTQB® Advanced Level Test Analyst (CTAL-TA)

Die Prüfung für das ISTQB®-Zertifikat Advanced Level Test Analyst (CTAL-TA) basiert auf diesem Lehrplan. Zur Beantwortung einer Prüfungsfrage kann die Verwendung von Material aus mehr als einem Abschnitt dieses Lehrplans erforderlich sein. Alle Abschnitte des Lehrplans sind prüfungsrelevant, mit Ausnahme der Einführung und der Anhänge. Im Lehrplan sind Standards und Fachbücher als Referenzen genannt, ihr Inhalt ist jedoch nicht prüfungsrelevant, abgesehen von dem, was im vorliegenden Lehrplan in zusammengefasster Form enthalten ist.

Weitere Einzelheiten können im Dokument "Exam Structures and Rules" des ISTQB® gefunden werden, das mit dem Lehrplan des ISTQB® Advanced Level Test Analyst (CTAL-TA) v4.0 kompatibel ist.

Voraussetzung für die Teilnahme an der Zertifizierung zum ISTQB® Advanced Level Test Analyst (CTAL-TA) ist, dass die Kandidaten ein Interesse an Softwaretests haben. Es wird jedoch dringend empfohlen, dass die Kandidaten auch

- ein Mindestmaß an Erfahrung in der Softwareentwicklung oder im Testen von Software haben, z. B. sechs Monate Erfahrung als System- oder Abnahmetester oder als Softwareentwickler,
- einen Kurs absolvieren, der nach ISTQB®-Standards akkreditiert ist (von einem der nationalen Mitgliedboards des ISTQB®).

Eingangsvoraussetzung für die Teilnahme: Das ISTQB®-Zertifikat Certified Tester Foundation Level muss vor der Zertifizierungsprüfung zum Advanced Level Test Analyst erworben werden.

0.7 Akkreditierung

Ein nationales ISTQB®-Mitgliedboard kann Schulungsanbieter akkreditieren, deren Lehrmaterial diesem Lehrplan entspricht. Schulungsanbieter sollten die Akkreditierungsrichtlinien vom nationalen Mitgliedsboard beziehen (in Deutschland: German Testing Board e.V.; in der Schweiz: Swiss Testing Board; in Österreich: Austrian Testing Board) oder von der Stelle, die die Akkreditierung in dessen Auftrag durchführt. Eine akkreditierte Schulung wird als konform mit diesem Lehrplan anerkannt und darf eine ISTQB®-Prüfung als Teil der Schulung enthalten. Die Akkreditierungsrichtlinien für diesen Lehrplan folgen den allgemeinen Akkreditierungsrichtlinien, die von der ISTQB®-Arbeitsgruppe "Processes Management and Compliance" veröffentlicht wurden.

0.8 Umgang mit Normen und Standards

Internationale Normungsgremien wie IEEE und ISO haben Normen zu Qualitätsmerkmalen und zum Testen von Software herausgegeben. Auf solche Normen wird in diesem Lehrplan verwiesen. Der Zweck dieser Verweise ist es, einen Rahmen zu schaffen (wie bei den Verweisen auf ISO/IEC 25010 bezüglich der Qualitätsmerkmale) oder eine Quelle für zusätzliche Informationen zu bieten, falls der Leser dies wünscht. Bitte beachten Sie, dass in den ISTQB®-Lehrplänen Normdokumente als Referenz verwendet werden. Die Inhalte der Normen sind nicht prüfungsrelevant. Weitere Informationen zu Normen finden Sie in Kapitel 6 - Referenzen.

0.9 Detaillierungsgrad

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

- allgemeinen Lehrzielen, die die Absicht des Advanced Level Test Analyst beschreiben,
- einer Liste von Begriffen, die die Lerenenden kennen müssen,
- Lernzielen für jeden Wissensbereich, die die zu erreichenden kognitiven Lernergebnisse beschreiben,
- einer Beschreibung der Schlüsselkonzepte, einschließlich Verweisen auf Quellen wie anerkannte Literatur oder Normen.

Der Inhalt des Lehrplans beschreibt nicht das gesamte Wissensgebiet des Softwaretestens. Er spiegelt den Detaillierungsgrad wider, der in den Schulungen für ISTQB® Advanced Level Test Analyst (CTAL-TA) abgedeckt werden soll.

Der Lehrplan verwendet die Terminologie (d.h. den Namen und die Bedeutung) der Begriffe, die beim Testen von Software und bei der Qualitätssicherung verwendet werden, gemäß dem ISTQB®-Glossar (ISTQB-Glossary, 2024).

Für die Terminologie verwandter Disziplinen wird auf deren jeweilige Glossare verwiesen: IREB-CPRE für Requirements Engineering (IREB-Glossary, 2024) und IEEE-Pascal für Softwaretechnik (Computer Society u. a., 2024).

0.10 Aufbau des Lehrplans

Es gibt fünf Kapitel mit prüfbarem Inhalt. Die Überschrift auf der obersten Ebene jedes Kapitels gibt die Zeit für das Kapitel an. Unterhalb der Kapitelebene werden keine Zeitangaben gemacht. Für akkreditierte Ausbildungskurse verlangt der Lehrplan mindestens **20,25 Unterrichtsstunden** (**1215 Minuten**), die sich wie folgt auf die fünf Kapitel verteilen:

- Kapitel 1 (225 Minuten): Die Aufgaben der Testanalysten im Testprozess
  - Der Lernende versteht, wie Testanalysten in verschiedene Softwareentwicklungslebenszyklen eingebunden sind.
  - Der Lernende versteht, wie Testanalysten in verschiedene Testaktivitäten eingebunden sind.
  - Der Lernende versteht die Aufgaben der Testanalysten im Hinblick auf die Testmittel kennen.

- Kapitel 2 (90 Minuten) Die Aufgaben der Testanalysten beim risikobasierten Testen
  - Der Lernende versteht, wie Testanalysten zur Risikoanalyse des Produkts beitragen.
  - Der Lernende kann die Auswirkungen von Änderungen analysieren, um den Umfang von Regressionstests zu bestimmen.
- Kapitel 3 (615 Minuten) Testanalyse und Testentwurf
  - Der Lernende kann datenbasierte Testverfahren anwenden, wie Wertebereichstest, kombinatorischen Test und Zufallstest.
  - Der Lernende kann verhaltensbasierte Testverfahren anwenden, wie CRUD-Test, Zustandsübergangstest und szenariobasierten Test.
  - Der Lernende kann regelbasierte Testverfahren anwenden, wie Entscheidungstabellentest und metamorphen Test.
  - Der Lernende versteht erfahrungsbasiertes Testen, wie sitzungsbasiertes Testen und Massentest.
  - Der Lernende kann geeignete Testverfahren auswählen, um Produktrisiken zu mindern.
- Kapitel 4 (60 Minuten) Testen von Qualitätsmerkmalen
  - Der Lernende versteht, wie man verschiedene Arten von funktionalen Tests durchführt.
  - Der Lernende versteht, wie er das spezifische Wissen über Funktionalität nutzen kann, um zu nicht-funktionalen Testarten beizutragen, wie Gebrauchstauglichkeitstests, Flexibilitätstests und Kompatibilitätstests.
- Kapitel 5 (225 Minuten) Fehlerprävention im Softwarebereich
  - Der Lernende versteht die verschiedenen Praktiken zur Fehlerprävention.
  - Der Lernende kann verschiedene Ansätze zur Fehlereindämmung innerhalb der Phase anwenden.
  - Der Lernende kann das Wiederauftreten von Fehlern eindämmen.

1 Die Aufgaben der Testanalysten im Testprozess - 225 Minuten

Schlüsselbegriffe

abstrakter Testfall, konkreter Testfall, Schlüsselwort, schlüsselwortgetriebener Test, Softwareentwicklungslebenszyklus, Testanalyse, Testanalyst, Testbedingung, Testdaten, Testdurchführung, Testentwurf, Testfall, Testmittel, Testorakel, Testrealisierung, Testskript, Testumgebung

Lernziele für Kapitel 1:

Die Lernenden können ...

1.1 Testen im Softwareentwicklungslebenszyklus

TA-1.1.1 (K2) ... die Beteiligung von Testanalysten an verschiedenen Softwareentwicklungslebenszyklen zusammenfassen

1.2 Beteiligung an Testaktivitäten

TA-1.2.1 (K2) ... die Aufgaben von Testanalysten im Rahmen der Testanalyse zusammenfassen

TA-1.2.2 (K2) ... die Aufgaben von Testanalysten im Rahmen des Testentwurfs zusammenfassen

TA-1.2.3 (K2) ... die Aufgaben von Testanalysten im Rahmen der Testrealisierung zusammenfassen

TA-1.2.4 (K2) ... die Aufgaben von Testanalysten im Rahmen der Testdurchführung zusammenfassen

1.3 Aufgaben im Zusammenhang mit den Arbeitsprodukten

TA-1.3.1 (K2) ... zwischen abstrakten Testfällen und konkreten Testfällen unterscheiden

TA-1.3.2 (K2) ... die Qualitätskriterien für Testfälle erklären

TA-1.3.3 (K2) ... Beispiele für Anforderungen an die Testumgebung geben

TA-1.3.4 (K2) ... das Testorakelproblem und mögliche Lösungen erklären

TA-1.3.5 (K2) ... Beispiele für Anforderungen an Testdaten geben

TA-1.3.6 (K3) ... den schlüsselwortgetriebenen Test zur Entwicklung von Testskripten nutzen

TA-1.3.7 (K2) ... die Werkzeugarten zur Verwaltung von Testmitteln zusammenfassen

Einführung in die Aufgaben des Testanalysten im Testprozess

Der Foundation-Level-Lehrplan (ISTQB-CTFL, v4.0.2 DE) beschreibt zwei Hauptrollen beim Testen: die Rolle des Testmanagements und die Rolle des Testers. In diesem Lehrplan werden Personen, die in einer Testerrolle für das Testen der geschäftlichen Aspekte der Software verantwortlich sind, als [Testanalyst]en{.index} bezeichnet. Obwohl diese Verantwortung nur selten einer speziellen Position oder Rolle zugewiesen wird, haben Testanalysten klar definierte Aufgaben und Anforderungen an Kompetenzen. Was die Teststufen angeht, so konzentrieren sich Testanalysten auf Systemtests, Abnahmetests und Systemintegrationstests. Bei den Qualitätsmerkmalen konzentrieren sich die Kompetenzen der Testanalysten auf die funktionale Eignung und decken auch bestimmte, dem Benutzer zugewandte nichtfunktionale Qualitätsmerkmale wie Gebrauchstauglichkeit, Anpassbarkeit, Installierbarkeit und Interoperabilität ab.

1.1 Testen im Softwareentwicklungslebenszyklus

Die Testaktivitäten können je nach Softwareentwicklungslebenszyklus (SDLC) unterschiedlich organisiert sein. Daher kann die Beteiligung der Testanalysten an den Testaktivitäten je nach dem angewendeten SDLC-Modell variieren.

In **sequenziellen Entwicklungsmodellen** werden die Entwicklungsaktivitäten in Phasen durchgeführt und beginnen, wenn die vorherige Phase abgeschlossen ist. In der Regel gibt es nur wenige Überschneidungen zwischen diesen Aktivitäten. Daher ändern sich die Aufgaben der Testanalysten üblicherweise im Laufe der Zeit. In den frühen Phasen des SDLC konzentrieren sich Testanalysten auf die Unterstützung der Testplanung. Bei der Erstellung der Testbasis beginnen Testanalysten mit der Testanalyse. Testentwurf und Testrealisierung folgen parallel zum Softwareentwurf und zur Implementierung. In den späten Phasen des SDLC führen Testanalysten Tests aus und unterstützen den Testabschluss.

Inkrementelle Entwicklungsmodelle unterteilen die Software in kleinere, überschaubare Inkremente. Jedes Inkrement wird unabhängig entwickelt und getestet. Dabei führen Testanalysten für jedes Inkrement die gleichen Tätigkeiten aus (d. h. Testanalyse, Testentwurf, Testrealisierung, Testdurchführung und Unterstützung des Testmanagements). Allerdings kann die Arbeit der Testanalysten für jedes Inkrement anders organisiert sein. Das Testen konzentriert sich auf die neuen oder geänderten Funktionen. Aufgrund des erhöhten Regressionsrisikos müssen Testanalysten außerdem der Restrukturierung (Refactoring) und dem Aufbau von Testsuiten für Regressionstests besondere Aufmerksamkeit widmen.

In **iterativen Entwicklungsmodellen** ist der Entwicklungsprozess zyklisch. Das Projekt durchläuft wiederholt mehrere Zyklen von Prototypenerstellung, Testen, Verfeinerung und Bereitstellung. Die Rolle der Testanalysten ist dynamisch und anpassungsfähig. Testanalysten arbeiten eng mit Entwicklern und Vertretern der Geschäftsseite zusammen und passen sich an das entstehende Produkt an. Wenn sich die Software weiterentwickelt, passen Testanalysten die Testbedingungen und Testfälle an und geben bei jeder Iteration Feedback zur Verbesserung des Testprozesses. Je häufiger die Iterationen sind, desto wichtiger ist die laufende Wartung und Entwicklung der Regressionstests durch Testanalysten.

Ein SDLC kann Elemente verschiedener Modelle und spezifischer Verfahren und Ansätze kombinieren (z. B. kombiniert agile Softwareentwicklung Aspekte des iterativen und des inkrementellen Modells). In solchen Fällen hängt die Beteiligung der Testanalysten von den spezifischen Merkmalen des SDLC und deren Kombination ab. Eine gute Praxis, die allen SDLC-Modellen gemeinsam ist, besteht darin, dass Testanalysten bereits in den Anfangsphasen des SDLC einbezogen werden sollten.

1.2 Beteiligung an Testaktivitäten

Der Foundation-Level-Lehrplan (ISTQB-CTFL, v4.0.2 DE) beschreibt sieben Aktivitäten innerhalb des Testprozesses. Testanalysten konzentrieren sich hauptsächlich auf vier von ihnen: Testanalyse, Testentwurf, Testrealisierung und Testdurchführung.

Die Aufgaben der Testanalysten im Rahmen dieser vier Aktivitäten werden in den folgenden Abschnitten ausführlich beschrieben.

1.2.1 Testanalyse

Bei der Testanalyse prüfen Testanalysten die Testbasis auf Vollständigkeit und sammeln alle zusätzlichen Informationen, die für das Testen relevant sind. Dazu gehören die Dokumentation und auch verbale Informationen, wie z. B. Gespräche beim kollaborativen Schreiben von User-Storys (siehe ISTQB-CTFL, v4.0.2 DE, Abschnitt 4.5.1). Änderungen an der Testbasis können in Abstimmung mit dem Testmanagement zu Anpassungen am Testumfang führen.

Um mit der Testanalyse effektiv fortzufahren, prüfen Testanalysten die folgenden Eingangskriterien:

- Die Testplanung wurde durchgeführt und der Testumfang, die Testziele und der Testansatz sind klar.
- Die Testbasis (mit Informationen wie Anforderungen oder User-Storys) ist definiert.
- Die bereits identifizierten Produktrisiken wurden bewertet und bei Bedarf dokumentiert.

Testanalysten bewerten die Testbasis, um mögliche Fehlerzustände zu identifizieren und die Testbarkeit zu beurteilen und geben so frühzeitig Rückmeldung an die Product Owner. Dies kann die Modellierung des Systemverhaltens entsprechend den anzuwendenden Testverfahren beinhalten (siehe *Abschnitt 3*, *Abschnitt 5.2.1*). Reviewverfahren werden ebenfalls als Teil des Prozesses angewendet (siehe *Abschnitt 5.2.2*). Fehlerzustände in der Testbasis, die nicht direkt behoben werden, müssen dokumentiert werden. Außerdem bestimmen Testanalysten die benötigten Testorakel (siehe *Abschnitt 1.3.4*).

Testanalysten bestimmen und priorisieren für jedes Testelement Testbedingungen. Die Testbedingungen beziehen sich auf die Testziele (siehe ISTQB-CTFL, v4.0.2 DE, Abschnitt 1.1.1) und müssen zu den Elementen der Testbasis verfolgbar sein. Der Umfang und der Fokus der Testbedingungen berücksichtigen die Produktrisiken. Bei inkrementellen oder iterativen Entwicklungsmodellen gehört dazu auch die Festlegung des Umfangs von Regressionstests auf Basis einer Auswirkungsanalyse. In der agilen Softwareentwicklung können die Testbedingungen als Akzeptanzkriterien formuliert werden, die die Risiken der User-Storys widerspiegeln.

Testanalysten können stufenweise vorgehen, beginnend mit übergeordneten Testbedingungen wie "Die Funktionalität von Bildschirm x". Als Nächstes bestimmen Testanalysten detailliertere Testbedingungen wie "Der Bildschirm x lehnt Kontonummern ab, die eine Ziffer zu kurz sind". Dieser Ansatz ermöglicht eine ausreichende Überdeckung und einen frühen Beginn des Testentwurfs, z. B. für User-Storys, die noch verfeinert werden müssen.

Testanalysten beziehen die Stakeholder in das Review der Testbedingungen ein, um sicherzustellen, dass die Testbasis richtig verstanden wird und die Tests mit den Testzielen übereinstimmen.

1.2.2 Testentwurf

Der Testentwurf beschreibt, wie das Testen durchgeführt werden soll, um die gesetzten Testziele zu erreichen. Dies geschieht in der Regel mit Testfällen. Die Art und Weise, wie der Testentwurf ausgeführt

wird, hängt von vielen Faktoren ab, u. a. von der geforderten Überdeckung, der Testbasis, dem SDLC, den Projektbeschränkungen sowie dem Wissen und der Erfahrung der Tester.

Während des Testentwurfs bestimmen Testanalysten, in welchen Bereichen abstrakte Testfälle oder konkrete Testfälle angemessen sind (siehe *Abschnitt 1.3.1*). In beiden Fällen müssen Testanalysten klare Bestehens-/Fehlschlagkriterien festlegen. Testanalysten entwerfen Testfälle für neue oder geänderte Testbedingungen, die den Qualitätskriterien entsprechen (siehe *Abschnitt 1.3.2*). Für Regressionstests reicht in der Regel die Auswahl bestehender abstrakter Testfälle oder die Anpassung bestehender konkreter Testfälle auf der Grundlage ihrer Priorisierung aus.

Testanalysten erfassen die Verfolgbarkeit zwischen Testbasis, Testbedingungen und Testfällen. Beim erfahrungsbasierten Test sind Testfälle nicht immer dokumentiert; stattdessen können Testbedingungen (neben anderen) die Testdurchführung leiten. Testanalysten können auf Grundlage übergeordneter Testziele einige Testfälle entwerfen.

Zusätzlich zu diesen Aufgaben definieren Testanalysten die Anforderungen an die Testumgebung (siehe *Abschnitt 1.3.3*) und identifizieren, erstellen und spezifizieren die Anforderungen an die Testdaten (siehe *Abschnitt 1.3.5*).

Testanalysten verwenden die in der Testplanung festgelegten Endekriterien, um festzustellen, wann genügend Testfälle entworfen worden sind. Aber auch andere Endekriterien, wie die verbleibenden Risikostufen oder Projektbeschränkungen (z. B. Budget oder Zeit), geben an, wann der Testentwurf beendet werden kann.

Der Testentwurf kann von Werkzeugen unterstützt werden, sollte aber werkzeug- und technologieunabhängig sein, um unabhängig von den Werkzeugen zu bleiben. Der Testentwurf wendet einen systematischen Testansatz unter Einsatz von Testverfahren an oder erfolgt ansonsten ad hoc.

Testfälle haben eine kommunikative Funktion und sollten für die relevanten Stakeholder verständlich sein. Da ein Testfall nicht immer von seinem Autor ausgeführt werden kann, müssen andere Tester verstehen, wie er auszuführen ist, und seine Bedeutung und Testziele (d. h. die zugrunde liegenden Testbedingungen) kennen. Testfälle müssen auch für Entwickler verständlich sein, die die Tests implementieren oder sie im Falle einer Fehlerwirkung wiederholen können, sowie für Auditoren, die sie möglicherweise genehmigen müssen.

1.2.3 Testrealisierung

Bei der Testrealisierung stellen Testanalysten die für die Testdurchführung benötigten Testmittel bereit. Testanalysten können Testabläufe und Testskripte in Testsuiten organisieren oder Testfälle für die Automatisierung vorschlagen. Die Definition von Testabläufen erfordert eine sorgfältige Identifizierung der Einschränkungen und Abhängigkeiten, die die Reihenfolge der Testdurchführung beeinflussen können. Zusätzlich zu den in den Testfällen enthaltenen Schritten umfassen die Testabläufe alle Schritte zur Einrichtung der initialen Vorbedingungen (z. B. Laden von Testdaten aus einem Datenrepository), zur Überprüfung der erwarteten Ergebnisse und der Nachbedingungen sowie zum Zurücksetzen nach Testdurchführung (z. B. Zurücksetzen von Datenbank, Umgebung und System).

Testanalysten priorisieren Testabläufe und Testskripte für die Testdurchführung auf Grundlage der bei Risikoanalyse und Testplanung ermittelten Priorisierungskriterien und identifizieren Testabläufe oder Testskripte, die für die aktuelle Version des Testobjekts ausgeführt werden sollen. Auf diese Weise können zusammengehörige Tests (z. B. für neue Funktionen oder Regressionstests) gemeinsam in einem bestimmten Testlauf ausgeführt werden. Testanalysten aktualisieren die Verfolgbarkeit zwischen der Testbasis und anderen Testmitteln wie Testabläufe, Testskripte und Testsuiten.

Testanalysten können das Testmanagement bei der Festlegung eines Testausführungsplans, einschließlich der Ressourcenzuweisung, unterstützen, um eine effiziente Testdurchführung zu ermöglichen, indem sie die Reihenfolge der Testausführung festlegen (siehe ISTQB-CTFL, v4.0.2 DE, Abschnitt 5.1.5).

Testanalysten erstellen Eingabe- und Umgebungsdaten, die in Datenbanken und andere Repositories geladen werden (siehe *Abschnitt 1.3.5*). Diese Daten müssen für die Unterstützung spezifischer Testziele geeignet sein.

Testanalysten sollten auch überprüfen, ob die Testumgebung vollständig eingerichtet und für die Testdurchführung bereit ist (siehe *Abschnitt 1.3.3*). Dies geschieht am besten durch das Entwerfen und Ausführen eines Smoke-Tests. Die Testumgebung sollte bei der Testdurchführung Fehlerzustände des Testobjekts aufdecken, wenn keine Fehlerwirkungen auftreten normal funktionieren und falls erforderlich, die Produktions- oder Benutzerumgebung angemessen nachbilden.

Der Detaillierungsgrad und die damit verbundene Komplexität der während der Testrealisierung durchgeführten Arbeiten kann durch den Detaillierungsgrad der Testbedingungen und Testfälle beeinflusst werden. In einigen Fällen gelten gesetzliche Vorschriften und die Testmittel sollten den Nachweis der Konformität mit den geltenden Normen erbringen (z. B. *RTCA DO-178C*, 2011).

1.2.4 Testdurchführung

Die Testdurchführung erfolgt gemäß dem Testausführungsplan. Zu den typischen Aufgaben des Testanalysten gehört es, Tests auszuführen, tatsächliche mit erwarteten Ergebnissen zu vergleichen, Anomalien zu analysieren, Fehler zu melden und Testergebnisse zu protokollieren.

Testanalysten führen Tests manuell durch. Dazu gehören explorative Tests, das Ausführen von Testabläufen, Regressionstests und Fehlernachtests. Für explorative Tests können Testanalysten sitzungsbasiertes Testen mit Test-Chartas verwenden (siehe *Abschnitt 3.4.1*). Testanalysten können auch automatisierte Testskripte ausführen, aber es kann die Aufgabe von Entwicklern, Testautomatisierungsentwicklern oder technischen Testanalysten sein, automatisierte Testskripte auszuführen und sie im Falle eines Fehlschlags zu analysieren.

Testanalysten analysieren Anomalien, die bei manuellen oder automatisierten Testausführungen auftreten, um deren wahrscheinliche Ursachen zu ermitteln. Eine Anomalie kann die Folge eines Fehlerzustands in einem Testobjekt sein. Es kann aber auch andere Gründe für Anomalien geben, z. B. fehlende Vorbedingungen, falsche Testdaten, Fehlerzustände in Testskripten oder in der Testumgebung oder ein Missverständnis bei den Spezifikationen. Testanalysten protokollieren die Istergebnisse der Testdurchführung, teilen Fehlerzustände auf der Grundlage der beobachteten Fehlerwirkungen mit und erstellen bei Bedarf Fehlerberichte.

Testanalysten aktualisieren die Verfolgbarkeit zwischen der Testbasis und anderen Testmitteln unter Berücksichtigung der Testergebnisse. Diese Informationen ermöglichen die Umwandlung von Testergebnissen in Risiko- oder Überdeckungsinformationen auf hoher Ebene, wodurch die Stakeholder in der Lage sind, fundierte Entscheidungen zu treffen. So wird den Stakeholdern beispielsweise klar, wie viele Testfälle im Zusammenhang mit einer Testbedingung bestanden oder fehlgeschlagen sind.

Zusätzlich zu diesen typischen Aufgaben werten Testanalysten Testergebnisse aus, einschließlich der folgenden Aufgaben:

- Anhäufung von Fehlerzuständen erkennen, die darauf hindeuten, dass ein bestimmter Teil des Testobjekts verstärkt getestet werden muss (siehe *Abschnitt 5.3.1*).

- Fehlgeschlagene automatisierte Tests manuell wiederholen, um sicherzustellen, dass die Testautomatisierung kein falsch-positives Ergebnis geliefert hat.
- Zusätzliche Tests vorschlagen, auf der Grundlage der Erkennisse aus früheren Tests.
- Neue Risiken anhand der bei der Testdurchführung gewonnenen Informationen identifizieren.
- Verbesserungen des Testentwurfs oder der Testrealisierung vorschlagen (z. B. Verbesserungen der Testabläufe) oder sogar für das System unter Test (SUT).
- Verbesserungen der Regressionstestsuiten vorschlagen, einschließlich Restrukturierung (Refactoring), Anpassung des Testumfangs und der Testautomatisierung (siehe *Abschnitt 2.2*).

1.3 Aufgaben im Zusammenhang mit Testmitteln

Testanalysten müssen die Qualität der Testmittel für die sie verantwortlich sind sicherstellen. Dazu gehören Testfälle, Testumgebungen, Testdaten, Testorakel und Testskripte. In diesem Abschnitt werden die Aufgaben der Testanalysten im Zusammenhang mit Testmitteln und die Werkzeugarten zur Verwaltung von Testmitteln erörtert.

1.3.1 Abstrakte und konkrete Testfälle

Ein abstrakter Testfall (auch *logischer Testfall* genannt) beschreibt die Umstände, unter denen das Testobjekt untersucht wird, und zeigt an, welche Testbedingungen durch den Testfall überdeckt werden. Abstrakte Testfälle sind daher geeignet sicherzustellen, dass alle relevanten Testbedingungen durch Tests überdeckt werden. Abstrakte Testfälle enthalten keine konkreten Informationen über Vorbedingungen, Testdaten, erwartete Ergebnisse oder Nachbedingungen. Diese werden alle auf einer abstrakten Ebene ausgedrückt (z. B. "Bestelle mehr als ein Buch, so dass der Bestellpreis zu einem Rabatt führt; erwartetes Ergebnis: Rabatt wird gewährt").

Ein konkreter Testfall ist die detaillierte Verfeinerung eines abstrakten Testfalls. Konkrete Testfälle beschreiben, welche Daten aufbereitet werden müssen, welche Aktionen der Tester ausführen muss (falls erforderlich) und was das konkrete erwartete Ergebnis ist. Konkrete Testfälle enthalten spezifische Vorbedingungen, Testdaten, erwartete Ergebnisse und Nachbedingungen (z. B. "Bestelle die Bücher B1 (10 €) und B2 (20 €), mit einem Gesamtpreis von 30 €; erwartetes Ergebnis: 10% Rabatt, Gesamtpreis: 27 €").

In der Regel entwerfen Testanalysten zunächst abstrakte Testfälle, die die Grundlage für die Entwicklung konkreter Testfälle bilden. Ein abstrakter Testfall kann dabei zu einem oder mehreren konkreten Testfällen führen. In manchen Fällen kann ein Testfall abstrakt bleiben. In diesem Fall werden die konkreten Informationen während der Testdurchführung durch die Testanalysten bestimmt. Beispielsweise können abstrakte Testfälle die Testanalysten bei der Erstellung von Testzielen für eine Test-Charta beim sitzungsbasierten Testen leiten, um diese Ziele während der Testausführung zu vertiefen (siehe *Abschnitt 3.4.1*).

Der Übergang von abstrakten Testfällen zu konkreten Testfällen ist mehr als nur die Konkretisierung von Werten, es ist auch ein Schritt vom konzeptionellen zum technischen Testmittel. Dieser Schritt wird häufig vom Testentwurf in die Testrealisierung verschoben, insbesondere wenn spezifische Testdaten benötigt werden. Die Testanalysten müssen sicherstellen, dass alles, was zur Ausführung konkreter Testfälle erforderlich ist, bekannt ist (Koomen u. a., 2008). In der Praxis sind viele Testfälle hybrid, d. h. sie sind in einigen Aspekten konkret und in anderen abstrakt. Dies ist häufig auf einen Kompromiss zwischen Wartbarkeit und Verständlichkeit der Testfälle zurückzuführen.

1.3.2 Qualitätskriterien für Testfälle

Die Vernachlässigung der Qualität eines Testfalls kann zu vielen Problemen führen, wie z. B. hohen Wartungskosten, geringerer Verständlichkeit oder Verzögerungen bei der Ausführung. Qualitätskriterien für Testfälle sind ein erster Schritt zu besser wartbaren Testfällen. Sie umfassen:

- **Korrektheit**: Ein Testfall muss eine genaue Verifizierung der Testbedingungen ermöglichen, auf denen er basiert.
- **Durchführbarkeit**: Es muss möglich sein, einen Testfall auszuführen.
- **Notwendigkeit**: Jeder Testfall sollte ein klares Testziel überdecken, das im Titel oder in der Zusammenfassung des Testfalls beschrieben ist. Duplikate sollten vermieden werden. Für Dinge, die nicht getestet werden sollen, sollten keine Testfälle entworfen werden.
- **Verständlichkeit**: Testfälle können von anderen Personen als dem Autor überprüft, modifiziert und ausgeführt werden. Testanalysten sollten Testfälle in einer Sprache und einem Format schreiben, die für alle beteiligten Stakeholder verständlich sind, ohne dabei das Offensichtliche zu erklären. Komplexe Testfälle sollten vereinfacht oder aufgeteilt werden.
- **Verfolgbarkeit**: Testfälle sollten zu Testbedingungen, Anforderungen und Risiken zurückverfolgt werden können, damit die Testanalysten sie auf dem neuesten Stand halten können (siehe ISTQB-CTFL (v4.0.2 DE), Abschnitt 1.4.4).
- **Konsistenz**: Konsistenz in Sprache, Formatierung und Struktur macht die Testfälle leichter verständlich und wartbar. Testanalysten können zu diesem Zweck ein Glossar verwenden.
- **Genauigkeit**: Es sollte nur eine Interpretation eines Testfalls geben, um falsch-negative und falsch-positive Testergebnisse zu vermeiden. Uneindeutige Begriffe wie "geeignet", "nach Bedarf" oder "mehrere" sollten vermieden werden.
- **Vollständigkeit**: Alle notwendigen Attribute (z. B. siehe *ISO/IEC/IEEE 29119-3*, 2021) sollten vorhanden sein, einschließlich der erforderlichen Testdaten (siehe *Abschnitt 1.3.5*) und eines eindeutigen erwarteten Ergebnisses, um Zweifel beim Vergleich mit dem Istergebnis zu vermeiden.
- **Prägnanz**: Die Granularität der Testfälle (d. h. ein großer Testfall mit vielen Testaktionen gegenüber mehreren kleineren Testfällen) sollte der Testbasis und den Testbedingungen entsprechen. Kleinere Testfälle, die sich auf wenige Überdeckungselemente konzentrieren, sind vorzuziehen, da sie die Identifikation der Ursache von Fehlerwirkungen vereinfachen, flexibel zu Testabläufen und Testsuiten kombinierbar sind und ein Fehlschlag bei ihrer Ausführung keine weiteren Tests blockieren würde.

Das Format und der Detaillierungsgrad der Testfälle hängen vom Projekt- und Produktkontext ab und sollten innerhalb des Testteams vereinbart werden.

1.3.3 Anforderungen an die Testumgebung

Die Testumgebung ist ein kritischer Erfolgsfaktor sowohl für die manuelle als auch für die automatisierte Testausführung. Die Implementierung der Testumgebung wirkt sich auf die Testbarkeit, die Erkennung von Fehlerzuständen, die gesamten Testkosten und die Zuverlässigkeit der Testergebnisse aus. Ein Testfall, der in der Testumgebung bestanden oder fehlgeschlagen ist, sollte bei der Ausführung in der Produktion das gleiche Testergebnis liefern. Im Idealfall ist eine Testumgebung robust, vorhersehbar und bei Bedarf in ein Testautomatisierungsframework integriert. Testanalysten können die Anforderungen an die Testumgebung auf Grundlage der Analyse nachfolgender Aspekte während des Testentwurfs definieren:

- Testbedingungen, Testfälle und Anforderungen an Testdaten, damit die Testumgebung die Vorbedingungen für die Testdurchführung erfüllt,
- Teststufen und Testarten, die einen Kompromiss zwischen Flexibilität der Testumgebung und der Ähnlichkeit mit der Produktivumgebung finden müssen,
- Verfügbarkeit und Unabhängigkeit von Komponenten und Systemen, was den Einsatz von Testdoubles (z. B. Platzhalter oder Treiber) erforderlich machen kann.

Die Anforderungen an die Testumgebung beschreiben die Elemente der Testumgebung, die in verschiedene Kategorien eingeteilt werden können, wie Hardware, Middleware, Software, virtualisierte Dienste, Netzwerk, Schnittstellen, Werkzeuge, Sicherheit (Security), Konfiguration und Standort. Jede Anforderung für ein Testumgebungselement sollte die folgenden Informationen enthalten (*ISO/IEC/IEEE 29119-3*, 2021):

- Eindeutige Kennung – dient der Verfolgbarkeit
- Beschreibung – ausreichend detailliert, um die Anforderungen zu erfüllen
- Zuständigkeit – beschreibt, wer für die Bereitstellung verantwortlich ist
- Bedarfszeitraum – gibt an, wann und wie lange das Element benötigt wird
- Genauigkeit – der Grad, zu dem dieses Element die Produktivumgebung repräsentiert oder von ihr abweicht

Die Anforderungen sollten auch die übergreifenden Bedarfe der Testumgebung berücksichtigen, einschließlich Einrichtung, Sicherung und Wiederherstellung, Sicherheitsanforderungen, die Möglichkeit, die Testumgebung zu ändern, sowie Rollen und Berechtigungen (Koomen u. a., 2008).

Testanalysten sollten die Anforderungen an die Testumgebung klar, kompakt und kohärent dokumentieren und können dafür Diagramme oder Tabellen verwenden. Um Redundanzen und unnötige Dokumentation zu vermeiden, können Testanalysten auf bestehende Testumgebungen verweisen und sich auf die spezifischen Anforderungen der Teststufe konzentrieren. Die relevanten Stakeholder (z. B. Entwickler, technische Testanalysten, Testautomatisierungsentwickler, Geschäftsanalysten, Sponsoren und Produktverantwortliche) sollten die Anforderungen an die Testumgebung ebenfalls überprüfen, genehmigen und aktualisieren.

1.3.4 Bestimmung von Testorakeln

Ein Testorakel wird benötigt, um die erwarteten Ergebnisse für den dynamischen Test zu ermitteln. Im Idealfall stellt die Testbasis die Testorakel zur Verfügung (z. B. in einer textlichen oder formalen Spezifikation). Auch menschliche Erfahrung oder Wissen über das Testobjekt kann als Testorakel dienen. Ein automatisiertes Testorakel kann aus Gründen der Kosteneffizienz erforderlich sein (z. B. wenn Eingaben automatisch generiert werden oder menschliche Orakel kostspielig sind).

Abhängig von der Qualität und Vollständigkeit der Testbasis oder den Systemeigenschaften kann es sein, dass ein kostengünstiges Testorakel nicht zur Verfügung steht. Dies wird als *Testorakelproblem* bezeichnet. Typische Faktoren, die zum Testorakelproblem beitragen, sind datenbezogene Komplexität, Nichtdeterminismus (z. B. KI-basierte Systeme), probabilistisches Verhalten und fehlende oder mehrdeutige Anforderungen.

Einige bekannte Lösungen für das Testorakelproblem sind (Barr u. a., 2014):

- Pseudo-Orakel sind unabhängig entwickelte Systeme, die dieselbe Spezifikation wie das Testobjekt erfüllen (z. B. Altsysteme, vereinfachte Versionen von Testobjekten). Die Entwicklung eines Pseudo-Orakels für ein komplexes Testobjekt kann teuer sein, ist aber typisch für kritische Systeme.
- Modellbasiertes Testen kann das Testorakel als Teil des Testmodells formalisieren. Es ermöglicht die Generierung von erwarteten Ausgaben und die Ableitung von Tests aus dem Modell. Verhaltensbasierte Testverfahren schließen oft die Implementierung eines automatisierten Testorakels mit ein (siehe *Abschnitt 3.2.2* und *Abschnitt 3.2.3*).
- Der Propert-based Test verwendet spezifizierte Eigenschaften des Testobjekts, um Relationen zwischen der Eingabe und dem erwarteten Ergebnis der einzelnen Testfälle zu verifizieren. Wird eine solche Relation nicht erfüllt, schlägt der Testfall fehl. Diese Lösung eignet sich gut für die Testautomatisierung, aber ihre Effektivität hängt von den Relationen ab, die unter Umständen schwer zu bestimmen sind.
- Der metamorphe Test (siehe *Abschnitt 3.3.2*).
- Menschliche Orakel nutzen die Fähigkeiten von Menschen, um die erwarteten Ergebnisse zu bestimmen. Menschliche Ressourcen können kostspielig und knapp sein. Bei einigen Testansätzen (z. B. exploratives Testen) wird jedoch ein [menschliches Orakel]{.index] bevorzugt.

Assertions können in den Code für die Testautomatisierung oder in das Testobjekt selbst eingebaut werden, um ein automatisiertes Testorakel zu implementieren. Assertions sind ausführbare Anweisungen, die den Zustand oder das Verhalten des Testobjekts verifizieren. Wenn sie in das Testobjekt eingebaut sind, überprüfen sie in der Regel nur das, was für die Fortführung der Aufgabe erforderlich ist.

1.3.5 Anforderungen an die Testdaten

Während des Testentwurfs identifizieren und fordern Testanalysten Testdaten an, die für die Testausführung vorbereitet oder bereitgestellt werden müssen. Dabei werden unter anderem Zweck, Format und Nutzungskontext berücksichtigt. Die Norm (*ISO/IEC/IEEE 29119-3*, 2021, Abschnitt 8.5) beschreibt auch die Testdatenanforderungen. Die wichtigsten Aspekte sind:

- **Ähnlichkeit mit Produktivdaten**: Produktivdaten spiegeln reelle Daten wider, sind aber möglicherweise nicht sehr abwechslungsreich. Synthetische Daten ermöglichen eine kontrollierte Variabilität der Testdaten. Synthetische Daten sollten wichtige Aspekte wie Muster, Verteilungen und Ausreißer von Produktivdaten widerspiegeln, können aber bestimmte Fehlerzustände übersehen, die nur bei Produktivdaten auftreten. Bei Systemen, für die keine Produktivdaten vorliegen, müssen synthetische Daten realistische geschäftliche und technische Szenarien widerspiegeln. Personas helfen dabei, indem sie realistische, benutzerzentrierte Profile liefern, die die Erstellung von Daten leiten, um verschiedene Szenarien und Verhaltensweisen der Benutzer widerzuspiegeln.
- **Vertraulichkeit**: Sensible Testdaten (z. B. personenbezogene Daten) müssen geschützt werden. Pseudonymisierte Daten ersetzen persönliche Informationen durch künstliche Identifikatoren. Anonymisierte Daten entfernen identifizierbare Informationen über Einzelpersonen. Falls erforderlich müssen Testanalysten Datenschutzbestimmungen wie die DSGVO (Datenschutz-Grundverordnung) in der Europäischen Union (Kommission, 2016) oder den HIPAA (Health Insurance Portability and Accountability Act) in den USA (Health u. a., 2024) beachten.
- **Zweck**: Testdaten sind entscheidend für die Bestimmung von Vorbedingungen und erwarteten Ergebnissen, die sich auf den Systemzustand und seine Konfiguration auswirken (z. B. Systemzeit und

Datum). Dazu gehören auch die Festlegung von Benutzerberechtigungen und die Herstellung von Beziehungen zwischen Produkten, Abteilungen und Kategorien.

- **Überdeckungskriterien**: Die Testdaten müssen sich an den Überdeckungskriterien des gewählten Testverfahrens ausrichten. Neben gültigen Testdaten kann dies auch ungültige Testdaten erfordern, z. B. für Negativtests.
- **Datenformat**: Systematisches Datenmanagement (z. B. bei API-Tests) kann strukturierte Daten (z. B. CSV, JSON, XML oder Datenbank) erfordern.
- **Verfolgbarkeit**: Verfolgbarkeit gewährleistet die Wartbarkeit der Testdaten bei Änderungen an Testfällen.
- **Wartbarkeit**: Hart kodierte Testdaten in konkreten Testfällen sollten vermieden werden. Dadurch wird die Erkennung von Fehlerzuständen und die Wartbarkeit der Testfälle unterstützt. Testfälle sollten Testlogik und Testdaten voneinander trennen (siehe *Abschnitt 1.3.2*).
- **Abhängigkeiten**: Abhängige Daten erfordern eine Abfolge von Schritten, um sie zu erstellen.
- **Verfügbarkeit**: Dienst-Virtualisierung (service virtualization) kann die Bereitstellung von Daten simulieren, die von nicht verfügbaren oder nicht zugänglichen Diensten oder externen Systemen bereitgestellt werden.
- **Zeitabhängigkeit und Datenalterung**: Testfälle mit veralteten oder zeitabhängigen Daten können das Systemverhalten auf unerwartete oder inkorrekte Weise beeinflussen.

1.3.6 Den schlüsselwortgetriebenen Test zur Entwicklung von Testskripten anwenden

Wenn der schlüsselwortgetriebene Test angewendet wird, erstellen Testanalysten Testskripte unter Verwendung von Schlüsselwörtern. Die Implementierung der Testskripte ist Aufgabe von technischen Testanalysten, Testautomatisierern oder Entwicklern.

Testanalysten identifizieren und spezifizieren Schlüsselwörter durch Analyse der Testbasis oder durch Zusammenarbeit mit den Stakeholdern. Schlüsselwörter lassen sich in zwei Kategorien einteilen: Aktion und Verifizierung. Aktionsschlüsselwörter interagieren mit dem Testobjekt (z. B. Ausführen von Funktionen, Übermitteln von Daten, Navigieren innerhalb des Testobjekts), der Testumgebung (z. B. Einrichten von Konfigurationen und Aktivieren von Simulatoren) oder anderen Komponenten oder Systemen (z. B. Aufrufen einer Schnittstelle des Testobjekts). Verifizierungsschlüsselwörter repräsentieren Assertions, die bewerten, ob das vom Testobjekt erzeugte Istergebnis mit dem erwarteten Ergebnis übereinstimmt.

Schlüsselwörter befinden sich auf mindestens zwei Abstraktionsebenen: der Anwendungsschicht und der Testschnittstellenschicht. Schlüsselwörter der Anwendungsschicht entsprechen geschäftsbezogenen Aktionen und spiegeln die Terminologie des Anwendungsbereichs wider. Sie abstrahieren von den technischen Details der Schnittstellen des Testobjekts. Schlüsselwörter der Testschnittstellenschicht kommunizieren über die Testschnittstelle mit den Testobjekten, Testumgebungselementen oder anderen Komponenten oder Systemen. Sie befinden sich auf der untersten Abstraktionsebene. Zusätzliche Zwischenschichten können die Wartbarkeit der Schlüsselwörter unterstützen.

Schlüsselwörter können atomar oder aus mehreren Schlüsselwörtern zusammengesetzt sein. Struktur und Abstraktionsebene eines Schlüsselworts sind unabhängige Attribute, zusammengesetzte Schlüsselwörter befinden sich jedoch häufig auf einer höheren Abstraktionsebene, während atomare Schlüsselwörter eher auf der Testschnittstellenschicht liegen.

Zu den Aufgaben der Testanalysten beim schlüsselwortgetriebenen Test gehören:

- Festlegen von Schlüsselwörtern und deren Parametern
- Spezifizierung von Schlüsselwort-Testfällen, d. h. Testskripten mit Schlüsselwörtern
- Spezifizierung zusätzlicher Schritte zu Testskripten unter Verwendung von Schlüsselwörtern, wie Vorbedingungen, Verifizierungen und Aufräumen der Testumgebung nach dem Test
- Pflege von Schlüsselwort-Testfällen, um Änderungen am Testobjekt zu berücksichtigen
- Ausführen von Schlüsselwort-Testfällen, entweder automatisiert oder manuell
- Analyse fehlgeschlagener Schlüsselwort-Testfälle, um die Ursache der Fehlerwirkung zu ermitteln

Bei der Analyse der Testbasis suchen die Testanalysten nach Interaktionen zwischen dem Testobjekt und seiner Umgebung (z. B. Benutzer, andere Systeme und Geräte). Ein Beispiel ist die User-Story "Als Mitglied möchte ich mich authentifizieren, damit ich Zugang zu den Einrichtungen erhalte" mit den Akzeptanzkriterien "Gültige Mitgliedskarten können zur Authentifizierung verwendet werden". Testanalysten können ein Aktionsschlüsselwort "Mitglied authentifizieren" mit dem Parameter "Mitgliedskarte" und ein Verifizierungsschlüsselwort "Zugang verifizieren" spezifizieren. Zudem prüfen Testanalysten, ob Schlüsselwörter sich auf der entsprechenden Abstraktionsebene befinden.

Bei der Spezifikation von Schlüsselwörtern müssen die Testanalysten darauf achten, dass Schlüsselwörter

- ein Verb (Aktion) und ein Substantiv (Objekt) enthalten, beispielsweise "Mitglied authentifizieren" oder "Authentifiziere Mitglied",
- in einheitlichem Stil formuliert sind,
- in ihrer Bedeutung eindeutig sind,
- angemessen dokumentiert sind,
- das Vokabular des Anwendungsbereichs widerspiegeln und
- wiederverwendbar sind.

In Deutsch ist der Formulierungsstil mit der Infinitivform des Verbs üblich, wie z. B. "Mitglied authentifizieren".

Beim Verfassen von Schlüsselwort-Testfällen können Testanalysten fehlende Schlüsselwörter erkennen und diese unmittelbar spezifizieren. Schlüsselwort-Testfälle können in verschiedenen Formaten geschrieben werden, z. B. als Listen oder Tabellen.

Schlüsselwörter können sich während eines Projekts ändern und sind anfällig für redundante Spezifikationen (Rwemalika u. a., 2019). Die in diesem Abschnitt genannten Empfehlungen zielen darauf ab, Redundanz zu vermeiden und den Wartungsaufwand zu verringern.

Obwohl der schlüsselwortgetriebene Test ein Ansatz zur Testautomatisierung ist, kann auch das manuelle Testen davon profitieren. Wird er für das manuelle Testen eingesetzt, unterstützt er effektiv den späteren Übergang vom manuellen zum automatisierten Testen.

Weitere Einzelheiten zur Automatisierung der Testdurchführung und zum schlüsselwortgetriebenen Test sind in (ISTQB-TTA, v4.0 DE), (ISTQB-TAE, v2.0 DE) und (*ISO/IEC/IEEE 29119-5*, 2016) zu finden.

1.3.7 Werkzeuge für die Verwaltung der Testmittel

Eine sachgerechte werkzeuggestützte Verwaltung der Testmittel kann den Testanalysten helfen, den gesamten Testprozess zu unterstützen. Werkzeuge können den Status von Arbeitsergebnissen bereitstellen und die Testüberwachung und Teststeuerung unterstützen. Treten Fehlerwirkungen im SUT auf, können Testanalysten auf die Werkzeuge zurückgreifen, um die Testergebnisse aus früheren Testläufen zu überprüfen und den Zeitpunkt zu analysieren, an dem Fehlerzustände aufgetreten sind.

Zu den wichtigsten Werkzeugarten für die Verwaltung der Testmittel gehören:

- **Testmanagementwerkzeuge** zur Bereitstellung einer strukturierten Ablage für alle relevanten Testmittel (z. B. Testbedingungen, Testfälle, Testskripte, Testsuiten und Testläufe). Diese Werkzeuge erleichtern die Verfolgbarkeit (z. B. über die Verfolgbarkeitsmatrix), das Abrufen von Testfällen, die Planung von Testläufen, die Aufzeichnung von Testergebnissen und die Gesamtberichterstattung über Testfortschritt und Qualität.
- **Fehlermanagementwerkzeuge** zur Protokollierung, Priorisierung und Überwachung der Fehlerbehebung.
- **Testdatenmanagementwerkzeuge** zur Erstellung und Pflege von Testdaten, einschließlich des Schutzes sensibler Daten (siehe *Abschnitt 1.3.5*).
- **Konfigurationsmanagementwerkzeuge** zur Erleichterung testbezogener Aktivitäten in den Entwicklungs-, Freigabe- und Betriebsprozessen, einschließlich der Verwaltung der Konfiguration und Verfügbarkeit von Testumgebungen.
- **Anforderungsmanagementwerkzeuge** zur Erfassung und Verfolgung von Anforderungen auf hoher Ebene, um sicherzustellen, dass diese klar definiert, versioniert und während des gesamten SDLC nachverfolgt werden.

Testanalysten unterstützen die Verwaltung von Testmitteln durch:

- Analyse des Projekts und des Releases, um die richtige Teilmenge an Testmitteln auszuwählen (z. B. eine Spezifikation, die durch ihre Version identifiziert wird, so dass sie mit der Version des SUT übereinstimmt)
- Definition einer geeigneten funktionalen (z. B. nach Features oder Modulen) oder technischen Struktur (z. B. nach Testart oder Testumgebung) für die Organisation der Testfälle im Testmanagementwerkzeug
- Hinzufügen von Metadaten zu den Testfällen (z. B. Informationen über den Aufwand, der mit der Testdurchführung verbunden ist, oder die erforderliche Testumgebung)
- Sicherstellung der Verfolgbarkeit zwischen Anforderungen, Testbedingungen, Tests, Testläufen und Fehlerzuständen
- Auswahl der richtigen Testsuiten für Regressionstests (für manuelle/automatisierte Testausführung)
- Konfigurationsmanagement der Testfälle, einschließlich der Identifizierung veralteter Testfälle.

Werkzeuge helfen den Testprozess zu organisieren, zu verfolgen und seine Qualität zu sichern. Testanalysten unterstützen diese Bemühungen durch die Auswahl, Strukturierung und Wartung von Testfällen, die Gewährleistung der Wartbarkeit und die Verwaltung von Testfallversionen für effizientes Testen und Teststeuerung.

2 Die Aufgaben des Testanalysten beim risikobasierten Testen – 90 Minuten

Schlüsselbegriffe

Auswirkungsanalyse, Produktrisiko, Regressionstest, Risikoanalyse, risikobasierter Test, Risikobewertung, Risikoidentifizierung, Risikominderung, Risikosteuerung, Risikoüberwachung

Lernziele für Kapitel 2:

Die Lernenden können ...

2.1 Risikoanalyse

TA-2.1.1 (K2) ... den Beitrag von Testanalysten zur Risikoanalyse des Produktrisikos zusammenfassen

2.2 Risikosteuerung

TA-2.2.1 (K4) ... die Auswirkungen von Änderungen analysieren, um den Testumfang von Regressionstests zu bestimmen

Einführung in die Aufgaben des Testanalysten beim risikobasierten Testen

Der risikobasierte Test ist ein Testansatz, bei dem der Testaufwand auf Grundlage der Risikostufen der Testelemente priorisiert wird. Der Testmanager bestimmt diesen Testansatz. Die Testanalysten spielen eine aktive Rolle bei der Umsetzung. Der risikobasierte Test wird in (ISTQB-TM, v3.0 DE) ausführlicher behandelt.

Der risikobasierte Test umfasst die Risikoanalyse und die Risikosteuerung (ISTQB-CTFL, v4.0.2 DE, Abschnitt 5.2).

Da Risiken und Risikostufen nicht statisch sind und sich im Laufe der Zeit ändern, beinhaltet der risikobasierte Test außerdem eine regelmäßige Risikoüberwachung. In einem iterativen Lebenszyklus wird dies mit einer vom Team festgelegten Häufigkeit durchgeführt (wahrscheinlich einmal pro Iteration). In anderen Lebenszyklen wird die Häufigkeit von der für das Produktrisikomanagement verantwortlichen Person, häufig dem Testmanager, festgelegt. Die Testanalysten tragen dazu bei, indem sie das Risikoregister auf der Grundlage der vorgenommenen Änderungen aktualisieren und die Risikominderungsmaßnahmen anpassen.

2.1 Risikoanalyse

Die Risikoanalyse umfasst die Risikoidentifizierung und die Risikobewertung. Dieser Lehrplan konzentriert sich darauf, wie Testanalysten an den Risikoanalyseaktivitäten teilnehmen sollten, um sicherzustellen, dass der risikobasierte Test korrekt umgesetzt wurde.

Testanalysten verfügen oft über einzigartige Kenntnisse des Systems sowie Erfahrung und intuitives Verständnis dafür, was typischerweise schiefgeht, welche Auswirkungen dies hat und wie Testen das Risiko mindern kann. Dies macht die Testanalysten zu wertvollen Stakeholdern bei der Produktrisikoanalyse.

Bei der **Risikoidentifizierung** tragen Testanalysten mit ihrer eigenen Erfahrung und ihrem Wissen zu Retrospektiven, Risikoworkshops, Brainstorming und der Erstellung von Checklisten bei. Testanalysten können auch Interviews mit Stakeholdern durchführen, um besser zu verstehen, was aus deren Sicht die bedeutendsten Risiken sind.

Während der **Risikobewertung** tragen Testanalysten zusammen mit anderen Stakeholdern zur Bestimmung der Risikostufe bei, indem sie verschiedene Faktoren einschätzen, wie z. B.:

- Häufigkeit der Nutzung und Kritikalität der betroffenen Features
- Kritikalität der betroffenen Geschäftsziele
- Finanzielle, ökologische und reputationsbezogene Schäden
- Qualität der Testbasis
- Rechtliche oder sicherheitsrelevante (safety) Anforderungen

Testanalysten helfen außerdem dabei, Produktrisiken nach den betroffenen Qualitätsmerkmalen zu kategorisieren, z. B. anhand des Produktqualitätsmodells von *ISO/IEC 25010* (2023). Die Risikostufe ist oft nicht gleichmäßig über das Testobjekt verteilt. In solchen Fällen sollten Testanalysten das Testobjekt in Testelemente (z. B. Komponenten, Schnittstellen und Features) aufschlüsseln und das jeweilige Risiko für jedes Testelement separat bewerten.

Abschließend schlagen Testanalysten im Rahmen der Produktrisikobewertung geeignete Testaktivitäten zur Minderung jedes identifizierten Produktrisikos vor. Diese Aktivitäten können statische Tests und dynamische Tests umfassen. Abhängig von Faktoren wie der Risikostufe, der zugehörigen Art des Testelements und dem betroffenen Qualitätsmerkmal geben die Testanalysten die erforderlichen Teststufen, Testarten, Testverfahren, den Grad der Unabhängigkeit des Testens und die Gründlichkeit der Tests an. Im Sinne des Shift-Left-Ansatzes zeigen die Testanalysten auf, mit welchen Testaktivitäten das Risiko möglichst frühzeitig gemindert werden kann, um den Testaufwand zu minimieren.

2.2 Risikosteuerung

Die Risikosteuerung umfasst die Risikominderung und die Risikoüberwachung. Testanalysten verstehen die Funktionalitäten des Systems und die damit verbundenen potenziellen Risiken. Daher ist die Rolle der Testanalysten bei mehreren Risikominderungsmaßnahmen entscheidend, die in (ISTQB-CTFL, v4.0.2 DE, Abschnitt 5.2.4) erwähnt werden:

- Die Durchführung von Reviews wird in *Abschnitt 5.2.2* dieses Lehrplans behandelt.
- Die Anwendung geeigneter Testverfahren und Überdeckungsgrade wird in *Abschnitt 3.5* besprochen.
- Die Anwendung der geeigneten Testarten wird in Kapitel 4 besprochen.
- Die Durchführung von Regressionstests wird in diesem Kapitel besprochen.

Das Hauptziel von Regressionstests ist es, das Vertrauen in die Qualität des Testobjekts nach einer Änderung sicherzustellen. Es kann jedoch aufgrund von Einschränkungen (z. B. Zeit, Budget, Testumgebung oder Testdaten) unmöglich sein, alle Regressionstests auszuführen. Dies gilt vor allem für manuell ausgeführte Tests, aber auch für automatisierte Tests, z. B. wenn die Testzyklen kurz sind und es viele langlaufende automatisierte Tests gibt. Dies macht es notwendig, geeignete Regressionstests nach bestimmten Kriterien auszuwählen. Es ist unerlässlich, den Umfang der Regressionstests bei jedem Testzyklus zu überprüfen. Das Review kann zu dem Ergebnis führen, dass die bestehende Regressionstestsuite angepasst werden muss.

Das zuverlässigste Verfahren zur Auswahl automatisierter Tests ist eine Auswirkungsanalyse, die von Werkzeugen unterstützt werden kann. Solche Werkzeuge basieren auf einem automatisierten Konfigurationsmanagementsystem. Sie registrieren, welche Konfigurationselemente während der Ausführung der einzelnen Testfälle aktiviert werden. Bei einer Änderung verfolgen sie, welche Konfigurationselemente geändert wurden, und wählen Regressionstests aus, die mit den geänderten Elementen interagieren. Auf diese Weise stellen die Werkzeuge sicher, dass sich die Tests nach einer Änderung eines Elements auf die Bereiche konzentrieren, in denen am ehesten eine Fehlerwirkung zu erwarten ist (Juergens u. a., 2018).

Was die manuelle Testdurchführung betrifft, so gibt es keinen schlüssigen Beweis für ein eindeutig überlegenes Verfahren bei der [Auswahl von Regressionstests], da die Ergebnisse von verschiedenen Faktoren abhängen (Engström u. a., 2010). Daher müssen Testanalysten je nach der gegebenen Situation entscheiden, welches Verfahren anzuwenden ist. Häufig eingesetzte Verfahren sind unter anderem:

- **Risikobasierte Testauswahl**, bei der die Testanalysten die Verfolgbarkeit der Regressionstestsuite mit einem Risikoregister sicherstellen. Wenn eine Änderung vorgenommen und das Risikoregister entsprechend aktualisiert wird, passen die Testanalysten die Regressionstestsuite an, um die höchsten Risikostufen zu überdecken.

- **Historienbasierte Testauswahl**, bei der die Testanalysten frühere Testdurchführungen analysieren und ermitteln, welche Tests Fehlerzustände aufgedeckt haben oder empfindlich auf ähnliche Änderungen reagiert haben, wie die seit der letzten Testdurchführung vorgenommenen. Die erneute Ausführung der entsprechenden Tests als Teil des Regressionstests erhöht die Wahrscheinlichkeit, dass ähnliche Fehlerzustände aufgedeckt werden. Zusätzlich können die Testanalysten auch einige Tests einbeziehen, die seit längerer Zeit nicht mehr ausgeführt wurden, um sicherzustellen, dass das Testobjekt sie weiterhin erfolgreich besteht.
- **Überdeckungsbasierte Testauswahl**, bei der die Testanalysten eine kleine Anzahl von Tests auswählen, die auf der Grundlage der gewählten Testverfahren eine möglichst hohe Überdeckung erreichen. Die Anzahl der Tests muss sorgfältig mit der Erhöhung der Überdeckung pro Test abgewogen werden.
- **Die Verfolgbarkeitsmatrix für Anforderungen**, die dazu dient, die Auswirkungen von Änderungen der Anforderungen auf die zugehörigen Tests zu bewerten. Sie kann besonders wertvoll sein, wenn sich neue oder geänderte Anforderungen indirekt auf bestehende Features auswirken. Testanalysten wählen Regressionstests sowohl für die direkt betroffenen als auch für verwandte Features aus, um mögliche unbeabsichtigte Seiteneffekte zu überdecken. In der agilen Softwareentwicklung kann dies auf ähnliche Weise erfolgen, indem Tests ausgewählt werden, die die von neuen oder geänderten User-Storys betroffenen Akzeptanzkriterien überdecken.
- **Testauswahl basierend auf Nutzungsprofilen**, wobei die Testanalysten die auszuführenden Regressionstestfälle auf Grundlage der Nutzungsmuster des Testobjekts auswählen. Beim Testen eines Online-Shops kann ein solcher Test z. B. beinhalten, dass sich der Benutzer anmeldet, nach Produkten sucht, diese in den Warenkorb legt und eine Bestellung aufgibt. Wenn eine Anwendung erheblich geändert wird, bietet dieses Verfahren einen schnellen Überblick über die gesamte Systemfunktionalität. Wenn sich daraus zu viele Tests ergeben, priorisieren die Testanalysten kritische Nutzungsmuster, die häufig auftreten und kritische Funktionalitäten und Geschäftsprozesse überdecken.
- **Auswirkungsanalyse**, die auch für die Auswahl von Regressionstestfällen für die manuelle Ausführung verwendet werden kann, wenn die Testanalysten wissen, welche Regressionstestfälle mit den geänderten Konfigurationselementen interagieren.

Für eine umfassendere und effektivere Regressionstestsuite ist es oft notwendig, eine Kombination von Testverfahren zu verwenden. Testanalysten müssen jedoch sorgfältig abwägen zwischen dem Bedarf an Überdeckung und einer überschaubaren Größe der Testsuite. Nach jedem Testzyklus analysieren die Testanalysten die Testergebnisse, um die Effektivität der angewandten Verfahren zu ermitteln (siehe *Abschnitt 5.3.1*). Beim nächsten Mal behalten die Testanalysten die effektiven Verfahren bei und ersetzen die ineffektiven. Dadurch wird die Auswahl von Regressionstests im Laufe der Zeit kontinuierlich verbessert. Dies ist besonders wichtig bei iterativen und inkrementellen Entwicklungsmodellen mit häufigen Änderungen.

3 Testanalyse und Testentwurf – 615 Minuten

Schlüsselbegriffe

Äquivalenzklassenbildung,checklistenbasierter Test, CRUD-Test, datenbasiertes Testverfahren, Entscheidungstabellentest, erfahrungsbasierter Test, kombinatorischer Test, Massentest, metamorphe Relation, metamorpher Test, regelbasiertes Testverfahren, sitzungsbasierter Test, szenariobasierter Test, Test-Charta verhaltensbasiertes Testverfahren, Wertebereichstest, Zufallstest, Zustandsübergangstest

Lernziele für Kapitel 3:

Die Lernenden können ...

3.1 Datenbasierte Testverfahren

TA-3.1.1 (K3) ... den Wertebereichstest anwenden

TA-3.1.2 (K3) ... den kombinatorischen Test anwenden

TA-3.1.3 (K2) ... die Vorteile und Grenzen von Zufallstests zusammenfassen

3.2 Verhaltensbasierte Testverfahren

TA-3.2.1 (K2) ... den CRUD-Test erläutern

TA-3.2.2 (K3) ... den Zustandsübergangstests anwenden

TA-3.2.3 (K3) ... den szenariobasierten Test anwenden

3.3 Regelbasierte Testverfahren

TA-3.3.1 (K3) ... den Entscheidungstabellentest anwenden

TA-3.3.2 (K3) ... den metamorphen Test anwenden

3.4 Erfahrungsbasiertes Testen

TA-3.4.1 (K3) ... Test-Chartas für den sitzungsbasierten Test vorbereiten

TA-3.4.2 (K3) ... Checklisten für den erfahrungsbasierten Test vorbereiten

TA-3.4.3 (K2) ... Beispiele für die Vorteile und Grenzen von Massentests geben

3.5 Anwenden der am betsen geeigneten Testverfahren

TA-3.5.1 (K4) ... geeignete Testverfahren zur Minderung von Produktrisiken in einer bestimmten Situation auswählen

TA-3.5.2 (K2) ... die Vorteile und Risiken der Automatisierung des Testentwurfs erläutern

Einführung in Testanalyse und Testentwurf

Testverfahren werden hauptsächlich in der Testanalyse und dem Testentwurf eingesetzt. Die in diesem Kapitel behandelten Testverfahren umfassen Black-Box-Testverfahren und erfahrungsbasierte Testverfahren. White-Box-Testverfahren werden in (ISTQB-TTA, v4.0 DE) besprochen.

In diesem Lehrplan werden Black-Box-Testverfahren in drei Kategorien unterteilt, die auf den zugrundeliegenden Testbedingungen basieren und durch die nachfolgenden Elemente repräsentiert werden:

- Elemente der Daten (datenbasiert)
- Elemente des dynamischen Verhaltens (verhaltensbasiert)
- Elemente statischer Verhaltensregeln (regelbasiert)

3.1 Datenbasierte Testverfahren

Eingabedaten aus verschiedenen Wertebereichen führen zu verschiedenen Verhaltensweisen des Testelements. Datenbasierte Testverfahren zielen darauf ab, zu überprüfen, ob die Implementierung mit bestimmten Gebieten des Wertebereichs korrekt umgeht. Der Foundation-Level-Lehrplan (ISTQB-CTFL, v4.0.2 DE, Abschnitt 4.2) behandelt zwei Testverfahren, die als datenbasiert eingestuft werden können: Äquivalenzklassenbildung und Grenzwertanalyse. In diesem Lehrplan werden drei weitere datenbasierte Testverfahren vorgestellt:

- Wertebereichstest – erweitert die Äquivalenzklassenbildung und Grenzwertanalyse auf Bereiche mit mehreren Parametern und komplexen Äquivalenzklassen
- Kombinatorischer Test – konzentriert sich auf die Wechselwirkungen zwischen mehreren Parametern in einem mehrdimensionalen Wertebereich
- Zufallstest – wählt zufällige Eingaben aus dem Wertebereich auf der Grundlage einer bestimmten Wahrscheinlichkeitsverteilung aus.

Zu berücksichtigen ist, dass sich der Zufallstest im Allgemeinen sowohl auf zufällige Eingabedaten als auch auf zufällige Ereignisse beziehen kann. In diesem Lehrplan wird nur das Testen mit zufälligen Eingabedaten behandelt.

3.1.1 Wertebereichstest

Beim Wertebereichstest wird überprüft, ob sich das Testelement in den Äquivalenzklassen des Bereichs und an deren Grenzen wie spezifiziert verhält. In diesem Fall werden Äquivalenzklassen mithilfe von Ausdrücken definiert, die atomare Bedingungen mit logischen Operatoren (z. B. AND, OR und NOT) verknüpfen und eine oder mehrere interagierende Variablen umfassen. Jede atomare Bedingung definiert eine Grenze der Äquivalenzklasse. Geschlossene Grenzen werden durch relationale Ausdrücke mit den Operatoren $\leq$, $\geq$ oder = gebildet und offene Grenzen werden durch relationale Ausdrücke mit den Operatoren <, > oder $\neq$ gebildet.

Zum Beispiel definiert im Kontext der Klassifizierung des Body-Maß-Index (BMI) *Körpergröße > 1,29 Meter UND Gewicht / Höhe*^{2} *≥ 30* eine Äquivalenzklasse eines zweidimensionalen Wertbereichs der zwei Grenzen hat, eine offene und eine geschlossene, und aus zwei Variablen besteht.

Beim Wertebereichstest sollen Fehlerzustände in der Impplementierung der Äquivalenzklassen (z. B. falsche Operatoren oder Konstanten im Code) durch die Auswahl geeigneter Überdeckungselemente aufgedeckt werden. Die Überdeckungskriterien in Wertebereichstests beziehen sich auf die ON-, OFF-, IN- und OUT-Punkte:

- Bei einer geschlossenen Grenze liegt ein ON-Punkt auf dieser Grenze. Bei einer offenen Grenze liegt ein ON-Punkt innerhalb der Äquivalenzklasse und ist gemäß der angegebenen Genauigkeit der Grenze am nächsten.
- Bei einer geschlossenen Grenze liegt ein OFF-Punkt {.index} außerhalb der Äquivalenzklasse und liegt entsprechend der angegebenen Genauigkeit am nächsten zur Grenze. Bei einer offenen Grenze liegt ein OFF-Punkt auf dieser Grenze.
- Ein IN-Punkt liegt innerhalb der Äquivalenzklasse und ist kein ON-Punkt.
- Ein OUT-Punkt liegt außerhalb der Äquivalenzklasse und ist kein OFF-Punkt.

Jede dieser Punktarten hat einen Bezug zu der Grenze der Äquivalenzklasse. Ein und derselbe Punkt kann eine unterschiedliche Punktart für verschiedene Grenzen darstellen.

Zu den Überdeckungskriterien gehören:

Vereinfachte Wertebereichsüberdeckung (Jeng u. a., 1994), die die folgenden Überdeckungselemente erfordert:

- Ein ON-Punkt und ein OFF-Punkt für jede Grenze, die durch einen der Operatoren $<$, $\leq$, $>$ oder $\geq$ definiert ist.
- Ein ON-Punkt und zwei OFF-Punkte, die sich auf verschiedenen Seiten der Grenze für den Operator = befinden.
- Ein OFF-Punkt und zwei ON-Punkte, die sich auf verschiedenen Seiten der Grenze für den $\neq$-Operator befinden.

Jeder OFF-Punkt muss so nahe wie möglich am entsprechenden ON-Punkt liegen, entsprechend der vorgegebenen Genauigkeit.

Zuverlässige Wertebereichsüberdeckung (Forgács u. a., 2024), die die folgenden Überdeckungselemente erfordert:

- Ein ON-Punkt, ein OFF-Punkt, ein IN-Punkt und ein OUT-Punkt für jede Grenze, die durch einen der Operatoren $<$, $\leq$, $>$ oder $\geq$ definiert ist.
- Ein ON-Punkt und zwei OFF-Punkte, die sich auf verschiedenen Seiten der Grenze für den Operator = befinden.
- Ein OFF-Punkt und zwei ON-Punkte, die sich auf verschiedenen Seiten der Grenze für den $\neq$-Operator befinden.

Für beide Überdeckungskriterien gilt, dass die Anzahl der Überdeckungselemente linear von der Anzahl der Grenzen abhängt. Sie kann für beide Überdeckungskriterien optimiert werden, indem dasselbe Überdeckungselement für verschiedene Grenzen verwendet wird. Beispielsweise können alle Grenzen einer Äquivalenzklasse einen gemeinsamen IN-Punkt verwenden oder eine Kombination von OFF- und ON-Punkten einer Grenze einer Äquivalenzklasse kann als ON- und OFF-Punkte für die Grenze der benachbarten Äquivalenzklasse verwendet werden. Für Wertebereiche mit vielen Grenzen ist ein erweiterter Optimierungsalgorithmus verfügbar (Forgács u. a., 2024).

Die zuverlässige Wertebereichsüberdeckung führt zu einer etwas größeren Anzahl von Überdeckungselementen als die vereinfachte Wertebereichsüberdeckung, kann aber wesentlich mehr Fehlerzustände aufdecken (*Site of Software Test Design*, 2020).

Wertebereichstests können auf jeder Teststufe angewendet werden. Sie verallgemeinern Grenzwertanalyse und Äquivalenzklassenbildung (siehe ISTQB-CTFL, v4.0.2 DE, Abschnitt 4.2) für komplexere Wertebereiche. Weitere Testansätze zum Wertebereichstest finden sich in Lehrbüchern wie (Beizer, 1990, Kap. 6; Binder, 2000, Abschnitt 10.2.4; Kaner; Padmanabhan u. a., 2013; Jorgensen, 2014, Kap. 5).

3.1.2 Kombinatorischer Test

Einige Softwarefehler entstehen durch bestimmte Kombinationen von Parameterwerten, auch Interaktionsfehler genannt. Kombinatorische Tests zielen darauf ab, solche Fehlerwirkungen aufzudecken, indem diese Parameterwertekombinationen untersucht werden.

Bei kombinatorischen Tests kombinieren die Testbedingungen in der Regel Konfigurationsparameter oder Eingabewerte.

Daher gibt es zwei Hauptansätze für kombinatorische Tests. Der erste Ansatz verwendet Kombinationen von Konfigurationsparametern, und jede dieser Kombinationen kann mit demselben Testfall getestet werden. Beim zweiten Ansatz werden Kombinationen von Eingabewerten genutzt, aus denen vollständige Testfälle entstehen. Diese Testfälle bilden gemeinsam eine Testsuite für das SUT (Kuhn; Yu u. a., 2013).

Ein spezifisches Parameter-Wert-Paar besteht aus einem Parameter und seinem Wert (z. B. "(Farbe, rot)").

Zu den kombinatorischen Überdeckungskriterien gehören die folgenden (Ammann u. a., 2008), (Forgács u. a., 2019):

Die **Basisauswahlüberdeckung** geht davon aus, dass manche Parameter-Wert-Kombinationen wichtiger sind als andere. Deshalb wird für jeden Parameter ein Basis-Parameter-Wert-Paar bestimmt. Die Kombination aller Basis-Parameter-Wert-Paare bildet das Basis-Überdeckungselement. Weitere Überdeckungselemente werden erstellt, indem man ein Basis-Parameter-Wert-Paar durch einen anderen Wert des Parameters ersetzt, bis keine weiteren Kombinationen mehr gebildet werden können.

Bei der **paarweisen Überdeckung** sind die Überdeckungselemente Paare von Parameter-Wert-Paaren für zwei beliebige Parameter. Es gibt Werkzeuge zur Erzeugung dieser Überdeckungselemente, dennoch ist es generell schwierig, eine minimale Menge von Testfällen zu finden, die eine paarweise Überdeckung erreichen.

Bei Parametern mit vielen Werten kann zunächst die Äquivalenzklassenbildung angewendet werden, um die Anzahl und Menge der resultierenden Kombinationen zu reduzieren. Die Erfassung der Parameter und ihrer Werte in einem Klassifikationsbaum oder einem Feature-Modell (siehe IREB-Glossary, 2024) unterstützt diesen Vorgang. Die endgültige Anzahl der Testfälle kann durch Einschränkungen zwischen Parameter-Wert-Paaren, durch manuell hinzugefügte Kombinationen, die als problematisch bekannt sind, oder durch ungültige oder nicht durchführbare Kombinationen beeinflusst werden.

Die entscheidende Erkenntnis des kombinatorischen Tests ist, dass nicht jeder Parameter zu einer Fehlerwirkung beiträgt und dass die meisten Fehlerwirkungen durch einen einzigen Parameterwert oder durch Wechselwirkungen zwischen einer relativ kleinen Anzahl von Parametern ausgelöst werden (Cohen u. a., 1994). Dies stimmt mit der Hypothese des Kopplungseffekts überein, die besagt, dass die Erkennung einfacher Fehlerzustände in einem Programm oft auch komplexe Fehlerzustände aufdecken kann (Offutt, 1992). In einer begrenzten Studie (Kuhn; Wallace u. a., 2004) zeigten die Ergebnisse, dass etwa 97 % der

Fehlerwirkungen durch nur eine oder zwei interagierende Bedingungen verursacht werden, was darauf hindeutet, dass paarweises Testen ein effektives Testverfahren ist.

Weitere Informationen zum kombinatorischen Test, einschließlich anderer Überdeckungskriterien wie diff-pair-t oder n-weise Überdeckung, sind in (Ammann u. a., 2008), (Forgács u. a., 2019) zu finden. Werkzeuge zum kombinatorischen Testen sind auch unter (Czerwonka, 2004) verfügbar.

3.1.3 Zufallstest

Beim Zufallstest werden Testdaten nach dem Zufallsprinzip aus dem Eingabebereich des Testelements auf Grundlage einer bestimmten Wahrscheinlichkeitsverteilung ausgewählt. Für Zwecke der Validierung wird eine Verteilung basierend auf Nutzungsprofilen empfohlen. Für Zwecke der Verifizierung sollte die Verteilung nutzungsunabhängig sein, um Verzerrungen zu vermeiden. Die erwarteten Ergebnisse können den Testfällen mithilfe eines Testorakels hinzugefügt werden, was in den meisten Fällen ein automatisiertes Testorakel erfordert.

Der Zufallstest kann geleitet oder ungeleitet sein. Bei ungeleiteten Zufallstests bleibt die Wahrscheinlichkeitsverteilung während des gesamten Prozesses unveränderbar. Bei geleiteten Zufallstests, z. B. bei Verfahren wie dem adaptiven Zufallstests (Huang u. a., 2019), wird die Verteilung auf Grundlage zuvor ausgewählter Werte angepasst und entwickelt sich im Laufe der Zeit weiter. Geleitete Zufallstests zielen darauf ab, den Eingabebereich effektiv zu überdecken, wobei Fehlerzustände oftmals in bestimmten Bereichen gehäuft auftreten.

Für Zufallstests gibt es keine anerkannten Überdeckungskriterien. Daher können sich die Endekriterien nur auf die Anzahl der durchgeführten Tests, die Testzeit oder ähnliche Messungen des Abschlusses stützen.

Zufallstests sind besonders wertvoll, wenn die Fachkenntnisse im Anwendungsbereich begrenzt sind oder eine große Menge an Testdaten benötigt wird. Sie sind kosteneffizient und bieten, in probabilistischer Hinsicht, Einblicke in die Zuverlässigkeit von Testobjekten. Zufallstests helfen dabei, Fehlerzustände zu vermeiden, die beim manuellen Testen aufgrund von falschem Vertrauen in bestimmte Codebereiche oder Funktionalitäten übersehen werden. Zufallstests sind jedoch auch mit einigen Herausforderungen und Einschränkungen verbunden, darunter die Vernachlässigung der Datensemantik, das mögliche Übersehen von Fehlerzuständen im Zusammenhang mit der Datenbedeutung, das Übersehen bestimmter Fehlerzustände, die Generierung redundanter Tests, die Abhängigkeit von einem automatisierten Testorakel und zufällige Ausgaben, die zu inkonsistenten Testergebnissen führen. Die Abwägung der Vorteile und Grenzen von Zufallstests ist in jedem Testkontext von entscheidender Bedeutung.

Traditionell gelten Zufallstests als weniger effektiv als andere Testverfahren. In den letzten Jahren ist diese Hypothese in vielen empirischen Studien untersucht worden. Dabei hat sich gezeigt, dass Zufallstests unter den oben genannten Umständen effektiver und effizienter sein können als andere datenbasierte Testverfahren (Arcuri u. a., 2012), (Wu u. a., 2020). Zufallstests werden auch beim Fuzzing und Chaos Engineering eingesetzt.

3.2 Verhaltensbasierte Testverfahren

Verhaltensbasierte Testverfahren leiten Testfälle aus Spezifikationen des zustandsabhängigen dynamischen Verhaltens des Testelements ab. In diesem Lehrplan werden drei verhaltensbasierte Testverfahren behandelt:

- CRUD-Tests

- Zustandsübergangstests
- Szenariobasierte Tests

Der CRUD-Test und der szenariobasierte Test erweitern das Spektrum der aus (ISTQB-CTFL, v4.0.2 DE, Abschnitt 4.2) bekannten Black-Box-Testverfahren. Dieser Lehrplan ergänzt die Zustandsübergangstests um zusätzliche Überdeckungskriterien.

3.2.1 CRUD-Test

Der CRUD-Test überprüft den Lebenszyklus der von Testelementen verarbeiteten Datenentitäten. CRUD steht für *create* (erstellen), *read* (lesen/auslesen), *update* (aktualisieren), und *delete* (löschen). Dies sind die vier grundlegenden Operationen, die Funktionen mit Entitäten durchführen können.

Die CRUD-Matrix gibt einen Überblick über den Lebenszyklus der Datenentitäten. Ihre Spalten stehen für die Entitäten, ihre Zeilen für die Funktionen. Angenommen, eine Funktion führt eine oder mehrere bestimmte Create-, Read-, Update- oder Delete-Operationen für eine bestimmte Entität aus, dann wird dies in der Matrix mit den Anfangsbuchstaben dieser Operationen dargestellt: C, R, U oder D. Um eine CRUD-Matrix zu erstellen, bestimmen die Testanalysten für jede Funktion, welche der vier Operationen sie auf welche Entitäten ausführt. Besondere Aufmerksamkeit sollte der Read-Operation gewidmet werden, die oft implizit mit den C-U-D-Operationen verbunden ist.

Der CRUD-Test besteht aus zwei Teilen (Koomen u. a., 2008):

- Der **Vollständigkeitstest** ist ein statischer Test, der überprüft, ob alle möglichen Operationen (d. h. C, R, U und D) bei jeder Entität auftreten (d. h., ob der gesamte Lebenszyklus für jede Entität implementiert wurde). Das Fehlen einer Operation ist eine Anomalie, die untersucht werden muss.
- Der **Konsistenztest** ist ein dynamischer Test, der darauf abzielt, die verschiedenen Funktionen zu integrieren und zu prüfen, ob die Entität konsistent verwendet wird. Es wird überprüft, ob die Funktionen bei der Handhabung einer Entität korrekt zusammenwirken. Die Testfälle sollten alle Operationen in der CRUD-Matrix überdecken. Darüber hinaus sollten auch Negativtests enthalten sein (z. B. das Lesen einer noch nicht erstellten Entität). Testfälle werden pro Entität entworfen, indem Funktionen kombiniert werden, um ihren gesamten Lebenszyklus zu überdecken.

Die **CRUD-Überdeckung** wird gemessen, indem die Anzahl der von der Testsuite ausgeführten Operationen in der CRUD-Matrix durch die Gesamtzahl der Operationen in der CRUD-Matrix dividiert wird. Eine strengere CRUD-Überdeckung kann spezifische Kombinationen von Operationen als Überdeckungselemente berücksichtigen (z. B. sollten nach jedem U alle möglichen Rs abgedeckt werden).

Der CRUD-Test wird vor allem beim Systemtest eingesetzt und konzentriert sich auf Fehlerzustände im Verhalten des Testelements bei der Behandlung von Entitäten, wie z. B. Verletzungen der Integrität von Daten, Fehler bei der Zugriffskontrolle oder Dateninkonsistenzen.

3.2.2 Zustandsübergangstest

Viele komplexe Systeme sind zustandsbehaftet (d. h., die Reaktion des Systems auf ein Ereignis hängt vom aktuellen Zustand des Systems ab). Beispiele für zustandsabhängige Systeme sind eingebettete Systeme, dialogbasierte Systeme, Kontrollsysteme oder Systeme, die sich mit Entitäten und deren Lebenszyklen befassen.

Ein Zustand vereinfacht komplexe interne Details, wodurch es für die Beteiligten einfacher wird, das erwartete Verhalten zu verstehen. Bei der Testanalyse müssen die Testanalysten sicherstellen, dass das

zustandsbasmodell das erwartete Verhalten des Testelements repräsentiert, und zwar in dem für das Testen erforderlichen Detaillierungsgrad. Wenn die Testbasis bereits ein solches Modell enthält, müssen die Testanalysten prüfen, ob das Modell die Testbedingungen enthält und es gegebenenfalls anpassen oder ein neues Modell entwerfen.

In zustandsmodellen stellen die Knoten Zustände und die Kanten Zustandsübergänge dar. Ein Zustandsübergang wird durch Ereignisse ausgelöst und kann [Wächterbedingung]en{.index} (guard conditions) und Aktionen enthalten (siehe ISTQB-CTFL, v4.0.2 DE, Abschnitt 4.2.4). Es sind mehrere Varianten dieser Modelle in Gebrauch, wie z. B. erweiterte endliche Zustandsautomaten (extended finite state machines - efsm) (Bochmann u. a., 1994), Harel-Zustandsdiagramme (Harel, 1987) oder UML-Zustandsautomaten (*OMG® UML*, 2017).

Zusätzlich zu den in (ISTQB-CTFL, v4.0.2 DE) diskutierten Kriterien für die Überdeckung von Zustandsübergängen werden im Folgenden zwei weitere Kriterien erläutert, für die empirisch eine hohe Effektivität bei der Fehlererkennung nachgewiesen wurde:

- **N-Switch-Überdeckung** gilt für gültige Sequenzen von N+1 aufeinanderfolgenden Zustandsübergängen, auch N-Switches genannt (Chow, 1978). 0-Switch-Überdeckung entspricht der *Überdeckung aller gültigen Zustandsübergänge* (siehe ISTQB-CTFL, v4.0.2 DE, Abschnitt 4.2.4). 1-Switch-Überdeckung ist ein Paar von eingehenden und ausgehenden Zustandsübergängen mit genau einem Zustand dazwischen. Die Erweiterung von N-Switches an ihrem Ende durch gültige nachfolgende Zustandsübergänge ergibt N+1-Switches. In der Praxis werden häufig 0- und 1-Switch-Überdeckungen verwendet. Eine 100%ige 2-Switch- oder höhere Überdeckung ist nur bei einem hohen Risiko einer Fehlerwirkung durch unerwartete Ereignisfolgen notwendig, da die Anzahl der N-Switches mit jedem höheren N exponentiell wachsen kann.
- **Rundreiseüberdeckung** (Ammann u. a., 2008) gilt für Pfade in einem zustandsmodell, die Schleifen bilden. Eine Rundreise ist eine Schleife, bei der Start- und Endzustand gleich sind und kein anderer Zustand in dieser Schleife zweimal vorkommt. Die Überdeckungselemente sind die Rundreisen. (Antoniol u. a., 2002) befand dieses Kriterium als sehr effektiv bei der Erkennung von Fehlerzuständen.

Zustandsübergangstests eignen sich gut für die Automatisierung mittels Werkzeugen für den modellbasierten Test. Wie die meisten Black-Box-Testverfahren trägt auch der Zustandsübergangstest zur Fehlervorbeugung bei, indem er Fehlerzustände in der Spezifikation während der Modellierung aufdeckt. Der Zustandsübergangstest kann auf jeder Teststufe angewendet werden.

Weitere Überdeckungskriterien werden in (Rechtberger u. a., 2022) diskutiert. Ein neueres zustandsmodell ist das Aktions-Zustands-Modell, das in (Forgács u. a., 2024) diskutiert wird.

3.2.3 Szenariobasierter Test

Beim szenariobasierten Test wird das Verhalten des Testelements in realistischen Szenarien evaluiert. Verfahren wie Benutzerstudie, User-Storys, Personas und User-Journey-Kartes (siehe *Abschnitt 11*) können helfen, wertvolle Szenarien zu identifizieren.

Beim szenariobasierten Test erstellen die Testanalysten ein Szenariomodell auf der Grundlage von Handlungssequenzen, die Abläufe durch das Testelement darstellen (vgl. *ISO/IEC/IEEE 29119-4*, 2021). In diesem Lehrplan werden die folgenden zwei Modelle behandelt: Aktivitätsdiagramm und Anwendungsfall. Andere Modelle sind Flussdiagramme, Geschäftsprozessmodelle (*OMG® BPMN*, 2013), Sequenzdiagramme oder Kollaborationsdiagramme (*OMG® UML*, 2017). Diese Modelle werden in diesem Lehrplan nicht behandelt. Weitere Einzelheiten finden Sie in (ISTQB-AcT, v1.0 DE).

Ein **Aktivitätsdiagramm** ist eine grafische Darstellung eines Arbeitsablaufs innerhalb eines Systems. Aktivitätsdiagramme sind besonders nützlich für die Modellierung von Geschäftsprozessen, können aber auch Kontrollflüsse modellieren. Aktivitätsdiagramme erweitern die Notation von Flussdiagrammen und ermöglichen die Modellierung von Nebenläufigkeit. Die Hauptelemente von Aktivitätsdiagrammen sind Start- und Endknoten, Aktionen, Kontroll- und Datenflüsse, Entscheidungsknoten, Merge-Knoten, Fork-Knoten, Join-Knoten und Swimlanes.

Ein **Anwendungsfall** ist eine textuelle oder grafische Beschreibung von Aktionen, die Interaktionen zwischen einem Benutzer und einem System oder zwischen Systemen darstellen. In einem Anwendungsfall werden drei Arten von Szenarien unterschieden:

- **Hauptszenario** (auch "happy path") – eine typische, erwartete Abfolge von Aktionen, die aus Sicht des Benutzers zum Erreichen eines bestimmten Ziels führt. Ein Anwendungsfall kann nur ein Hauptszenario besitzen.
- **Erweiterung** (oder Alternativszenario) – eine vom Hauptszenario abweichende Abfolge von Aktionen, die letztendlich zum Erreichen des Ziels des Hauptszenarios führt.
- **Ausnahme** – eine Abfolge von Aktionen, die das Erreichen des Ziels des Hauptszenarios aufgrund einer unerwarteten Aktion (z. B. abnormale Nutzung oder ungültige Eingabe) nicht zulässt.

Beim szenariobasierten Test entwerfen die Testanalysten Testfälle zur Überdeckung der Szenarien (die Überdeckungselemente), wobei die Testanalysten häufig eine risikobasierte Priorisierung vornehmen. Wenn das Szenariomodell keine Schleifen enthält, kann jedes Szenario mit einem separaten Testfall getestet werden (d. h., alle möglichen Szenarien im Modell können durch eine Testsuite ausgeführt werden). Die Anzahl der möglichen Szenarien (Pfade) kann unendlich sein, wenn Schleifen vorhanden sind. In diesem Fall kann die einfache Schleifenüberdeckung auf das Szenariomodell angewendet werden. Die einfache Schleifenüberdeckung erfordert das Testen, wenn jede Schleife nullmal (d. h. übersprungen), mit genau einer Iteration, mit mehr als einer Iteration (d. h. einer typischen Anzahl von Iterationen) und mit der maximalen Anzahl von Iterationen (wenn möglich) ausgeführt wird.

Die **szenariobasierte Überdeckung** wird ermittelt, indem die Anzahl der ausgeführten Szenarien durch die Anzahl aller identifizierten Szenarien dividiert wird. Szenarien können durch Anwendung verschiedener Überdeckungskriterien auf das Szenariomodell identifiziert werden (siehe Koomen u. a., 2008). Überdeckungskriterien können erfordern, dass jedes Szenario mehr als einmal getestet wird. So kann ein Szenario beispielsweise eine zusätzliche Äquivalenzklassen- oder Grenzwertüberdeckung der im Szenario vorkommenden Variablen erfordern. In solchen Fällen muss ein Szenario möglicherweise durch mehr als einen Testfall getestet werden.

Der szenariobasierte Test wird häufig bei Systemtests oder Abnahmetests eingesetzt. Es handelt sich um einen Ende-zu-Ende-Tests, der sich auf die funktionale Eignung eines Systems (siehe *Abschnitt 4.1*) aus Sicht des Benutzers konzentriert. Der szenariobasierte Test kann aber auch bei anderen Teststufen verwendet werden (z. B. Komponentenintegrationstests mit Szenarien auf Basis von Interaktionsprotokollen oder Komponententests von zustandsbehafteten, objektorientierten Klassen mit Szenarien, die verschiedene Methoden aufrufen) und auch bei nicht-funktionalen Tests eingesetzt werden (z. B. können Szenarien die Elemente der Nutzungsprofile darstellen, die bei Zuverlässigkeits-, Flexibilitäts- oder Kompatibilitätstests verwendet werden).

3.3 Regelbasierte Testverfahren

Mit regelbasierten Testverfahren wird die Implementierung des zustandsunabhängigen Verhaltens eines Testelements überprüft, das durch Regeln spezifiziert ist, die unabhängig von dessen Zustand gültig sind (z. B. Geschäftsregeln). In diesem Lehrplan werden zwei regelbasierte Testverfahren besprochen:

- Entscheidungstabellentest
- Metamorpher Test

3.3.1 Entscheidungstabellentest

Der Foundation-Level-Lehrplan (ISTQB-CTFL, v4.0.2 DE) behandelt die Grundlagen des Entscheidungstabellentests. Dieser Lehrplan baut darauf auf und behandelt vertiefende Themen. Die verwendete Terminologie und Notation entspricht dem Standard (*OMG® DMN*, 2024).

Beim Entscheidungstabellentest beginnt der Testanalyst in der Regel mit der Erstellung einer vollständigen Entscheidungstabelle oder mit der Analyse einer bestehenden Tabelle, die aus der Testbasis stammt. Die Anzahl der Regeln einer vollständigen Entscheidungstabelle ist das Produkt aus der Anzahl der Werte ihrer Bedingungen. Diese Zahl wächst exponentiell mit der Anzahl der Bedingungen und ihrer Werte und motiviert zur Minimierung der Entscheidungstabelle. Die Minimierung geht von einer ursprünglichen Entscheidungstabelle aus, die vollständig oder unvollständig sein kann, und leitet eine äquivalente Entscheidungstabelle mit weniger Regeln ab.

Entscheidungstabellen können durch das Zusammenführen von Regeln unter Verwendung von irrelevanten Bedingungen ("don't care"-Operator "-") minimiert werden. Es wird empfohlen, beim Zusammenführen von Regeln nicht erfüllbare Regeln zu ignorieren (d. h. Regeln mit einer Kombination von Bedingungswerten, die niemals eintreten kann). Häufig werden nicht erfüllbare Regeln bereits vor dem Zusammenführen aus der Entscheidungstabelle entfernt. Ein möglicher systematischer Minimierungsalgorithmus sucht nach aktionsäquivalenten Regeln, die sich nur in einer Bedingung unterscheiden und alle deren möglichen Werte abdecken. Diese Regeln werden zusammengeführt, und die Werte der Bedingung, die sich voneinander unterscheiden, werden durch "-" (irrelevant) ersetzt. Das Ergebnis der systematischen Minimierung kann von der Reihenfolge abhängen, in der die Spalten minimiert werden, und führt nicht immer zu einer optimalen Lösung. Testanalysten müssen prüfen, ob eine weitere Minimierung möglich ist.

Testanalysten haben die Aufgabe, die Entscheidungstabelle, eventuell zusammen mit anderen Stakeholdern, anhand der folgenden Kriterien zu reviewen:

- Konsistenz (d. h., wenn zwei verschiedene Regeln auf dieselbe Kombination von Bedingungswerten anwendbar sind, dann sind sie aktionsäquivalent)
- Durchführbarkeit (d. h., es gibt keine nicht erfüllbaren Regeln)
- Vollständigkeit (d. h., es fehlt keine erfüllbare Kombination von Bedingungswerten)
- Korrektheit (d. h. die Regeln modellieren das vorgesehene Verhalten des Systems)

Darüber hinaus ist es ratsam, dass sich die Regeln nicht überschneiden (d. h., für jede Kombination von Bedingungswerten gilt höchstens eine Regel). Zu Überschneidungen von Regeln kann es kommen, wenn die ursprüngliche Entscheidungstabelle bereits minimiert ist oder wenn Regeln nicht korrekt zusammengeführt werden.

Das *Prüfsummenverfahren* verwendet die Anzahl der Regeln in einer minimierten Entscheidungstabelle, um Überschneidungen und Lücken anzuzeigen. Für jede Regel in der minimierten Tabelle wird berechnet, wie viele Regeln der ursprünglichen Entscheidungstabelle sie darstellt. Eine Regel ohne ein "-" in den Bedingungen (d. h. mit Einzelwerten für alle Bedingungen) erhält einen Punkt. Jeder "-"-Wert multipliziert die

Punktzahl der Regel mit der Anzahl der Einzelwerte für die jeweilige Bedingung. Die Summe der Punktzahlen der Regeln ist die Prüfsumme der minimierten Entscheidungstabelle. Wenn die Prüfsumme kleiner ist als die Prüfsumme der ursprünglichen Entscheidungstabelle, ist die minimierte Entscheidungstabelle unvollständig. Wenn die Prüfsumme höher ist, überschneiden sich einige Regeln oder es gibt zusätzliche Regeln (z. B. nicht erfüllbare Kombinationen von Bedingungen). Wenn die Minimierung korrekt durchgeführt wurde, sind die Prüfsummen gleich. Gleiche Prüfsummen allein garantieren jedoch nicht, dass die minimierte Entscheidungstabelle mit der ursprünglichen äquivalent ist.

Die **Entscheidungstabellenüberdeckung** wird gemessen, indem die Anzahl der ausgeführten Spalten durch die Gesamtzahl der erfüllbaren Spalten in der Entscheidungstabelle geteilt wird. Bei der Realisierung von Testfällen, die sich aus einer Entscheidungstabellenregel ergeben, müssen Testanalysten die Bedingungen und Aktionen entsprechend umsetzen. Die Testanalysten müssen entscheiden, wie die "-"-Bedingungswerte für eine bestimmte Regel realisiert werden sollen, da "-" mindestens zwei Werte darstellt. Ist das mit der Entscheidungstabelle verbundene Risiko jedoch hoch, sollten Testanalysten von einer Minimierung absehen und die Entscheidungsüberdeckung anhand der erfüllbaren Spalten in der vollständigen Entscheidungstabelle messen.

3.3.2 Metamorpher Test

Metamorpher Test (MT) ist ein Verfahren zur Generierung von Testfällen, die auf einem bestehenden Ausgangstestfall basieren. Ein oder mehrere Folgetestfälle werden durch Veränderung (Metamorphisierung) des Ausgangstestfalls auf der Grundlage einer metamorphen Relation (MR) erzeugt. Die MR definiert eine Eigenschaft des Testelements und beschreibt, wie sich eine Änderung der Eingaben eines Testfalls auf die erwarteten Ergebnisse des Testfalls auswirkt.

Die Testanalysten fassen den Ausgangstestfall und die Folgetestfälle einer MR zu einem Testablauf mit einer gemeinsamen Ergebnisauswertung zusammen. Erfüllen sie die metamorphe Relation, ist der Test bestanden, andernfalls fehlgeschlagen. Im Falle des Fehlschlagens muss beim anschließenden Debugging festgestellt werden, welcher der beteiligten Testfälle im Einzelnen fehlgeschlagen ist.

Nehmen wir z. B. eine Funktion, die den Durchschnitt einer Reihe von Zahlen ermittelt. Es wird ein Ausgangstestfall mit einer Zahlenreihe und einem erwarteten Durchschnitt erstellt, und der Testfall wird ausgeführt, um zu bestätigen, dass er bestanden ist. Eine MR könnte besagen, dass jede Permutation der Zahlenreihe denselben Durchschnitt ergibt. Mithilfe dieser MR können Testanalysten mehrere Folgetestfälle erstellen, wobei jeder Testfall dieselben Zahlenwerte enthält, jedoch in einer anderen Reihenfolge. Das erwartete Ergebnis bleibt das gleiche.

Für dieselbe Durchschnittsfunktion können Testanalysten auch eine andere MR verwenden, die besagt, dass wenn jede Zahl der Reihe mit derselben Zahl x multipliziert wird, dann das erwartete Ergebnis ebenfalls mit x multipliziert wird. Mit dieser MR können die Testanalysten aus einem Ausgangstestfall eine beliebige Anzahl von Folgetestfällen erstellen, indem sie verschiedene Werte von x wählen. Dies kann besonders nützlich sein, wenn Testentwurf und Testdurchführung automatisiert sind. Die Testanalysten können auch zwei oder mehr MRs kombinieren, um Folgetestfälle zu erstellen (z. B. permutieren und multiplizieren mit 2).

MT kann auch verwendet werden, wenn ein Testorakelproblem vorliegt (siehe *Abschnitt 1.3.4*). In einer solchen Situation sind die erwarteten Ergebnisse des Ausgangstestfalls und der Folgetestfälle nicht verfügbar, so dass ihre Testergebnisse nicht einzeln ausgewertet werden können. Ein Beispiel ist ein KI-basiertes versicherungsmathematisches Programm, das auf der Grundlage einer großen Datenmenge das Sterbealter vorhersagt. Die MR könnte besagen, dass das vorhergesagte Sterbealter sinken sollte, wenn die Anzahl der gerauchten Zigaretten erhöht wird.

Derzeit gibt es keine anerkannten Messungen der Überdeckung für MT, die zweckdienliche Endekriterien liefern. Die einmalige Abdeckung jeder MR ist unzureichend, da nur eine teilweise Verifizierung der erwarteten Ergebnisse möglich ist. Testanalysten können MT mit Zufallstests kombinieren, um viele konkrete Ausgangstestfälle und Folgetestfälle für dieselbe MR zu generieren. Bei der oben erwähnten Durchschnittsberechnung kann z. B. ein Zufallszahlengenerator verwendet werden, um verschiedene Eingaben für den Ausgangstestfall sowie zufällige Permutationen und Multiplikatoren für Folgetestfälle zu erzeugen.

MT kann für die meisten Testelemente verwendet und für funktionale und nicht-funktionale Tests eingesetzt werden (z. B. Lasttests, Erzeugung von Last durch Folgetestfälle unter Verwendung von MRs oder Installierbarkeitstests mit verschiedenen Installationsparametern, die in mehreren unterschiedlichen Sequenzen ausgewählt werden können). Es ist ein bevorzugtes Testverfahren für KI-basierte Systeme (ISTQB-AI, v1.0 DE).

Für weitere Einzelheiten siehe auch die Norm (*ISO/IEC/IEEE 29119-4*, 2021) oder die Übersichtsartikel (Segura; Towey u. a., 2020) und (Segura; Fraser u. a., 2016).

3.4 Erfahrungsbasierter Test

Testanalysten setzen erfahrungsbasierte Testansätze und Testverfahren ein und nutzen ihr Fachwissen und ihre bisherigen Erfahrungen, um das Testen zu leiten. In diesem Kapitel wird die Dokumentation beim sitzungsbasierten Testen und checklistenbasierten Testen (siehe ISTQB-CTFL, v4.0.2 DE, Abschnitte 4.4.2 und 4.4.3) durch die Testanalysten beschrieben. Darüber hinaus wird der Massentest behandelt. Alle erfahrungsbasierten Testverfahren können beim kollaborativen Testen angewendet werden, wie z. B. bei der abnahmetestgetriebenen Entwicklung. Für weitere Einzelheiten siehe (ISTQB-CTFL, v4.0.2 DE, Abschnitt 4.5).

3.4.1 Test-Chartas für die Unterstützung sitzungsbasierter Tests

Beim explorativen Test stellt eine Test-Charta eine Leitlinie einer Testsitzung dar, in der der Testumfang und die Testziele sowie Informationen wie Einschränkungen, Zeitpläne und Risiken beschrieben werden. Die Test-Charta dient als Fahrplan für das Testen und gibt den Testsitzungen Struktur. Test-Chartas helfen den Testanalysten, sich auf bestimmte Bereiche oder Funktionen zu konzentrieren, die getestet werden sollen, und lassen ihnen gleichzeitig die Flexibilität, das System bei Bedarf zu erkunden. Eine Test-Charta spezifiziert jedoch nicht, welche Testsuiten in jeder Testsitzung ausgeführt werden.

Bei der Erstellung einer Test-Charta für den sitzungsbasierten Test müssen Testanalysten bestimmte Faktoren berücksichtigen, die den Entwurf der Test-Charta beeinflussen, insbesondere:

- Kunden- und Anforderungsfaktoren (z. B. von Kunden erhobene Anforderungen, geschäftliche Anwendungsfälle für das System, Qualitätsanforderungen) und User-Journey-Karten (d. h. die Interaktion des Benutzers mit dem System im Laufe der Zeit)
- Produktfaktoren (z. B. Funktionsabläufe, Hauptziele des Produkts, Produktmerkmale, Softwareentwurf und Schnittstellen)
- Projektmanagementfaktoren (z. B. zeitliche Beschränkungen, Projektziel, geschätzter Aufwand und Geschäftswert)

Eine Test-Charta besteht aus einem Leitbild, das das Testziel beschreibt, sowie verschiedenen Zusatzinformationen.

Ein beliebtes leichtgewichtiges Format für ein Leitbild ist "Erforsche [Ziel] mit [Ressourcen], um [Informationen] zu entdecken" (Hendrickson, 2013), wobei [Ziel] beschreibt, was erforscht werden soll (z. B. Bereich, Merkmal, Risiko, Komponente und Anforderung), [Ressourcen] beschreiben, was die Testanalysten verwenden werden (z. B. Testdaten, Konfigurationen, Werkzeuge, Einschränkungen, Heuristiken und Abhängigkeiten), und [Informationen] erklären, welche Art von Informationen die Testanalysten zu finden versuchen (z. B. Bewertungen von Qualitätsmerkmalen, erwartete Fehlerzustände und Verletzungen von Standards).

Test-Chartas können die folgenden zusätzlichen Informationen enthalten, sind aber nicht darauf beschränkt (Ghazi u. a., 2017):

- Organisatorische Informationen (z. B. Dauer der Testsitzung, Datum und Uhrzeit des Beginns und Name des Testers)
- Testziele (z. B. Motivation des Tests und Leitbild der Test-Charta)
- Testumfang (z. B. spezifische Interessenbereiche innerhalb des zu testenden Systems, Teststufe, zu verwendende Testverfahren, Testideen, Endekriterien, Prioritäten, die die Test-Charta überdecken soll, und eine Beschreibung dessen, was nicht getestet wird)
- Eingangskriterien (d. h. Vorbedingungen, die erfüllt sein müssen, um die Testsitzung beginnen zu können)
- Produktbezogene Informationen (z. B. Definition, Daten und Arbeitsabläufe zwischen den Komponenten und Systemarchitektur)
- Einschränkungen (d. h., was das Produkt auf keinen Fall tun darf)
- Beschreibung der Testumgebung
- Vorhandene Datenquellen, Produktinformationen und Werkzeuge, die das Testen erleichtern würden
- Historische Informationen (z. B. früher gefundene Fehlerzustände wie Kompatibilitäts- und Interoperabilitätsfehler, aktuelle offene Fragen über bestehende Anomalien, und testbezogene Fehlermuster der Vergangenheit)
- Beschränkungen und Risiken (z. B. regulatorische Vorschriften, Regeln, verwendete Normen und Standards)

Während einer Testsitzung zeichnen Testanalysten ein Testprotokoll auf, das neben den Testergebnissen auch Fragen, Beobachtungen oder Ideen für zukünftige Tests enthält. Diese Daten werden in einem Testsitzungsblatt (session sheet) dokumentiert.

Umfang und Detaillierungsgrad der in bestimmten Test-Chartas enthaltenen Informationen können variieren und beeinflussen den Grad der Flexibilität der Testanalysten. Wenn beispielsweise nur die allgemeinen Testziele definiert werden, bietet dies reichlich Spielraum für explorative Erkundungen, während das Hinzufügen von Informationen über die zu verwendenden Testverfahren die Testanalysten einschränken kann. Andererseits kann das Hinzufügen von Informationen (z. B. was das System definitiv nicht kann) dazu beitragen, die Wahrscheinlichkeit falsch-positiver Ergebnisse zu verringern.

3.4.2 Checklisten zur Unterstützung erfahrungsbasierter Testverfahren

Der checklistenbasierte Test ist aufgrund seiner Anpassbarkeit, Einfachheit und Effektivität bei der Sicherung der Softwarequalität ein weitverbreitetes Testverfahren. Durch die Verwendung von Checklisten stellen die Testanalysten sicher, dass alle bekannten wesentlichen Aspekte eines Testelements berücksichtigt

werden, wodurch verhindert wird, dass kritische Bereiche übersehen werden. Außerdem sorgen Checklisten für Konsistenz in den Testzyklen und zwischen den verschiedenen Testanalysten. Bei der Verwendung einer Checkliste konzentrieren sich die Testanalysten auf wichtige Aspekte. Checklisten sind eine Form der Aufzeichnung vergangener Erfahrungen der Testanalysten mit Fehlerwirkungen und Fehlerzuständen. Sie dienen als Gedächtnisstütze oder Inspirationsquelle, wenn Testanalysten die Ideen ausgehen (z. B. beim explorativen Test). Testanalysten sparen durch die Wiederverwendung einer Standard-Checkliste oder einer von einem Kollegen erstellten Checkliste Zeit, aber nur, wenn diese Checklisten für das Testelement relevant sind. Checklisten tragen dazu bei, den Arbeitsaufwand für die Dokumentation von Testfällen zu verringern, was ein großer Vorteil sein kann, wenn sich Anforderungen und Software häufig ändern.

Die Erstellung von Checklisten liegt oft in der Verantwortung der Testanalysten. Checklisten unterstützen den erfahrungsbasierten Test, da sie helfen, das Testen zu organisieren, zu strukturieren und zu leiten. Die Erstellung einer wiederverwendbaren, wartbaren, klaren und effizienten Checkliste erfordert Aufwand.

Die folgenden Schritte können als Leitfaden für die Erstellung einer geeigneten Checkliste für den erfahrungsbasierten Test dienen:

Zunächst bestimmen die Testanalysten den Umfang, die Ziele und das Format der Checkliste, da diese Attribute die Testtiefe und den geforderten Detaillierungsgrad beeinflussen. Eine Anleitungs-Checkliste enthält die wichtigsten Elemente, die für einen bestimmten Prozess zu berücksichtigen sind (z. B. enthalten die Elemente der Checkliste konkrete ungültige Eingaben für ein zu prüfendes Eingabefeld). Eine Bestätigungs-Checkliste dient als Gedankenstütze und liefert erfahrungsbasierte Testideen, um eine Anwendung weiter zu erforschen (z. B. könnte ein Checklistelement die Überprüfung der Relevanz von Suchergebnissen erfordern).

Als Nächstes sammeln die Testanalysten die Informationen, die für die Definition der Checklistenelemente erforderlich sind. Dies kann das Sammeln von Erkenntnissen erfahrener Fachexperten, das Durchsuchen von Fehlerbibliotheken und Fehlertaxonomien (Beizer, 1990), (Kaner; Falk u. a., 1999), das Review relevanter Dokumentation und die Analyse von Risiken, Testfällen und möglichen Szenarien beinhalten. Die Elemente der Checkliste sollten klar, spezifisch, eindeutig, konsistent, relevant, wartbar, umsetzbar und messbar sein. Sie sollten als Fragen formuliert werden, die mit "ja", "nein" oder "nicht anwendbar" beantwortet werden können. Sie erfordern die Zuweisung von Prioritäten auf der Grundlage ihrer Wichtigkeit, ihrer potenziellen Auswirkungen und der Risikostufe. Checklisten sind keine umfassenden Anleitungen, sondern vielmehr schnelle Hilfen und einfache Werkzeuge für Fachexperten.

Schließlich strukturieren und organisieren die Testanalysen die Checkliste, indem die Checklistenelemente in logische Gruppen eingeteilt werden, die auf Funktionsbereichen, Benutzerrollen, Teststufen oder anderen relevanten Kriterien basieren. Die Erstellung separater Kategorien kann besonders bei langen Checklisten sinnvoll sein.

Wo immer möglich, sollten Vorlagen und Standards verwendet werden. Durch die Verwendung vorhandener Vorlagen oder vordefinierter Checklisten, die sich an Industriestandards und bewährten Praktiken orientieren, können Testanalysten Aufwand einsparen.

Eine Checkliste ist nie abgeschlossen. Testanalysten überprüfen und verfeinern die Checklisten fortwährend und passen sie an neue Befunde, veränderte Prioritäten, Rückmeldungen anderer Tester oder Erfahrungen aus früheren Testzyklen an. Durch die Weitergabe der Checkliste an andere Tester fördern Testanalysten die Konsistenz und Zusammenarbeit und helfen dabei, die Testelemente und die kritischen Bereiche, auf die sich die Tester konzentrieren müssen, besser zu verstehen.

3.4.3 Massentest

Beim Massentest werden Tests auf eine Gruppe von internen oder externen Testern mit unterschiedlichem Hintergrund und an verschiedenen Orten verteilt. Es kann eine kostengünstige Methode zur Validierung der Gebrauchstauglichkeit sein und deckt sowohl funktionale als auch nicht-funktionale Qualitätsmerkmale ab (Alyahya, 2020), (Leicht u. a., 2017).

Zu den Vorteilen des Massentests gehören unter anderem:

- **Unterschiedliche Testumgebungen**: Tester können sich an verschiedenen geografischen Standorten befinden und verschiedene Umgebungskonfigurationen mit einer Vielzahl von Geräten, Browsern und Netzwerkbedingungen verwenden.
- **Mehr Flexibilität**: Leicht skalierbar, um viele Tests in kurzer Zeit durchzuführen.
- **Kostengünstig**: Massentests sind in der Regel kostengünstiger als ein großes und vielfältiges internes Testteam oder die Inanspruchnahme externer Testdienste.
- **Schnelles Feedback**: Tester können schnelles Feedback geben und so dazu beitragen, Fehlerwirkungen frühzeitig zu erkennen und zu beheben.
- **Echte Benutzerperspektive**: Tester können tatsächliche Benutzer der Anwendung sein und einen besseren Einblick in das Benutzererlebnis und die Gebrauchstauglichkeit der Anwendung geben. Dies kann besonders bei Benutzerabnahmetests von Vorteil sein.
- **Variabilität**: Da die Tests jedes Mal von einer Vielzahl von Testern durchgeführt werden, sind sie nicht leicht wiederholbar. Dies kann zwar eine Einschränkung darstellen, führt aber auch zu einer höheren Überdeckung, was die Wahrscheinlichkeit erhöht, Fehlerzustände zu finden.

Einige der Einschränkungen des Massentests sind die folgenden:

- **Unzuverlässige Qualität des Testens**: Die Qualität der Tests kann je nach den Fähigkeiten der einzelnen Tester erheblich schwanken, obwohl dies gegebenenfalls nicht relevant ist, z. B. wenn das Ziel ein Feedback zum Benutzererlebnis ist.
- **Kommunikationsprobleme**: Die Koordinierung mit vielen Testern von verschiedenen Standorten mit unterschiedlichen Zeitzonen, kulturellen Unterschieden und Sprachbarrieren kann herausfordernd sein.
- **Sicherheit (Security)**: Die gemeinsame Nutzung von Software mit externen Testern birgt Risiken für die Sicherheit (Security) und die Vertraulichkeit von Daten. Durch geeignete Maßnahmen können diese Risiken gemindert werden, so dass ein verantwortungsvoller Einsatz von Massentests möglich ist, ohne dass sensible Details preisgegeben oder Plagiate gefördert werden.
- **Dokumentation und Berichterstattung**: Die Sicherstellung einer umfassenden Testdokumentation und die Verwaltung einer großen Anzahl von Befunden, einschließlich Duplikaten und falschpositiven Ergebnissen, können bei einer großen und vielfältigen Gruppe von Testern eine Herausforderung darstellen.

Der Massentest kann die Überdeckung verschiedener Testumgebungen erhöhen, er ist aber kein Ersatz für die Anwendung von Testverfahren durch die Testanalysten.

3.5 Anwendung der am besten geeigneten Testverfahren

Das Testen sollte unter den jeweils gegebenen Umständen so effektiv und effizient wie möglich sein. Zu diesem Zweck unterstützen die Testanalysten die Testmanager bei der Auswahl der am besten geeigneten Testverfahren. Darüber hinaus können die Testanalysten die Testaktivitäten durch Automatisierung unterstützen. Dazu gehört die Automatisierung des Testentwurfs, die in diesem Abschnitt beschrieben wird, und die Unterstützung der Automatisierung der Testdurchführung, die in *Abschnitt 1.3.6* erläutert wird.

3.5.1 Auswahl von Testverfahren zur Minderung von Produktrisiken

Die Auswahl der am besten geeigneten Testverfahren ist für eine effektive und effiziente Risikominderung von entscheidender Bedeutung und wird u.a. von den folgenden Faktoren beeinflusst:

Testziele (siehe ISTQB-CTFL, v4.0.2 DE, Abschnitt 1.1.1) geben an, welche Aspekte des Testobjekts zu bewerten sind. Testziele unterstützen die Auswahl der Testverfahren und hängen von der Systemart ab (z. B. wertebereichsbasierte Tests für numerische Berechnungen im Ingenieurwesen gegenüber Entscheidungstabellentests im Kreditrisikomanagement).

Produktrisiken sind mit potenziellen Fehlerzuständen verbunden. Diese Fehlerzustände können am besten mit bestimmten Testverfahren aufgedeckt werden, da sich die meisten Testverfahren auf die Erkennung bestimmter Fehlerarten konzentrieren (siehe *Abschnitt 3.1*, *Abschnitt 3.2*, *Abschnitt 3.3* und *Abschnitt 3.4* in diesem Lehrplan), z. B.:

- Mit datenbasierten Testverfahren können Fehlerzustände bei der Datenverarbeitung, der Benutzungsschnittstelle, der Berechnung und der Kombination von Parametern aufgedeckt werden.
- Mit verhaltensbasierten Testverfahren können Fehlerzustände in den Benutzeranforderungen aufgedeckt werden, z. B. fehlende Funktionen, Kommunikationsfehler und Verarbeitungsfehler.
- Mit regelbasierten Testverfahren lassen sich Fehlerzustände in Logik- und Kontrollflüssen aufdecken.

Die Risikoanalyse trägt auch dazu bei, den geeigneten Testansatz zu bestimmen, z. B.:

- Überdeckungsbasierte Endekriterien: Je höher die Risikostufe, desto strenger kann die Anforderung an die Überdeckung sein (z. B. alle Kombinationen in der kombinatorischen Überdeckung anstelle der paarweisen Überdeckung). Die Testanalysten müssen jedoch immer die Abwägung zwischen der Stärke der Überdeckung und dem resultierenden Testaufwand berücksichtigen.
- Erfahrungsbasiertes Testen kann verwendet werden, wenn die Definition der Überdeckung schwierig ist, wenn die Risikostufe niedrig ist oder wenn der Projektplan zeitlich knapp bemessen ist.

Testbasis: Wenn die Spezifikation des Testobjekts anhand von Modellen erfolgt, können Testanalysten Testverfahren anwenden, die auf diesen Modellen basieren. Wenn es unmöglich oder schwierig ist, ein Testorakel aus der Testbasis abzuleiten, können Testverfahren wie der metamorphe Test oder erfahrungsbasierte Testverfahren angewendet werden.

Die Kenntnis von wiederkehrenden Fehlerarten kann die Auswahl von Testverfahren dahingehend beeinflussen, dass solche Testverfahren eingesetzt werden, die sich auf die Entdeckung entsprechender Fehlerarten konzentrieren (z. B. der checklistenbasierte Test). Wenn die Erwartung ist, ähnliche Fehlerzustände wie in früheren Iterationen oder Projekten zu entdecken, kann es sinnvoll sein, die gleichen erfolgreichen Testverfahren zu verwenden.

Wissen und Erfahrung des Testers: Wenn der Tester mit einem bestimmten Testverfahren nicht vertraut ist, ist es nicht empfehlenswert, dieses Testverfahren bei kritischen Tests einzusetzen. Auch Fachkenntnisse können sich auf die Auswahl der Testverfahren auswirken. Geringe oder fehlende Fachkenntnisse deuten beispielsweise eher darauf hin, dass Testverfahren wie der explorative Test unwirksam sein können.

Eingesetzter Softwareentwicklungslebenszyklus: Ein sequenzielles Entwicklungsmodell eignet sich für den Einsatz formalerer Techniken. Im Gegensatz dazu eignet sich ein iteratives Entwicklungsmodell eher für den Einsatz von leichtgewichtigen Testverfahren (z. B. erfahrungsbasierten Testverfahren) oder wenn der Testentwurf automatisiert werden kann.

Kunden- und Vertragsanforderungen: In Verträgen kann die Durchführung spezifischer Tests (z. B. in Bezug auf bestimmte Teststufen oder Testarten) ausdrücklich vorgeschrieben werden, was sich auf die Auswahl der Testverfahren auswirkt (z. B. Akzeptanzkriterien in Form einer Menge von durch den Kunden bereitgestellten Szenarien legen die Verwendung des szenariobasierten Tests nahe).

Regulatorische Anforderungen: Wenn ein Projekt einer Norm folgt, kann diese den Einsatz bestimmter Testverfahren vorschreiben. Die Norm (*ISO 26262*, 2018) schreibt beispielsweise die Verwendung von Testverfahren wie Äquivalenzklassenbildung, Grenzwertanalyse oder intuitive Testfallermittlung vor, je nachdem, welcher ASIL (Automotive Safety Integrity Level) einem Testobjekt zugeordnet ist.

Projektbeschränkungen, wie Zeit und Budget, können den Einsatz zeitaufwändiger Verfahren oder solcher, die teure Ressourcen erfordern, beeinflussen.

Testverfahren werden häufig kombiniert, um die Effizienz und Effektivität der Erkennung von Fehlerzuständen zu erhöhen. Zum Beispiel:

- Grenzwertanalyse kann für Wächterbedingungen (guard conditions) beim Zustandsübergangstest eingesetzt werden.
- Wertebereichstests können bei szenariobasierten Tests eingesetzt werden, um den Wert einer Bedingung aus einer Entscheidungstabelle oder einer Variablen zu bestimmen, die in einem SUT auftritt.
- Szenariobasierte Tests können mit der Entscheidungsüberdeckung, einem White-Box-Testverfahren, das in (ISTQB-TTA, v4.0 DE) behandelt wird, kombiniert werden, um Entscheidungen im Geschäftsprozess lückenlos zu überdecken. Ein Beispiel hierfür ist der Geschäftsprozesstest (Process cycle test) in (Koomen u. a., 2008).
- Szenariobasierte Tests können mit der Rundreiseüberdeckung kombiniert werden, um die spezifischen Risiken von zyklischen Geschäftsprozessaktivitäten zu berücksichtigen.

3.5.2 Vorteile und Risiken der Automatisierung des Testentwurfs

Testanalysten können Werkzeuge zur Anwendung von Testverfahren verwenden, insbesondere von Black-Box-Testverfahren. Bei der Automatisierung des Testentwurfs erstellen Testanalysten ein Testmodell und generierem automatisiert Testmittel aus diesem Modell. So entwerfen Testanalysten beispielsweise ein Zustandsübergangsmodell und lassen durch ein Werkzeug für den modellbasierten Test die Testfälle für die Rundreiseüberdeckung generieren.

Die Automatisierung des Testentwurfs verbessert häufig die Effizienz und Effektivität des Testens insgesamt. Die Vorteile umfassen:

- **Fehlervorbeugung**: Die Testmodellierung ist eine effektive Möglichkeit, die Qualität der Testbasis zu bewerten (siehe *Abschnitt 5.2.1*).
- **Erweiterte Fähigkeiten**: Die Automatisierung ermöglicht die Anwendung komplexerer Testverfahren und Überdeckungskriterien wie den kombinatorischen Test, Zufallstests oder N-Switch-Überdeckung, wodurch das Risiko von ungetestetem Code verringert wird.
- **Verbesserte Verständlichkeit**: In einem Werkzeug festgelegte Testauswahlkriterien beziehen sich klarer und sichtbarer auf die Testbedingungen und begründen die generierte Überdeckung verständlicher.
- **Weniger repetitive Arbeit**: Testmittel können aus dem Testmodell generiert werden, wodurch manuelle, sich wiederholende Arbeiten wie das Spezifizieren von Tests reduziert werden.
- **Weniger Wartungsaufwand**: Das Testmodell ist die einzige Quelle der Wahrheit, die für die Ableitung der Testmittel verwendet wird. Infolgedessen muss nur das Testmodell gewartet werden.
- **Weniger Fehlerzustände bei Testmitteln**: Manuelle Arbeit ist fehleranfällig. Mit Werkzeugen erzeugte Testmittel haben eine höhere Qualität und Konsistenz.
- **Verbesserte Zusammenarbeit im Team**: Gutachter können das Testmodell reviewen, um Fehlerzustände im Testmodell zu finden oder um die Testbedingungen besser zu verstehen.
- **Verbesserte Verfolgbarkeit**: Es ist einfacher, die Elemente eines Testmodells mit den Testbedingungen zu verknüpfen als die Testfälle selbst. Wenn das Werkzeug dies unterstützt, erben die generierten Testfälle diese Verknüpfungen und verbessern so die allgemeine Verfolgbarkeit beim Testen.
- **Verschiedene Ausgabeformate**: Testmittel können in verschiedenen Ausgabeformaten generiert werden, je nach Anforderung für andere Werkzeuge und Folgeaktivitäten.

Testanalysten müssen auch die Risiken der Automatisierung des Testentwurfs berücksichtigen. Dazu gehören das Übersehen von Testbedingungen, die nicht im Modell abgebildet sind, das Unterschätzen des Wartungsaufwands für das Testmodell, die Schwierigkeit für die Beteiligten, das Modell zu verstehen, und die allgemeinen Risiken der Testautomatisierung (siehe ISTQB-CTFL, v4.0.2 DE, Abschnitt 6.2).

4 Testen von Qualitätsmerkmalen – 60 Minuten

Schlüsselbegriffe

Anpassbarkeit, Benutzererlebnis, Flexibilität, funktionale Angemessenheit, funktionale Eignung, funktionale Korrektheit, funktionale Vollständigkeit, funktionaler Test, Gebrauchstauglichkeit, Installierbarkeit, Interaktionsfähigkeit, Interoperabilität, Kompatibilität

Lernziele für Kapitel 4:

Die Lernenden können ...

4.1 Funktionaler Test

TA-4.1.1 (K2) ... zwischen dem Testen der funktionalen Korrektheit, der funktionalen Angemessenheit und der funktionalen Vollständigkeit unterscheiden

4.2 Gebrauchstauglichkeitstest

TA-4.2.1 (K2) ... erläutern, wie Testanalysten zum Gebrauchstauglichkeitstest beitragen

4.3 Flexibilitätstest

TA-4.3.1 (K2) ... erläutern, wie Testanalysten zum Anpassbarkeitstest und Installierbarkeitstest beitragen

4.4 Kompatibilitätstest

TA-4.4.1 (K2) ... erläutern, wie Testanalysten zum Interoperabilitätstest beitragen

Einführung in das Testen von Qualitätsmerkmalen

Dieser Lehrplan verwendet das Softwarequalitätsmodell der Norm ISO 25010 (*ISO/IEC 25010*, 2023) als Leitfaden und erörtert die Qualitätsmerkmale, die für Testanalysten im Mittelpunkt stehen. Die verwendete Terminologie zur Gebrauchstauglichkeit weicht jedoch von dieser Norm ab, um mit dem Lehrplan für Gebrauchstauglichkeitstests (ISTQB-UT, v1.0 DE) übereinzustimmen.

Anhang F gibt einen Überblick über das Qualitätsmodell der aktuellen Norm ISO 25010, die Änderungen gegenüber der Vorgängerversion und die zugehörigen ISTQB®-Lehrpläne, die sich auf das Testen bestimmter Qualitätsmerkmale konzentrieren.

4.1 Funktionaler Test

Der funktionale Test ist eine der Kernaufgaben der Testanalysten. Während (ISTQB-CTFL, v4.0.2 DE) die Testart "funktionaler Test" kurz zusammenfasst, wird in diesem Lehrplan ausführlicher auf die Untermerkmale der funktionalen Eignung eingegangen.

Das Produktqualitätsmodell der ISO 25010 (*ISO/IEC 25010*, 2023) unterteilt die funktionale Eignung in drei Untermerkmale. Obwohl alle diese Untermerkmale durch den funktionalen Test bewertet werden können, werden sie nicht von jeder funktionalen Testaktivität gleichermaßen gut adressiert. Testanalysten sollten in der Lage sein, die geeigneten Teststufen und Testverfahren auszuwählen, um Produktrisiken in Bezug auf ein bestimmtes Untermerkmal der funktionalen Eignung anzugehen.

Der **funktionale Vollständigkeitstest** sollte alle spezifizierten Aufgaben der Software und die Ziele der vorgesehenen Benutzer überdecken. Die Schlüsselfrage ist, ob alles, was verlangt wird, implementiert ist.

Die funktionale Vollständigkeit sollte so früh wie möglich durch ein Review der Anforderungen in sequenziellen Entwicklungsmodellen bzw. durch die Diskussion von User-Storys, einschließlich der Akzeptanzkriterien, während des gemeinsamen Schreibens von User-Storys in der agilen Softwareentwicklung angesprochen werden. Bei Systemtests, Systemintegrationstests und Abnahmetests kann die funktionale Vollständigkeit dynamisch getestet werden.

Verhaltensbasierte Testverfahren wie der szenariobasierte Test sind gut geeignet, aber auch andere Black-Box-Testverfahren sind anwendbar. Die Verfolgbarkeit zwischen Testbasis, Testbedingungen und Testfällen ist für die Bestimmung der erreichten funktionalen Vollständigkeit von entscheidender Bedeutung.

Der **funktionale Korrektheitstest** beantwortet die Frage, ob die Istergebnisse für gültige und ungültige Eingaben korrekt (z. B. genau, präzise und konsistent) sind. Es ist entscheidend, ein effektives Testorakel zu finden, das die erwarteten Ergebnisse im Detail liefert.

Funktionale Korrektheit kann auf jeder Teststufe getestet werden. Im Sinne von Shift-Left sollten die meisten Tests zur funktionalen Korrektheit bei Komponenten- und Komponentenintegrationstests durchgeführt werden. Auch wenn Testanalysten nicht dafür verantwortlich sind, sollten sie zu diesen Teststufen beitragen, um die Testziele bestmöglich zu erreichen.

Alle Black-Box-Testverfahren, erfahrungsbasierten Testverfahren und auf Zusammenarbeit basierenden Testansätze sind geeignet.

Der **funktionale Angemessenheitstest** verifiziert, ob die Funktionen die Erfüllung der festgelegten Aufgaben und Ziele erleichtern. Der Schwerpunkt liegt auf der Prüfung, ob alles, was implementiert wurde, die Bedürfnisse der Benutzer erfüllt.

Das Testen der funktionalen Angemessenheit kann auch Reviews der Entwürfe von Benutzungsschnittstellen umfassen, insbesondere bei interaktiven Anwendungen. Dynamische Tests beginnen mit Systemtests und Abnahmetests in sequenziellen Entwicklungsmodellen bzw. mit Demo-Sitzungen in der agilen Softwareentwicklung.

Exploratives Testen und auf Zusammenarbeit basierende Testansätze sind am besten geeignet. Darüber hinaus sind auch verhaltensbasierte Black-Box-Testverfahren geeignet.

4.2 Gebrauchstauglichkeitstest

Gebrauchstauglichkeit (Usability) bezieht sich auf ein weit gefasstes Konzept von benutzerbezogenen Qualitätsmerkmalen, das die Interaktionsfähigkeit aus dem Produktqualitätsmodell (*ISO/IEC 25010*, 2023) und die Nützlichkeit aus dem ISO-25019-Quality-in-Use-Modell (*ISO/IEC 25019*, 2023) umfasst. Mehr über Gebrauchstauglichkeitstests kann man in (ISTQB-UT, v1.0 DE), (*ISO 9241-210*, 2019) und (UXQB-FL, v4.01) finden.

Gebrauchstauglichkeitstests konzentrieren sich in der Regel auf die Bewertung der folgenden Aspekte:

- Interaktionsfähigkeit – Befähigung der Benutzer, Aufgaben in bestimmten Nutzungskontexten (*ISO/IEC 25010*, 2023) effektiv, effizient und zufriedenstellend zu erledigen.
- Benutzererlebnis – Befassung mit den Wahrnehmungen der Benutzer vor, während und nach der Interaktion mit dem Testobjekt.
- Zugänglichkeit – Gewährleistung, dass Benutzer mit Behinderungen, unterschiedlichen kulturellen Hintergründen oder Sprachbarrieren das System sowohl effektiv als auch effizient nutzen können.

Testanalysten können bereits ab einer frühen Phase an Gebrauchstauglichkeitstests mitwirken, indem sie ihr Wissen über die Benutzerzielgruppen nutzen, einschließlich deren Ziele, Nutzungskontext, möglicher Schwierigkeiten bei der Nutzung des Systems oder negativer Benutzererlebnisse.

Testanalysten können zu den wichtigsten Verfahren der benutzerzentrierten Evaluierung wie folgt beitragen:

- **Gebrauchstauglichkeitsreviews** werden von Gutachtern durchgeführt, um potenzielle Gebrauchstauglichkeitsprobleme und Abweichungen von festgelegten Kriterien zu ermitteln. Die Reviewarten können von informellen Reviews bis hin zu Inspektionen reichen. Der Gutachter kann die Reviewkriterien an die spezifischen Bedürfnisse der Benutzergruppen, die jeweiligen Unternehmensziele und -prioritäten und den Nutzungskontext anpassen (z. B. eine generische Gebrauchstauglichkeits-Checkliste erstellen).
- **Gebrauchstauglichkeitstestsitzungen** beziehen zukünftige Benutzer oder deren Vertreter mit ein, die versuchen, vordefinierte Aufgaben zu lösen, um zu bewerten, ob die Aufgaben effektiv, effizient und zufriedenstellend erledigt werden können. Testanalysten können dazu beitragen, Szenarien für die Gebrauchstauglichkeitstestsitzungen nach Personas, Benutzergruppen oder Nutzungsprofilen zu entwerfen.
- **Benutzerfragebögen oder Umfragen**, einschließlich Bewertung und Feedback, messen die Zufriedenheit der Benutzer. Beispiele für Benutzerfragebögen sind das Software Usability Measurement Inventory (SUMI, 1991) und das Website Analysis and Measurement Inventory (WAMMI, 1999). Testanalysten können dabei helfen, einen Fragebogen zu entwerfen und die Antworten auszuwerten, um die spezifischen Ziele der anvisierten Benutzer in ihrem Nutzungskontext anzusprechen.

Zugänglichkeitstests haben vor allem das Testziel, die Konformität mit den Standards zu verifizieren. Die internationale Norm Web Content Accessibility Guidelines (WCAG, 2023) definiert drei Konformitätsstufen (A, AA und AAA), die einen zunehmenden Grad der Zugänglichkeit von Webinhalten darstellen. Zu den nationalen Normen gehören der britische Equality Act (UK Government, 2010) und der amerikanische Americans Disabilities Act (U.S. Department of Justice, 2010). Testanalysten können durch die Analyse des Nutzungskontexts die erforderliche Konformitätsstufe und die spezifischen Bedürfnisse der vorgesehenen Zielgruppe ermitteln.

4.3 Flexibilitätstest

Der Flexibilitätstest (auch bekannt als Übertragbarkeitstest oder Portabilitätstest) verifiziert, ob das Testobjekt an Änderungen seines Nutzungskontextes oder seiner Systemumgebung angepasst werden kann. Das ISO-25010-Produktqualitätsmodell (*ISO/IEC 25010*, 2023) unterscheidet zwischen den folgenden Untermerkmalen von Flexibilität:

- Anpassbarkeit
- Skalierbarkeit
- Installierbarkeit
- Austauschbarkeit

Flexibilität hat sowohl technische als auch nicht-technische Aspekte. In diesem Abschnitt geht es darum, wie Testanalysten zum Anpassbarkeitstest und Installierbarkeittest beitragen können. Skalierbarkeit und Austauschbarkeit beziehen sich auf technische Aspekte und werden in (ISTQB-PT, v1.0 DE) bzw. (ISTQB-TTA, v4.0 DE) behandelt.

Der **Anpassbarkeitstest** verifiziert, ob das Testobjekt an die vorgesehene Zielhardware, Zielsoftware oder andere Betriebs- oder Nutzungsumgebungen angepasst oder übertragen werden kann.

Testanalysten unterstützen den Anpassbarkeitstest, indem sie die beabsichtigten Zielumgebungen identifizieren (z. B. die Versionen der unterstützten mobilen Betriebssysteme und die Versionen der möglicherweise verwendeten Browser) und Tests entwerfen, die Kombinationen dieser Umgebungen überdecken. Da hierfür Testdaten für verschiedene Umgebungsparameterkonfigurationen erforderlich sind, werden häufig Testverfahren wie kombinatorische Tests angewendet (siehe *Abschnitt 3.1.2*). Ein weiteres Beispiel ist die Integration verschiedener Komponenten in Kundenprojekte, um die Kompatibilität in verschiedenen Zielumgebungen sicherzustellen. Je nach Produktrisiko entwerfen die Testanalysten Smoke-Tests oder eine umfangreichere Testsuite und führen sie durch, um zu überprüfen, ob das Testobjekt korrekt an die Zielumgebung angepasst ist.

Durch die Einhaltung bewährter Praktiken für Anpassbarkeitstests (z. B. frühzeitige Definition von Zielumgebungen, kombinatorische Tests, Smoke-Tests in neuen Umgebungen und Überwachung umgebungsspezifischer Fehlerzustände) können Testanalysten Fehler aufdecken, die die Lebensdauer und Gebrauchstauglichkeit der Software einschränken könnten (z. B. die Bedienbarkeit auf unterschiedlichen Bildschirmgrößen). Die Arbeit der Testanalysten beim Anpassbarkeitstest sollte auch durch automatisierte plattformübergreifende Tests unterstützt werden, die von Testautomatisierern implementiert wurden. Ein gründlicher Anpassbarkeitstest kann umgebungsspezifische Probleme frühzeitig aufdecken und dazu beitragen, häufige und größere Aktualisierungen oder Restrukturierungen nach der Bereitstellung zu vermeiden und die Kosten für die Wartung zu senken.

Der **Installierbarkeitstest** verifiziert, ob das Testobjekt in bestimmten Umgebungen korrekt installiert, deinstalliert, aktualisiert und neu konfiguriert werden kann. Der Installierbarkeitstest geht über die reine Prüfung hinaus, ob der Installationsvorgang vollständig durchläuft.

Die typischen Testziele für die Installierbarkeit, auf die sich die Testanalysten konzentrieren, sind:

- Überprüfen, ob die Installationsverfahren unter verschiedenen Umgebungsparameterkonfigurationen korrekt ausgeführt werden. Dazu sind Testverfahren wie kombinatorische Tests hilfreich, ähnlich wie beim Anpassbarkeitstest.
- Entwerfen und Ausführen von Tests, um festzustellen, ob das Testobjekt nach der Installation oder Aktualisierung ordnungsgemäß funktioniert.
- Überprüfen, wie einfach es für Benutzer ist, die Software zu installieren, zu deinstallieren oder zu aktualisieren. Dazu gehört auch das Review der Installationsdokumentation.
- Testen des Verhaltens in Bezug auf Berechtigungen, insbesondere für mobile Anwendungen (siehe ISTQB-MAT, v1.0 DE).

4.4 Kompatibilitätstest

Der Kompatibilitätstest verifiziert, ob ein Testobjekt bei seiner Nutzung mit anderen Komponenten oder Systemen kompatibel ist. Das ISO- 25010-Produktqualitätsmodell (*ISO/IEC 25010*, 2023) unterscheidet zwischen zwei Untermerkmalen von Kompatibilität: Interoperabilität und Koexistenz. Daher kann der Kompatibilitätstest in die folgenden Testarten unterteilt werden:

- Der **Interoperabilitätstest** verifiziert die Kompatibilität des Testobjekts mit Komponenten oder Systemen, mit denen es interagieren soll. Dieser Test nutzt meistens funktionale Black-Box-Testverfahren, so dass die Testanalysten normalerweise für diese Testart verantwortlich sind.
- Der **Koexistenztest** verifiziert, ob das Testobjekt seine Zielumgebung mit anderen Komponenten oder Systemen teilen kann, ohne diese zu beeinträchtigen. Diese technische Testart wird in (ISTQB-TTA, v4.0 DE) behandelt.

Das Ziel des Interoperabilitätstests ist es, zu überprüfen, ob zwei oder mehr Komponenten oder Systeme Informationen austauschen und die ausgetauschten Informationen gegenseitig nutzen können. Wenn der Informationsaustausch eine Datenumwandlung beinhaltet, muss der Interoperabilitätstest auch die Verifizierung dieser Umwandlung umfassen. Der Interoperabilitätstest ist besonders wichtig, wenn mehrere Systeme zusammenarbeiten, Daten austauschen oder Aufgaben gemeinsam ausführen müssen. Dies ist insbesondere bei modernen Softwarearchitekturen wie Cloud-Lösungen, Webservices, Microservices, Containerisierung und dem Internet der Dinge der Fall.

Interoperabilität findet in der Regel auf verschiedenen Architekturebenen statt. Testanalysten müssen die möglichen Interaktionen verstehen, um geeignete Testbedingungen zu definieren, die diese überdecken. Möglicherweise sind nicht alle Interaktionen dokumentiert. Die Testanalysten können Informationen über die Interaktionen indirekt aus der Architektur- und Entwurfsdokumentation abrufen. Daher ist es von entscheidender Bedeutung, diese Dokumentation zu verstehen, um sicherzustellen, dass alle wichtigen Aspekte der Interaktionen getestet werden.

Der Interoperabilitätstest kann Fehlerzustände in folgenden Bereichen aufdecken:

- Datentransformationen für den Datenaustausch

- Interpretation oder Verwendung der ausgetauschten Daten
- Kommunikationsflüsse und -protokolle
- Konformität mit Standards
- Ende-zu-Ende-Funktionalität
- Entwurfsdokumentation

Der Interoperabilitätstest wird in der Regel während der Integrationstests durchgeführt. Black-Box-Testverfahren, wie datenbasierte Testverfahren, die sich auf die ausgetauschten Daten konzentrieren, verhaltensbasierte Testverfahren für die Interpretation oder Verwendung der Daten und Ende-zu-Ende-Funktionalität oder regelbasierte Testverfahren für die Datentransformation, sind für Interoperabilitätstests gut geeignet. Erfahrungsbasierte Testverfahren können den Black-Box-Test sinnvoll ergänzen. Testfälle aus dem funktionalen Test oder Anpassbarkeitstest können für den Interoperabilitätstest wiederverwendet werden.

Beispiele für allgemeine Normen zur Interoperabilität sind (*ISO 15745*, 2003) und (*ISO 16100*, 2009). ETSI (*ETSI EG 202 237 v1.2.1*, 2010) liefert ein Beispiel für eine konkrete Methodik zum Testen der Interoperabilität im Telekommunikationsbereich.

5 Fehlerprävention in der Software – 225 Minuten

Schlüsselbegriffe

ad-hoc-Review, checklistenbasiertes Review, Fehlerprävention, Grundursachenanalyse, modellbasierter Test, perspektivisches Lesen, Reviewverfahren, rollenbasiertes Review, szenariobasiertes Review, Testergebnis

Lernziele für Kapitel 5:

Die Lernenden können ...

5.1 Praktiken zur Fehlerprävention

TA-5.1.1 (K2) ... erklären, wie Testanalysten zur Fehlerprävention beitragen können

5.2 Unterstützung der Fehlereindämmung innerhalb der Phase

TA-5.2.1 (K3) ... ein Modell des Testobjekts verwenden, um Fehlerzustände in einer Spezifikation zu erkennen.

TA-5.2.2 (K3) ... Reviewverfahren auf ein Dokument der Testbasis anwenden, um Fehlerzustände zu finden

5.3 Verminderung des Wiederauftretens von Fehlerzuständen

TA-5.3.1 (K4) ... Testergebnisse analysieren, um mögliche Verbesserungen bei der Erkennung von Fehlerzuständen zu identifizieren

TA-5.3.2 (K2) ... erläutern, wie die Klassifizierung von Fehlerzuständen die Grundursachenanalyse unterstützt

Einführung in die Fehlerprävention von Software

Das Ziel der Fehlerprävention besteht darin, Maßnahmen zu ergreifen, die die Wahrscheinlichkeit des (erneuten) Vorkommens von Fehlerzuständen in Arbeitsprodukten verringern und die Ausbreitung von Fehlerzuständen auf nachfolgende Phasen des SDLC eindämmen. Diese Bemühungen führen zu einer Reihe von wichtigen Vorteilen, wie z. B. reduzierte Kosten und Arbeitsaufwand, erhöhte Produktivität und verbesserte Qualität der Produkte. Die Fehlerprävention liegt in der Verantwortung des gesamten Teams. Testanalysten können mit ihrem spezifischen Wissen und ihrer Erfahrung zur Fehlerprävention beitragen.

Zu den Praktiken der Fehlerprävention gehören:

- Dem Einbringen von Fehlerzuständen vorbeugen, was zu den Qualitätssicherungsaktivitäten gehört,
- Dem Übergehen von Fehlerzuständen in nachfolgende Phasen des SDLC vorbeugen (siehe *Abschnitt 5.2*),
- Dem Wiederauftreten von erkannten Fehlerzuständen vorbeugen (siehe *Abschnitt 5.3*).

5.1 Praktiken der Fehlerprävention

Testanalysten können auf verschiedene Weise zur Fehlerprävention beitragen, indem sie ihre branchenspezifischen Fachkenntnisse, ihre Testexpertise und ihre analytischen Fähigkeiten einsetzen. Beispiele dafür sind:

- Teilnahme an der Risikoanalyse – Sicherstellung, dass ermittelte Risiken angemessen beachtet/mitigiert werden (z. B. Auswahl der am besten geeigneten Testverfahren).
- Review von Anforderungen, Modellen und Spezifikationen – ermöglicht die frühzeitige Erkennung von Fehlerzuständen in der Testbasis und verhindert, dass diese in den Code übergehen, wodurch die Kosten für ihre Behebung erheblich gesenkt werden.
- Teilnahme an Retrospektiven – ermöglicht die Identifikation potenzieller Verbesserungen in Testanalyse, Testentwurf, Testrealisierung und Testdurchführung, um besser zu verhindern, dass Fehlerzustände in die nächste Phase übergehen (z. B. durch Verwendung effektiverer Testverfahren, gezieltes Testen bestimmter Risikobereiche oder Verbesserung von Testdaten und Testumgebungen, um sowohl falsch-positive Ergebnisse als auch falsch-negative Ergebnisse zu reduzieren).
- Sammlung und Auswertung von Daten zu Fehlerzuständen (Fehlerdaten) – unterstützt die Grundursachenanalyse und Prozessverbesserung durch Sammlung detaillierter Daten über Fehlerzustände, um deren Klassifizierung und statistische Analyse zu ermöglichen.
- Teilnahme an der Grundursachenanalyse – beugt dem Wiederauftreten von Fehlerzuständen durch Vorschläge für Korrekturmaßnahmen zur Beseitigung der ermittelten Grundursachen vor.

Neben der Beteiligung an der Fehlerprävention bewerten Testanalysten (in der Regel in Absprache mit den Testmanagern), ob die vorgeschlagenen Maßnahmen die gewünschte Wirkung erzielt haben. Beispiele für Metriken, die helfen, die Effektivität dieser Maßnahmen zu bewerten, sind:

- Wirksamkeit der Fehlerbehebung (DRE – defect removal efficiency) – misst das Verhältnis zwischen den vor der Freigabe behobenen Fehlerzuständen und der Gesamtzahl der Fehlerzustände. Der (unbekannte) Nenner kann geschätzt oder durch die Gesamtzahl der bis zu einem bestimmten Zeitpunkt (z. B. sechs Monate nach der Freigabe) gefundenen Fehlerzustände ersetzt werden. Eine hohe DRE deutet darauf hin, dass weniger Fehlerzustände in die Produktion gelangen, was auf eine

bessere Fehlerprävention hinweisen könnte. DRE unterscheidet jedoch nicht zwischen Fehlerzuständen, die durch Fehlerprävention und Fehlerfindung entdeckt werden.

- Wirksamkeit der Fehlereindämmung innerhalb der Phase (PCE – phase containment effectiveness) – misst die Anzahl der in derselben Phase eingebrachten und beseitigten Fehlerzustände im Verhältnis zur Gesamtzahl der in dieser Phase eingebrachten Fehler. Eine hohe PCE zeigt an, dass weniger Fehlerzustände in spätere Phasen übergehen.
- Qualitätskosten – veranschaulicht die Beziehung zwischen Fehlerpräventions-, Fehlerfindungs- und Fehlerbehebungskosten (siehe ISTQB-TM, v3.0 DE, Abschnitt 3.2.1).

5.2 Unterstützung der Fehlereindämmung innerhalb der Phase

Ziel der Phaseneindämmung ist es, Fehlerzustände in derselben Phase des SDLC zu erkennen und zu beseitigen, in der sie eingebracht wurden. In der agilen Softwareentwicklung kann der Shift-Left-Ansatz in ähnlicher Weise verwendet werden. Diese Vorgehensweise reduziert die Qualitätskosten. Da die Testbasis ein wichtiger Input für die Testanalyse und den Testentwurf ist, können Testanalysten am besten zur Fehlereindämmung innerhalb der Phase beitragen, indem sie die Qualität der Testbasis bewerten. Eine frühzeitige Beachtung der Qualität der Testbasis minimiert spätere Aufwände und verhindert die Ausbreitung von Fehlerzuständen in nachfolgende Phasen. In diesem Lehrplan werden zwei gängige Möglichkeiten für Testanalysten erörtert, um Fehlerzustände in der Testbasis zu finden: die Modellierung zu Testzwecken und das Review der Testbasis unter Verwendung verschiedener Reviewverfahren (*ISO/IEC 20246*, 2017).

5.2.1 Verwendung von Modellen zur Fehlerfindung

Die Modellierung ermöglicht Abstraktionen eines Systems auf einem bestimmten Präzisions- und Detailniveau. Sie hilft Stakeholdern, das zu entwickelnde System besser zu verstehen. Wie unten erläutert, kann die Modellierung die Fehlereindämmung innerhalb einer Phase auf mindestens drei Arten unterstützen:

- Finden von Fehlerzuständen in Spezifikationen
- Finden von Fehlerzuständen in Modellen
- Finden von Fehlerzuständen in Testobjekten durch modellbasiertes Testen

Finden von Fehlerzuständen in Spezifikationen: Spezifikationen werden oft als informell geschriebener Text bereitgestellt. Testanalysten können die Spezifikationen mithilfe von Modellen formalisieren. Bei der Erstellung eines Modells werden die Testbedingungen (z. B. Anforderungen) auf Modellelemente abgebildet und mit diesen zwecks Verfolgbarkeit verknüpft. Durch die Formalisierung und Visualisierung der Testbedingungen in einem Modell werden Fehlerzustände wie Unvollständigkeit, Inkonsistenzen oder Mehrdeutigkeiten effizient aufgedeckt. Die Stärke der Modellierung liegt darin, dass eine Spezifikation transformiert wird, während Reviews sie in ihrer ursprünglichen Form prüfen. Dadurch tragen Testanalysten auch dazu bei, geeignete Lösungen für die festgestellten Fehlerzustände zu finden.

Zu den datenbasierten Modellen gehören Wertebereichsmodelle und Modelle für kombinatorische Tests. Sie ermöglichen es Testanalysten, Fehlerzustände wie überlappende Partitionen, Lücken in der Überdeckung des Wertebereichs, leere Partitionen oder fehlende oder falsche Parameterkombinationen zu finden.

Zu den verhaltensbasierten Modellen gehören CRUD-Matrizen, zustandsbasierte Modelle oder Szenariomodelle. Sie ermöglichen es Testanalysten, unvollständige oder inkonsistente Lebenszyklen von Entitäten,

fehlende oder fehlerhafte Zustandsübergänge, Verklemmungen (Deadlocks), Endlosschleifen, mehrdeutiges oder inkonsistentes Systemverhalten oder fehlende Ausnahmebehandlung zu erkennen.

Regelbasierte Modelle wie Entscheidungstabellen oder metamorphe Relationen ermöglichen es Testanalysten, Fehlerzustände in Geschäftsregeln wie Auslassungen, Inkonsistenzen, Mehrdeutigkeiten, Redundanzen, fehleranfällige Szenarien oder komplexe Geschäftslogik zu erkennen.

Die Modellierung kann auch Fehlerzustände wie Inkonsistenzen bei der Benennung, den Datenwerten (z. B. Grenzen) und den Ein- oder Ausgaben aufdecken. Sie kann auch fehlende, unvollständige, mehrdeutige oder unnötige Informationen identifizieren.

Finden von Fehlerzuständen in Modellen: Wenn die Testbasis Modelle enthält, können Testanalysten diese Modelle analysieren, um Fehlerzustände zu finden. In diesem Lehrplan wird die Fehlerfindung bei drei typischen Modellen behandelt, die beim Testen von Software verwendet werden: Zustandsübergangsdiagramme (siehe *Abschnitt 3.2.2*), Aktivitätsdiagramme (siehe *Abschnitt 3.2.3*) und Entscheidungstabellen (siehe *Abschnitt 3.3.1*). Beispiele für Fehlerzustände in den oben genannten Modellen sind:

- Zustandsübergangsdiagramm – fehlende/falsche Zustände, ungeeignete Übergänge, falsche Wächterbedingungen (guard conditions) oder Aktionen, redundante oder unerreichbare Zustände und nicht-deterministisches Verhalten
- Aktivitätsdiagramm – fehlende oder unerreichbare Aktionen, Sackgassen (im Kontrollfluss), falsche Reihenfolge von Aktionen, falsche, nicht-exklusive oder unvollständige Wächterbedingungen (guard conditions) in Entscheidungsknoten, fehlende Synchronisationspunkte oder ungeeignet synchronisierte parallele Abläufe, die zu unbeabsichtigtem Verhalten führen können
- Entscheidungstabelle – überlappende, inkonsistente oder nicht durchführbare Regeln, Unvollständigkeit (z. B. fehlende Kombinationen von Bedingungen oder fehlende Aktionen für eine bestimmte Kombination von Bedingungen)

Alle Modelle können auch Fehlerzustände wie Syntaxfehler, Tippfehler, Duplikate und inkonsistente Benennungen von Modellelementen enthalten.

Finden von Fehlerzuständen durch modellbasiertes Testen (MBT): Bei diesem Testansatz entwirft und generiert ein MBT-Werkzeug Testfälle und andere Testmittel des Testentwurfs auf der Grundlage eines MBT-Modells und von Testauswahlkriterien, die von den Testanalysten definiert wurden. MBT findet effektiv Anomalien in Spezifikationen, da es eine umfassende Überdeckung und systematische Erkundung des erwarteten Verhaltens des Testobjekts gemäß dem MBT-Modell ermöglicht. MBT-Modelle, insbesondere grafische, fördern die Kommunikation mit den Stakeholdern und schaffen eine gemeinsame Einschätzung sowie ein gemeinsames Verständnis der Testbasis. Weitere Einzelheiten zu MBT sind im Lehrplan Model-Based Tester (ISTQB-MBT, v1.1) zu finden.

5.2.2 Anwendung von Reviewverfahren

Eine gut definierte Testbasis dient als Eckpfeiler für die Sicherstellung der Qualität von Arbeitsprodukten und den Erfolg von Projekten. Reviews der Testbasis helfen, Fehlerzustände frühzeitig zu erkennen und zu beheben. Dadurch wird verhindert, dass Fehlerzustände in nachfolgende Phasen übergehen.

Testanalysten können während des individuellen Reviews verschiedene Reviewverfahren einsetzen, um Fehlerzustände in der Testbasis zu identifizieren. Die Auswahl des am besten geeigneten Reviewverfahrens kann die Effizienz und Effektivität des Reviews erhöhen. Bei dieser Auswahl sollten Faktoren wie Reviewziele, Projektziele, verfügbare Ressourcen, Art der Testbasis, damit verbundene Risiken, Geschäftsbereich und Unternehmenskultur berücksichtigt werden.

Im Folgenden werden fünf Reviewverfahren vorgestellt, die von den Gutachtern häufig verwendet werden.

Ad-hoc-Reviews werden von Gutachtern informell und ohne strukturiertes Verfahren durchgeführt. Die Gutachter erhalten nur wenige oder gar keine Anweisungen, wie die Aufgabe auszuführen ist. Ad-hoc-Reviews erfordern wenig Vorbereitung und sind in hohem Maße von den Fähigkeiten der Gutachter abhängig. Während des Reviews lesen die Gutachter die Testbasis und dokumentieren die Anomalien, sobald sie diese entdecken. Dieses Reviewverfahren kann, wenn es nicht koordiniert wird, zu einer hohen Anzahl von doppelten Berichten über Anomalien von mehreren Gutachtern führen.

Beim **checklistenbasierten Review** wird die Testbasis anhand einer vordefinierten Checkliste bewertet. Checklisten erinnern Gutachter daran, bestimmte Punkte zu überprüfen, und können das Review weniger subjektiv werden lassen. Checklisten können generisch oder spezifisch für Qualitätsmerkmale, Testziele oder der Art der Testbasis sein. Testanalysten passen die Checkliste an die Art der Testbasis, die Risikostufe oder die Testbedingungen an. Dadurch wird sichergestellt, dass sich das Review auf die wichtigsten Aspekte der Testbasis konzentriert. Checklisten sollten regelmäßig mit bisher übersehenen Fehlerzuständen aktualisiert werden. Checklisten aktuell zu halten verhindert, dass neu entdeckte Arten von Anomalien übersehen werden. Checklisten sind nicht allumfassend, daher sollten Testanalysten sich nicht nur auf die Prüfung der Checklisteneinträge beschränken. Dadurch können auch Fehlerzustände und Anomalien erkannt werden, die in den Checklisten möglicherweise nicht explizit aufgeführt sind.

Szenariobasierte Reviews beinhalten das Simulieren eines Prozesses oder einer Aktivität, um Anomalien zu erkennen und die Testbasis zu verfeinern. Dieses Reviewverfahren ist am effektivsten, wenn die Testbasis ein szenariobasiertes Format hat, wie z. B. ein Anwendungsfall- oder Aktivitätsdiagramm. In solchen Fällen können Gutachter einen "Probelauf" (dry run) auf Grundlage der erwarteten Nutzung des Arbeitsprodukts durchführen. Szenarien repräsentieren wertvolle Richtlinien, die Gutachter sind jedoch nicht auf dokumentierte Szenarien beschränkt und sollten auch über die Szenarien hinaus denken, um weitere Anomalien zu identifizieren.

Beim **rollenbasierten Review** werden den Gutachtern bestimmte Rollen oder Verantwortlichkeiten zugewiesen. Typische Rollen basieren auf bestimmten Endbenutzertypen (z. B. erfahrene, unerfahrene, ältere oder jüngere Benutzer) oder bestimmten Rollen im Unternehmen (z. B. Administrator oder regulärer Benutzer). Jede Rolle kann durch eine Persona beschrieben werden (d. h. eine konkrete, aber fiktive Figur, die entworfen wurde, um die Eigenschaften, Bedürfnisse, Ziele und Vorlieben einer bestimmten Gruppe von Benutzern zu repräsentieren). Durch die Verteilung der Zuständigkeiten auf die Gutachter aufgrund ihrer Rollen ermöglichen rollenbasierte Reviews, dass sich jeder Einzelne auf bestimmte Aspekte der Testbasis konzentrieren kann. Somit ist eine umfassende Überdeckung gewährleistet und gleichzeitig wird eine Duplizierung von Anomalien vermieden.

Beim **perspektivischen Lesen** wird die Testbasis aus verschiedenen Perspektiven oder Blickwinkeln (z. B. Designer, Tester, Marketing, Administrator und Benutzer) überprüft. Dies führt zu einem gründlicheren individuellen Review mit weniger Überschneidungen von Anomalien zwischen den verschiedenen Gutachtern. Darüber hinaus erfordert das perspektivische Lesen, dass die Gutachter versuchen, die zu prüfende Testbasis zu nutzen, um das Arbeitsprodukt zu erstellen, das sie daraus ableiten würden. Zum Beispiel würde ein Tester versuchen, einen Entwurf für Abnahmetests auf der Grundlage von Anforderungen zu erstellen, um zu sehen, ob alle erforderlichen Informationen enthalten sind.

5.3 Verminderung des Wiederauftretens von Fehlerzuständen

Testanalysten können proaktiv dazu beitragen, das Wiederauftreten von Fehlerzuständen in der Software zu minimieren. In diesem Lehrplan werden zwei Ansätze zur Verringerung des erneuten Auftretens von

Fehlern erörtert: die Analyse von Testergebnissen zur Verbesserung der Testanalyse und des Testentwurfs sowie die Unterstützung der Grundursachenanalyse durch Fehlerklassifikation.

5.3.1 Analyse von Testergebnissen zur Verbesserung der Fehlerfindung

Testergebnisse ermöglichen es den Testanalysten, Fehlerwirkungen zu identifizieren, liefern ihnen aber zugleich auch Feedback, um die Effektivität bei der Fehlerfindung {.index} zu verbessern. Im Folgenden werden einige häufig verwendete Verfahren zur Analyse von Testergebnissen beschrieben.

Vergleich der Vorhersage mit tatsächlichen Anhäufungen von Fehlerzuständen: Einige wenige Komponenten enthalten üblicherweise die meisten Fehlerzustände (siehe ISTQB-CTFL, v4.0.2 DE, Abschnitt 1.3). Nach dem Testen können die Testanalysten fehleranfällige Bereiche vorhersagen und die vorhergesagten mit den tatsächlichen Fehleranhäufungen vergleichen. Im Falle von Diskrepanzen können in Bereichen, in denen mehr Fehlerzustände als erwartet gefunden wurden, intensivere Tests durchgeführt werden. Bei der Bestimmung der Anhäufung von Fehlerzuständen sollten messbare Kriterien für Klarheit und Konsistenz sorgen (z. B. Fehlerdichte und Fehlerschweregrad). Unter diesen Kriterien sollte der Fehlerschweregrad eine wichtige Rolle spielen. Kleine Anhäufungen von kritischen oder schwerwiegenden Fehlern sind in der Regel wichtiger (d. h., sie müssen intensiver getestet werden) als größere Anhäufungen von kleineren oder trivialen Fehlern.

Analye des Fehlerfindungsanteils (DDP): DDP (defect detection percentage) ist eine der wichtigsten Messgrößen für die Testeffektivität auf einer Teststufe. Bei der Berechnung der DDP sollte die Anzahl der entgangenen Fehlerzustände auf diejenigen beschränkt werden, die auf der betrachteten Teststufe hätten erkannt werden können. Klare Abgrenzungen für die Fehlerzählung, wie z. B. zeitliche Begrenzungen (z. B. Fehlerzustände, die innerhalb eines definierten Zeitrahmens nach der Freigabe gefunden werden) und Ausschlusskriterien (z. B. Fehlerzustände in Komponenten von Drittanbietern oder in spezifischen Kundenumgebungen), sollten festgelegt werden, um die Konsistenz sicherzustellen. Eine niedrige DDP für eine bestimmte Teststufe deutet auf einen hohen Anteil entgangener Fehler hin, das bedeutet, dass die Fehlerfindung ineffektiv ist. In einem solchen Fall sollten die Testanalysten die Gründe analysieren und Maßnahmen zur Verbesserung vorschlagen, um die Teststufe gezielter und gründlicher zu gestalten. Die DDP sollte idealerweise nach Fehlerschweregraden aufgeschlüsselt werden, da die Priorität zur Reduzierung entgangener Fehlerzustände in der Regel von deren Fehlerschweregrad abhängt.

Strukturelle Überdeckungsanalyse: Die strukturelle Überdeckungsanalyse bewertet, inwieweit Tests bestimmte Bereiche des Testobjekts durchlaufen haben. Die Identifizierung von Bereichen mit geringer Überdeckung ermöglicht es den Testanalysten, die Testaktivitäten gezielt auf diese Bereiche auszurichten. Eine Erhöhung der Überdeckung hilft dabei, neue und bisher entgangene Fehlerzustände zu finden. Strukturelle Überdeckungen wie Anweisungsüberdeckung, Zweigüberdeckung oder Neuronenüberdeckung werden in der Regel mit Testwerkzeugen gemessen. Bei der Auswahl der zusätzlich abzudeckenden Bereiche sollten deren Risikostufen berücksichtigt werden.

Analyse von Testlücken: Die Analyse von Testlücken bewertet, inwieweit die Tests die jüngsten Codeänderungen überprüft haben. Dies ermöglicht es den Testanalysten, zusätzliche Testaktivitäten gezielt auf besonders fehleranfällige Bereiche zu konzentrieren (d. h. neue Änderungen, die bislang überhaupt nicht getestet wurden), anstatt sich auf alle Bereiche mit geringer Überdeckung zu konzentrieren (z. B. Code, der seit Langem nicht mehr geändert wurde und in früheren Versionen getestet worden ist).

Analyse der Fehlerauftrittsmuster: Die Anzahl oder Dichte der in aufeinanderfolgenden Projektphasen (z. B. Iterationen) gefundenen Fehlerzustände kann mit Mustern verglichen werden, die die theoretische Verteilung dieser Werte über die Zeit beschreiben. Ein klassisches Beispiel für ein Fehlerauftrittsmuster ist das *Rayleigh*-Modell (Elsayed, 2021). Es weist einen einzelnen Höhepunkt auf und ist nach rechts verzerrt.

Dies zeigt, dass die erwartete Anzahl der gefundenen Fehlerzustände zunächst mit der Zeit zunimmt und nach Erreichen des Maximalwerts jedoch langsam gegen null abfällt. Auf Basis der Analyse eines solchen Musters lassen sich Rückschlüsse auf die Wirksamkeit bestehender Testfälle und deren Verbesserungspotenzial ziehen. Bleibt beispielsweise die Fehlerfindung konstant auf einem niedrigen Niveau, obwohl das Muster einen Anstieg erwarten lässt, kann dies darauf hindeuten, dass die vorhandenen Tests zu schwach sind und keine zusätzlichen Fehlerzustände aufdecken können.

Die oben beschriebenen Analysemethoden verwenden verschiedene fehlerbezogene Metriken, wie die Anzahl der Fehlerzustände oder DDP. Diese Metriken werden auf der Basis von Testergebnissen (z. B. Anzahl der bestandenen/fehlgeschlagenen Tests), Fehlerberichten und strukturellen Metriken (z. B. Codeüberdeckung oder Fehlerdichte) berechnet. Es ist jedoch zu beachten, dass sich diese Metriken nicht immer so einfach aus den Testergebnissen ableiten lassen, wie es zunächst erscheinen mag. Beispielsweise entspricht die Anzahl der fehlgeschlagenen Tests nicht zwingend der Anzahl der durch diese Tests entdeckten Fehlerzustände. Um die tatsächliche Anzahl der entdeckten Fehlerzustände zu ermitteln, müssen die Ergebnisse des Debugging-Prozesses sorgfältig analysiert werden, da zwischen Testergebnissen und Fehlerzuständen eine Viele-zu-viele-Beziehung bestehen kann. Mehrere Tests können denselben Fehlerzustand aufdecken oder ein einzelner Test kann mehrere Fehlerzustände aufdecken. Außerdem kann der Fehlerschweregrad von der Kritikalität der Tests abweichen, da ein kritischer Testfall aufgrund eines trivialen Fehlerzustands fehlgeschlagen sein kann.

5.3.2 Unterstützung der Grundursachenanalyse durch Fehlerklassifikation

Die Grundursachenanalyse (RCA - root cause analysis) ist eine Technik, mit der die zugrundeliegenden oder grundlegenden Ursachen eines Fehlerzustands und nicht nur dessen Symptome identifiziert und gezielt angegangen werden. RCA unterstützt einen strukturierten Ansatz zur Qualitätsverbesserung und zielt in erster Linie darauf ab, das erneute Auftreten von Fehlerzuständen zu verhindern. Die Testanalysten setzen verschiedene Techniken ein, um die Grundursachen von Fehlerzuständen und Fehlerwirkungen zu ermitteln (z. B. Fehlertaxonomien, die 5-Why-Methode, Ursache-Wirkungs-Diagramme und Pareto-Analyse).

Bei der klassischen RCA untersuchen Fachexperten einen Fehlerzustand nach dessen Behebung sehr detailliert. In der Regel gibt es jedoch viele Fehlerzustände, die analysiert werden müssen. Daher wäre es sehr ineffizient und zeitaufwändig, für jeden Fehlerzustand Präventivmaßnahmen zu planen. Eine Möglichkeit, dieses Problem anzugehen, besteht darin, die Fehlerzustände zu klassifizieren und anschließend die RCA für die auftretenden Fehlerzustandsarten durchzuführen.

Die Fehlerklassifikation basiert auf der Erkenntnis, dass einzelne Fehlerzustände eine Vielzahl von Informationen über den Entwicklungsprozess und das SUT enthalten. Die Fehlerklassifikation ermöglicht es den Testanalysten, Informationen über verschiedene Aspekte des Entwicklungsprozesses aus dem Fehlerzustand zu extrahieren und ihn in prozessbezogene Messgröße umzuwandeln. Dies ermöglicht wiederum Einblicke in typische Fehlerarten, die während der Entwicklung gemacht wurden – eine wertvolle Grundlage für die Verbesserung von Entwicklungsprozessen. Die Fehlerklassifikation schließt somit die Lücke zwischen der quantitativen Fehlerstatistik und der qualitativen RCA. Um RCA effektiv zu unterstützen, sollten die Fehlerzustände während des gesamten SDLC, vom frühen Testen bis zur Produktion, einheitlich klassifiziert werden.

Die Testanalysten sollten ihr Unternehmen bei der Standardisierung der Software-Fehlerklassifikation unterstützen. Dies wird die Kommunikation und den Austausch von Informationen über Fehlerzustände zwischen Entwicklern und Unternehmen verbessern und die RCA erleichtern.

Beispiele für Methoden zur Klassifizierung von Fehlerzuständen sind:

- Orthogonale Fehlerklassifikation (Chillarege, 1992), die jeden Fehlerzustand in orthogonale (d. h. sich gegenseitig ausschließende) Attribute klassifiziert, die sowohl bei der Meldung eines Fehlerzustands als auch bei seiner Behebung erfasst werden.
- IEEE 1044 (*IEEE 1044*, 2009), eine Standardklassifikation für Softwareanomalien, die einen Kernsatz von Attributen für die Klassifizierung von Fehlerwirkungen und Fehlerzuständen enthält.
- Schweregradbasierte Klassifizierung, die Fehlerzustände nach ihrem Fehlerschweregrad klassifiziert (z. B. kritisch, schwerwiegend, geringfügig, trivial).
- Fehlertaxonomie-Modelle, wie z. B. (Beizer, 1990) oder (Catolino u. a., 2019).

Fehlerzustände können auch mithilfe von Softwarequalitätsmodellen wie (*ISO/IEC 25010*, 2023) oder dem FURPS-Modell (Grady u. a., 1987) auf Qualitätsattribute abgebildet werden.

Weitere Informationen zu RCA finden sich in (ISTQB-ITP, v1.0).

6 Referenzen

Normen

ETSI EG 202 237 v1.2.1: Internet Protocol Testing (IPT), 2010. Standard. European Telecommunications Standards Institute.

IEEE 1044: Standard Classification for Software Anomalies, 2009. Standard. Institute of Electrical und Electronics Engineers.

ISO 15745: Industrial automation systems and integration – Open systems application integration framework, 2003. Standard. International Organization for Standardization.

ISO 16100: Industrial automation systems and integration – Manufacturing software capability profiling for interoperability, 2009. Standard. International Organization for Standardization.

ISO 26262: Road vehicles – functional safety – Part 6: Product development at the software level, 2018. Standard. International Organization for Standardization.

ISO 9241-210: Ergonomics of human-system interaction, part 210: Human-centred design for interactive systems, 2019. Standard. International Organization for Standardization.

ISO/IEC 20246: Software and systems engineering – Work product reviews, 2017. Standard. International Organization for Standardization.

ISO/IEC 25010: Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model, 2023. Standard. International Organization for Standardization.

ISO/IEC 25019: Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality-in-use model, 2023. Standard. International Organization for Standardization.

ISO/IEC/IEEE 29119-3: Software testing, Part 3: Test documentation, 2021. Standard. International Organization for Standardization.

ISO/IEC/IEEE 29119-4: Software testing, Part 4: Test techniques, 2021. Standard. International Organization for Standardization.

ISO/IEC/IEEE 29119-5: Software testing, Part 5: Keyword-Driven Testing, 2016. Standard. International Organization for Standardization.

OMG® BPMN: Business Process Model and Notation, version 2.0, 2013. Standard. Object Management Group, OMG®.

OMG® DMN: Decision Model and Notation, version 1.5, 2024. 2024-01. Standard. Object Management Group.

OMG® UML: Unified Modeling Language, version 2.5, 2017. Standard. Object Management Group.

RTCA DO-178C: Software Considerations in Airborne Systems and Equipment Certification, 2011. Standard. Radio Technical Commission for Aeronautics, RTCA Inc.

ISTQB® Dokumente

ISTQB-ACT, ISTQB®, 2019. *Certified Tester Specialist Mobile Acceptance Testing: Lehrplan, deutschsprachige Ausgabe*. 1.0 DE.

ISTQB-AI, ISTQB®, 2021. *Certified Tester AI Testing: Lehrplan, deutschsprachige Ausgabe*. 1.0 DE.

ISTQB-CTFL, ISTQB®, 2025. *Certified Tester Foundation Level: Lehrplan, deutschsprachige Ausgabe*. 4.0.2 DE.

ISTQB-ITP, ISTQB®, 2011. *Certified Tester Expert Level Improving the Test Process: Syllabus*. Version 1.0.

ISTQB-MAT, ISTQB®, 2019. *Certified Tester Specialist Mobile Application Testing: Lehrplan, deutschsprachige Ausgabe*. 1.0 DE.

ISTQB-MBT, ISTQB®, 2024. *Certified Tester Model-Based Tester: Syllabus*. Version 1.1.

ISTQB-PT, ISTQB®, 2019. *Certified Tester Performance Testing: Lehrplan, deutschsprachige Ausgabe*. 1.0 DE.

ISTQB-TAE, ISTQB®, 2024. *Certified Tester Test Automation Engineering: Lehrplan, deutschsprachige Ausgabe*. 2.0 DE.

ISTQB-TM, ISTQB®, 2025. *Certified Tester Advanced Level Test Management: Lehrplan, deutschsprachige Ausgabe*. 3.0 DE.

ISTQB-TTA, ISTQB®, 2022. *Certified Tester Advanced Level Technical Test Analyst: Lehrplan, deutschsprachige Ausgabe*. 4.0 DE.

ISTQB-UT, ISTQB®, 2020. *Certified Tester Usability Testing: Lehrplan, deutschsprachige Ausgabe*. 1.0 DE.

Bücher

AMMANN, Paul; OFFUTT, Jeff, 2008. *Introduction to Software Testing*. Cambridge University Press.

BEIZER, Boris, 1990. *Software Testing Techniques (2nd Ed.)* International Thomson Computer Press.

BINDER, Robert V., 2000. *Testing Object-Oriented Systems*. Addison-Wesley.

ELSAYED, Elsayed A., 2021. *Reliability Engineering*. Wiley.

FORGÁCS, Istvan; KOVÁCS, Attila, 2019. *Practical Test Design: Selection of traditional and automated test design techniques*. BCS, The Chartered Institute for IT.

FORGÁCS, Istvan; KOVÁCS, Attila, 2024. *Modern Software Testing Techniques*. Apress.

GRADY, Robert; CASWELL, Deborah, 1987. *Software Metrics: Establishing a Company-wide Program*. Prentice Hall.

HENDRICKSON, Elisabeth, 2013. *Explore It!* Pragmatic Bookshelf.

JORGENSEN, Paul, 2014. *Software Testing. A Craftsman's Approach*. CRC Press.

KANER, Cem; FALK, Jack; NGUYEN, Hung Q., 1999. *Testing Computer Software*. Wiley.

KANER, Cem; PADMANABHAN, Sowmya; HOFFMAN, Douglas, 2013. *The Domain Testing Workbook*. Context Driven Press.

KOOMEN, Tim; AALST, Leo van der; BROEKMAN, Bart; VROON, Michiel, 2008. *TMap Next - Praktischer Leitfaden für ergebnisorientiertes Softwaretesten*. dpunkt.verlag.

KUHN, D. Richard; YU, Lei; KACKER, Raghu N., 2013. *Introduction to Combinatorial Testing*. CRC Press.

Artikel

ALYAHYA, Sultan, 2020. Crowdsourced Software Testing: A Systematic Literature Review. *Information and Software Technology*. Jg. 127, S. 106363. Abger. unter DOI: 10.1016/j.infsof.2020.106363.

ANTONIOL, Giuliano; BRIAND, Lionel; DI PENTA, Massimiliano; LABICHE, Yvan, 2002. A case study using the round-trip strategy for state-based class testing. *Proceedings 13th International Symposium on Software Reliability Engineering*, S. 269–279. ISBN 0-7695-1763-3. Abger. unter DOI: 10.1109/ISSRE.2002.1173268.

ARCURI, Andrea; IQBAL, Muhammad Zohaib; BRIAND, Lionel, 2012. Random Testing: Theoretical Results and Practical Implications. *IEEE Transactions on Software Engineering*. Jg. 38, S. 258–277. Abger. unter DOI: 10.1109/TSE.2011.121.

BARR, Earl; HARMAN, Mark; MCMINN, Phil; SHAHBAZ, Muzammil; YOO, Shin, 2014. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering*. Jg. 41, Nr. 5, S. 507–525. Abger. unter DOI: 10.1109/TSE.2014.2372785.

BOCHMANN, Gregor; PETRENKO, Alexandre, 1994. Protocol Testing: Review of Methods and Relevance for Software Testing. *ISSTA '94: Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis*, S. 109–124. Abger. unter DOI: 10.1145/186258.187153.

CATOLINO, Gemma; PALOMBA, Fabio; ZAIDMAN, Andy; FERRUCCI, Filomena, 2019. Not All Bugs Are the Same: Understanding, Characterizing, and Classifying Bug Types. *Journal of Systems and Software*. Jg. 152, S. 165–181. Abger. unter DOI: 10.1016/j.jss.2019.03.002.

CHILLAREGE, R. et al., 1992. Orthogonal Defect Classification - A Concept for In-Process Measurements. *IEEE Transactions on Software Engineering*. Jg. 18, S. 943–956. Abger. unter DOI: 10.1109/32.177364.

CHOW, Tsun, 1978. Testing Software Design Modeled by Finite-State Machines. *IEEE Transactions on Software Engineering*. Jg. 4, S. 178–187. Abger. unter DOI: 10.1109/TSE.1978.231496.

COHEN, D.M.; DALAL, Siddhartha; KAJLA, A.; PATTON, G.C., 1994. The Automatic Efficient Test Generator (AETG) system. *Proceedings of 1994 IEEE International Symposium on Software Reliability Engineering*, S. 303–309. ISBN 0-8186-6665-X. Abger. unter DOI: 10.1109/ISSRE.1994.341392.

ENGSTRÖM, Emelie; RUNESON, Per; SKOGLUND, Mats, 2010. A systematic review on regression test selection techniques. *Information and Software Technology*. Jg. 52, S. 14–30. Abger. unter DOI: 10.1016/j.infsof.2009.07.001.

GHAZI, Ahmad Nauman; GARIGAPATI, Ratna; PETERSEN, Kai, 2017. Checklists to Support Test Charter Design in Exploratory Testing. *Agile Processes in Software Engineering and Extreme Programming. XP 2017*. ISBN 978-3-319-57632-9. Abger. unter DOI: 10.1007/978-3-319-57633-6_17.

HAREL, David, 1987. Statecharts: A Visual Formalism For Complex Systems. *Science of Computer Programming*. Jg. 8, S. 231–274. Abger. unter DOI: 10.1016/0167-6423(87)90035-9.

HUANG, Rubing; SUN, Weifeng; XU, Yinyin; CHEN, Haibo; TOWEY, Dave; XIA, Xin, 2019. A Survey on Adaptive Random Testing. *IEEE Transactions on Software Engineering*. Jg. 47, Nr. 10, S. 2052–2083. Abger. unter DOI: 10.1109/TSE.2019.2942921.

JENG, Bingchiang; WEYUKER, Elaine, 1994. A Simplified Domain-Testing Strategy. *ACM Transactions on Software Engineering and Methodology*. Jg. 3, S. 254–270. Abger. unter DOI: 10.1145/196092.193171.

JUERGENS, Elmar; PAGANO, Dennis; GOEB, Andreas, 2018. Test Impact Analysis: Detecting Errors Early Despite Large, Long-Running Test Suites. *Whitepaper, CQSE GmbH*.

KUHN, D. Richard; WALLACE, Dolores R.; GALLO, Albert M. Jr, 2004. Software Fault Interactions and Implications for Software Testing. *IEEE Transactions on Software Engineering*. Jg. 30, S. 418–421. Abger. unter DOI: 10.1109/TSE.2004.24.

LEICHT, Niklas; BLOHM, Ivo; LEIMEISTER, Jan Marco, 2017. Leveraging the Power of the Crowd for Software Testing. *IEEE Software*. Jg. 34, S. 62–69. Abger. unter DOI: 10.1109/MS.2017.37.

OFFUTT, A. Jefferson, 1992. Investigations of the software testing coupling effect. *ACM Transactions on Software Engineering and Methodology*. Jg. 1, S. 5–20.

RECHTBERGER, Vaclav; BURES, Miroslav; AHMED, Bestoun, 2022. Overview of Test Coverage Criteria for Test Case Generation from Finite State Machines Modelled as Directed Graphs. *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Valencia, Spain*, S. 207–214.

RWEMALIKA, Renaud; KINTIS, Marinos; PAPADAKIS, Mike; LE TRAON, Yves; LORRACH, Pierre, 2019. On the Evolution of Keyword-Driven Test Suites. *12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, S. 335–345.

SEGURA, Sergio; FRASER, Gordon; SANCHEZ, Ana; RUIZ-CORTÉS, Antonio, 2016. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering*, S. 46–53.

SEGURA, Sergio; TOWEY, Dave; ZHOU, Zhi Quan; CHEN, Tsong Yueh, 2020. Metamorphic Testing: Testing the Untestable. *IEEE Software*. Jg. 42, Nr. 9, S. 805–824.

WU, Huayao; NIE, Changhai; PETKE, Justyna; JIA, Yue; HARMAN, Mark, 2020. An Empirical Comparison of Combinatorial Testing, Random Testing and Adaptive Random Testing. *IEEE Transactions on Software Engineering*. Jg. 46, Nr. 3, S. 302–320.

Webseiten

COMPUTER SOCIETY, IEEE; JTC 1/SC7, ISO/IEC, 2024. *SE VOCAB: Software and Systems Engineering Vocabulary* [online]. [besucht am 2024-09-15]. Abger. unter: https://pascal.computer.org/.

CZERWONKA, Jacek, 2004. *Pairwise Testing: Combinatorial Test Case Generation* [online]. [besucht am 2024-09-15]. Abger. unter: www.pairwise.org.

HEALTH, U.S. Department of; SERVICES, Human, 2024. *Health Insurance Portability and Accountability Act: Standards for Privacy of Individually Identifiable Health Information (Privacy Rules)* [online]. [besucht am 2024-09-15]. Abger. unter: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html.

IREB-GLOSSARY, IREB®, 2024. *The CPRE Glossary: Core terms of Requirements Engineering* [online]. [besucht am 2024-09-15]. Abger. unter: https://glossary.istqb.org.

ISTQB-GLOSSARY, ISTQB®, 2024. *Standard Glossary of Terms Used in Software Testing* [online]. [besucht am 2024-09-15]. Abger. unter: https://glossary.istqb.org.

KOMMISSION, Europäische, 2016. *Datenschutz-Grundverordnung: Verordnung (EU) 2016/679* [online]. [besucht am 2024-09-15]. Abger. unter: https://commission.europa.eu/law/law-topic/data-protection/legal-framework-eu-data-protection_en?.

Site of Software Test Design, 2020 [online]. [besucht am 2024-09-15]. Abger. unter: https://test-design.org/.

SUMI, 1991. *Software Usability Measurement Inventory: Standard evaluation questionnaire for assessing quality of use* [online]. [besucht am 2024-09-15]. Abger. unter: sumi.uxp.ie.

U.S. DEPARTMENT OF JUSTICE, Civil Rights Division, 2010. *Americans with Disabilities Act: Guidance & Resource Materials* [online]. [besucht am 2024-09-15]. Abger. unter: www.ada.gov/resources/.

UK GOVERNMENT, Equalities Office, 2010. *Equality Act 2010: guidance: Information and guidance on the Equality Act 2010, including age discrimination and public sector Equality Duty.* [online]. [besucht am 2024-09-15]. Abger. unter: www.gov.uk/guidance/equality-act-2010-guidance.

WAMMI, 1999. *Website Analysis and Measurement Inventory: Professional service for analyzing user experience* [online]. [besucht am 2024-09-15]. Abger. unter: www.wammi.com.

WCAG, 2023. *Web Content Accessibility Guidelines 2.2: W3C Recommendation* [online]. [besucht am 2024-09-15]. Abger. unter: www.w3.org/TR/WCAG22/.

Die obigen Referenzen verweisen auf Information, die im Internet und an anderen Orten verfügbar sind. Obwohl diese Referenzen zur Zeit der Veröffentlichung dieses Lehrplans geprüft worden sind, kann das ISTQB® nicht dafür verantwortlich gemacht werden, wenn sie nicht mehr verfügbar sind.

7 Anhang A – Lernziele/kognitive Wissensstufen

Die spezifischen Lernziele für diesen Lehrplan sind am Anfang jedes Kapitels aufgeführt. Jedes Thema des Lehrplans wird entsprechend dem jeweiligen Lernziel geprüft.

Die Lernziele beginnen mit einem Aktionsverb, das dem jeweiligen kognitiven Wissensstand entspricht, wie unten aufgeführt.

Stufe 1: Sich erinnern (K1)

Die Lernenden können sich an einen Begriff oder ein Konzept erinnern, es erkennen und wiedergeben.

Aktionsverben: erinnern, erkennen

Hinweis: Für die Advanced-Level-Lehrpläne gibt es keine spezifischen Lernziele auf K1-Niveau. Der Lehrplaninhalt der Kapitel 1 - 5 und die Definitionen aller Begriffe, die als Schlüsselwörter direkt unter den Kapitelüberschriften aufgeführt sind, sollen jedoch bekannt sein (K1), auch wenn sie nicht ausdrücklich in den Lernzielen erwähnt werden.

Stufe 2: Verstehen (K2)

Die Lernenden können die Gründe oder Erklärungen für Aussagen zum Thema auswählen und können das zugehörige Konzept des Softwaretests zusammenfassen, vergleichen, einordnen und Beispiele dafür geben.

Aktionsverben: klassifizieren, vergleichen, differenzieren, unterscheiden, erklären, Beispiele geben, interpretieren, zusammenfassen

Beispiele	Anmerkungen
Die Lernenden können zwischen dem Testen der funktionalen Korrektheit, der funktionalen Angemessenheit und der funktionalen Vollständigkeit unterscheiden.	Sucht nach Unterschieden zwischen den Konzepten.
Die Lernenden können den CRUD-Test erläutern	
Die Lernenden können Beispiele für Anforderungen an Testumgebungen geben	
Die Lernenden können die Beteiligung des Testanalysten an verschiedenen Softwareentwicklungslebenszyklen zusammenfassen	

Stufe 3: Anwenden (K3)

Die Lernenden können ein Verfahren durchführen, wenn sie mit einer vertrauten Aufgabe konfrontiert werden, oder das richtige Verfahren auswählen und es auf einen gegebenen Kontext anwenden.

Aktionsverben: anwenden, umsetzen, vorbereiten, nutzen

Beispiele	Anmerkungen
Die Lernenden können den Wertebereichstest anwenden	Sollte sich auf ein Verfahren / eine Technik / einen Prozess etc. beziehen.
Die Lernenden können Test-Chartas für sitzungsbasiertes Testen vorbereiten	
Die Lernenden können den schlüsselwortgetriebenen Test zur Entwicklung von Testskripten nutzen	Kann in einem Lernziel verwendet werden, bei dem die Lernenden in der Lage sein sollen, eine Technik oder ein Verfahren zu nutzen. Ähnlich wie "anwenden".

Stufe 4: Analysieren (K4)

Die Lernenden können Informationen, die sich auf ein Verfahren beziehen, zum besseren Verständnis in ihre Bestandteile zerlegen und zwischen Fakten und Schlussfolgerungen unterscheiden. Eine typische Anwendung ist die Analyse eines Dokuments, einer Software oder einer Projektsituation und das Vorschlagen geeigneter Maßnahmen, um ein Problem zu lösen oder eine Aufgabe zu erfüllen.

Aktionsverben: analysieren, zerlegen, gliedern, Prioritäten setzen, auswählen

Beispiele	Anmerkungen
Die Lernenden können die Auswirkungen von Änderungen analysieren, um den Umfang von Regressionstests zu bestimmen	Nur in Verbindung mit einem messbaren Ziel der Analyse prüfbar. Sollte von der Form "Analysiere X in Bezug auf Y" (oder ähnlich) sein.
Die Lernenden können geeigneter Testverfahren zur Minderung von Produktrisiken in einer gegebenen Situation auswählen	Erforderlich, wenn die Auswahl eine Analyse erfordert.

Referenz

(Für die kognitiven Wissensstufen der Lernziele)

Anderson, L. W. und Krathwohl, D. R. (Hrsg.) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom’s Taxonomy of Educational Objectives, Allyn & Bacon

8 Anhang B – Matrix zur Verfolgbarkeit des geschäftlichen Nutzen (Business Outcomes) mit Lernzielen (Learning Objectives)

Dieser Abschnitt listet die Anzahl der Lernziele des vorliegenden Lehrplans auf, die mit dem geschäftlichen Nutzen in Verbindung stehen, sowie die Verfolgbarkeit zwischen dem geschäftlichen Nutzen des Lehrplans und den Lernzielen des Advanced Level Test Analyst.

Geschäftlicher Nutzen: Advanced Level Test Analyst		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
TA-BO1	Für den angewendeten Softwareentwicklungslebenszyklus angemessenes Testen unterstützen und durchführen.	6								
TA-BO2	Die Grundsätze des risikobasierten Tests anwenden.		3							
TA-BO3	Geeignete Testverfahren auswählen und anwenden, um die Erreichung der Testziele zu unterstützen.			13						
TA-BO4	Dokumentation mit angemessenem Detaillierungsgrad und geeigneter Qualität bereitstellen.				5					
TA-BO5	Die geeigneten funktionalen Testarten bestimmen, die durchgeführt werden sollen.					1				
TA-BO6	Zu nicht-funktionalen Tests beitragen.						3			
TA-BO7	Zur Fehlerprävention beitragen.							5		
TA-BO8	Die Effizienz des Testprozesses durch den Einsatz von Werkzeugen verbessern.								4	
TA-BO9	Die Anforderungen an Testumgebungen und Testdaten spezifizieren. .									2

Geschäftlicher Nutzen: Advanced Level Test Analyst		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
LO-Nummer	Lernziel (K-Stufe)									
1	Die Aufgaben der Testanalysten im Testprozess - 225 Minuten									
1.1	Testen im Softwareentwicklungslebenszyklus									
TA-1.1.1	... die Beteiligung von Testanalysten an verschiedenen Softwareentwicklungslebenszyklen zusammenfassen (K2)	X								
1.2	Beteiligung an Testaktivitäten									
TA-1.2.1	... die Aufgaben von Testanalysten im Rahmen der Testanalyse zusammenfassen (K2)	X								
TA-1.2.2	... die Aufgaben von Testanalysten im Rahmen des Testentwurfs zusammenfassen (K2)	X								
TA-1.2.3	... die Aufgaben von Testanalysten im Rahmen der Testrealisierung zusammenfassen (K2)	X								
TA-1.2.4	... die Aufgaben von Testanalysten im Rahmen der Testdurchführung zusammenfassen (K2)	X								
1.3	Aufgaben im Zusammenhang mit den Arbeitsprodukten									
TA-1.3.1	... zwischen abstrakten Testfällen und konkreten Testfällen unterscheiden (K2)				X					
TA-1.3.2	... die Qualitätskriterien für Testfälle erklären (K2)				X					
TA-1.3.3	... Beispiele für Anforderungen an die Testumgebung geben (K2)				X					X
TA-1.3.4	... das Testorakelproblem und mögliche Lösungen erklären (K2)			X	X					
TA-1.3.5	... Beispiele für Anforderungen an Testdaten geben (K2)				X					X
TA-1.3.6	... den schlüsselwortgetriebenen Test zur Entwicklung von Testskripten nutzen (K3)	X							X	
TA-1.3.7	... die Werkzeugarten zur Verwaltung von Testmitteln zusammenfassen (K2)								X	

Geschäftlicher Nutzen: Advanced Level Test Analyst		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
2	Die Aufgaben des Testanalysten beim risikobasierten Testen – 90 Minuten									
2.1	Risikoanalyse									
TA-2.1.1	... den Beitrag von Testanalysten zur Risikoanalyse des Produktrisikos zusammenfassen (K2)		X							
2.2	Risikosteuerung									
TA-2.2.1	... die Auswirkungen von Änderungen analysieren, um den Testumfang von Regressionstests zu bestimmen (K4)		X						X	
3	Testanalyse und Testentwurf – 615 Minuten									
3.1	Datenbasierte Testverfahren									
TA-3.1.1	... den Wertebereichstest anwenden (K3)			X						
TA-3.1.2	... den kombinatorischen Test anwenden (K3)			X						
TA-3.1.3	... die Vorteile und Grenzen von Zufallstests zusammenfassen (K2)			X						
3.2	Verhaltensbasierte Testverfahren									
TA-3.2.1	... den CRUD-Test erläutern (K2)			X						
TA-3.2.2	... den Zustandsübergangstests anwenden (K3)			X						
TA-3.2.3	... den szenariobasierten Test anwenden (K3)			X						
3.3	Regelbasierte Testverfahren									
TA-3.3.1	... den Entscheidungstabellentest anwenden (K3)			X						
TA-3.3.2	... den metamorphen Test anwenden (K3)			X						
3.4	Erfahrungsbasiertes Testen									

Geschäftlicher Nutzen: Advanced Level Test Analyst		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
TA-3.4.1	... Test-Chartas für den sitzungsbasierten Test vorbereiten (K3)			X						
TA-3.4.2	... Checklisten für den erfahrungsbasierten Test vorbereiten (K3)			X						
TA-3.4.3	... Beispiele für die Vorteile und Grenzen von Massentests geben (K2)			X						
3.5	Anwenden der am betsen geeigneten Testverfahren									
TA-3.5.1	... geeignete Testverfahren zur Minderung von Produktrisiken in einer bestimmten Situation auswählen (K4)		X	X						
TA-3.5.2	... die Vorteile und Risiken der Automatisierung des Testentwurfs erläutern (K2)								X	
4	Testen von Qualitätsmerkmalen – 60 Minuten									
4.1	Funktionaler Test									
TA-4.1.1	... zwischen dem Testen der funktionalen Korrektheit, der funktionalen Angemessenheit und der funktionalen Vollständigkeit unterscheiden (K2)					X				
4.2	Gebrauchstauglichkeitstest									
TA-4.2.1	... erläutern, wie Testanalysten zum Gebrauchstauglichkeitstest beitragen (K2)						X			
4.3	Flexibilitätstest									
TA-4.3.1	... erläutern, wie Testanalysten zum Anpassbarkeitstest und Installierbarkeitstest beitragen (K2)						X			
4.4	Kompatibilitätstest									
TA-4.4.1	... erläutern, wie Testanalysten zum Interoperabilitätstest beitragen (K2)						X			
5	Fehlerprävention in der Software – 225 Minuten									
5.1	Praktiken zur Fehlerprävention									

Geschäftlicher Nutzen: Advanced Level Test Analyst		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
TA-5.1.1	... erklären, wie Testanalysten zur Fehlerprävention beitragen können (K2)							X		
5.2	Unterstützung der Fehlereindämmung innerhalb der Phase									
TA-5.2.1	... ein Modell des Testobjekts verwenden, um Fehlerzustände in einer Spezifikation zu erkennen. (K3)							X		
TA-5.2.2	... Reviewverfahren auf ein Dokument der Testbasis anwenden, um Fehlerzustände zu finden (K3)							X		
5.3	Verminderung des Wiederauftretens von Fehlerzuständen									
TA-5.3.1	... Testergebnisse analysieren, um mögliche Verbesserungen bei der Erkennung von Fehlerzuständen zu identifizieren (K4)							X		
TA-5.3.2	... erläutern, wie die Klassifizierung von Fehlerzuständen die Grundursachenanalyse unterstützt (K2)							X		

9 Anhang C – Versionshinweise

Die vorliegende Version 4.0 ist eine umfassende Aktualisierung des ISTQB®-Lehrplans Advanced Level Test Analyst. Aus diesem Grund gibt es keine einzelnen Änderungshinweise pro Kapitel und Abschnitt. Im Folgenden wird jedoch eine Zusammenfassung der wichtigsten Änderungen bereitgestellt. Darüber hinaus bietet ISTQB® ein separates Release-Notes-Dokument in Englisch mit einer detaillierten Verfolgbarkeit zwischen den Lernzielen der Versionen v3.1.2 und v4.0 des Lehrplans.

In dieser Hauptversion wurden die folgenden Änderungen vorgenommen:

Struktur des Lehrplans

Alle Lernziele wurden überarbeitet, um sie atomar zu machen und eine Eins-zu-eins-Beziehung zwischen Lernzielen zu Inhalten zu schaffen. Dadurch wird vermieden, dass es Inhalte ohne ein entsprechendes Lernziel gibt. Jedes Lernziel wird entweder in einem Unterabschnitt mit der gleichen Nummer (z. B. das Lernziel TA-1.2.1 im Unterabschnitt 1.2.1) oder, wenn der übergeordnete Abschnitt nicht untergliedert ist, in diesem beschrieben (z. B. das Lernziel TA-1.1.1 in Abschnitt 1.1). Das Hauptziel dieser Überarbeitung der Lernziele war es, diese Version leichter lesbar, verständlich, erlernbar und übersetzbar zu machen, mit Schwerpunkt auf der Erhöhung des praktischen Nutzens und der Ausgewogenheit zwischen Wissen und Fähigkeiten.

Es gibt 36 Lernziele in Version 4.0 gegenüber 31 in Version 3.1.2. Mehrere K4-Lernziele wurden auf K2 oder K3 reduziert. Bei den Lernzielen, die sich auf Testverfahren beziehen, war das dadurch motiviert, dass der Schwerpunkt auf der Anwendung der Testverfahren liegen sollte statt auf der Analyse der Testbasis im Hinblick auf den Testentwurf. Einige Lernziele wurden zu einem einzigen Lernziel zusammengefasst. Die nachstehende Tabelle enthält eine Gegenüberstellung der Anzahl der Lernziele nach kognitiver Wissensstufe und die Mindest-Trainingszeiten in den beiden Versionen.

Lehrplanversion	#K2-Lernziele (Zeit)	#K3-Lernziele (Zeit)	#K4-Lernziele (Zeit)	#Gesamt (Zeit)
aktuell (4.0)	22 (330 Min.)	11 (660 Min.)	3 (225 Min.)	36 (1215 Min.)
vorher (3.1.2)	16 (240 Min.)	5 (300 Min.)	10 (750 Min.)	31 (1290 Min.)

Klassifizierung von Testverfahren

Die Struktur des Kapitels 3 spiegelt eine detailliertere Klassifizierung der Testverfahren im Vergleich zu v3.1.2 wider. Black-Box-Testverfahren werden nun je nach Art des zugrunde liegenden Testobjektmodells in datenbasierte, verhaltensbasierte und regelbasierte Testverfahren eingeteilt.

Erweiterung des Kapitels 5

Das alte Kapitel 5, das ausschließlich Reviews gewidmet war, wurde erweitert, um andere wesentliche Formen der Fehlerprävention zu erörtern, die von Testanalysten angewandt werden, wie z. B. die Verwendung von Modellen zur Fehlerfindung in Spezifikationen, die Analyse von Testergebnissen zur Verbesserung der Fehlerfindung und die Verwendung der Fehlerklassifizierung zur Unterstützung der Grundursachenanalyse.

Änderungen der Lernziele

- Kapitel 1 wurde umstrukturiert in einen Abschnitt über Testaktivitäten und einen über Testmittel.
- Das Thema über abstrakte Testfälle und konkrete Testfälle (altes TA-1.4.2) wurde von K4 auf K2 (TA-1.3.1) herabgestuft.

- Das alte K3-Lernziel TA-2.1.1 über risikobasiertes Testen wurde in zwei aufgeteilt, TA-2.1.1 (K2) zur Risikoanalyse und TA-2.2.1 (K4) zur Risikosteuerung.
- Äquivalenzklassenbildung (altes K4-Lernziel TA-3.2.1) und Grenzwertanalyse (altes K4-Lernziel TA-3.2.2) wurden durch das allgemeinere K3-Lernziel TA-3.1.1 über den Wertebereichstest ersetzt.
- Paarweises Testen (altes K4-Lernziel TA-3.2.6) und Klassifikationsbaumverfahren (altes K2-Lernziel TA-3.2.5) wurden zu einem allgemeineren K3-Lernziel TA-3.1.2 über den kombinatorischen Test zusammengeführt.
- Zustandsübergangstest (altes K4-Lernziel TA-3.2.4) wurde durch das K3-Lernziel TA-3.2.2 ersetzt, das sich auf N-Switch-Überdeckung und Rundreise-Überdeckung konzentriert. Redundanzen mit CTFL wurden entfernt.
- Anwendungsfallbasierter Test (altes K4-Lernziel TA-3.2.7) wurde durch das allgemeinere K3-Lernziel TA-3.2.3 über den szenariobasierten Test ersetzt.
- Entscheidungstabellentest (altes K4-Lernziel TA-3.2.3) wurde zu K3 (TA-3.3.1) herabgestuft.
- Exploratives Testen (altes K3-Lernziel TA-3.3.2) wurde durch zwei separate Lernziele ersetzt: über die Erstellung von Test-Chartas (K3-Lernziel TA-3.4.1) und über die Erstellung von Checklisten zur Unterstützung erfahrungsbasierter Tests (K3-Lernziel TA-3.4.2). Redundanzen mit dem aktuellen CTFL v4.0 wurden entfernt.
- Vier Lernziele, die sich auf den Vergleich von Testverfahren und die Anwendung des am besten geeigneten Verfahrens bezogen haben (altes K4-Lernziel TA-3.2.8 und die K2-Lernziele TA-3.3.3, TA-3.4.1 und TA-3.3.1), wurden zu einem K4-Lernziel TA-3.5.1 zusammengefasst.
- Vier Lernziele zum funktionalen Test (alte K2-Lernziele TA-4.2.1, TA-4.2.2, TA-4.2.3 und das K4-Lernziel TA-4.2.7) wurden zu einem K2-Lernziel TA-4.1.1 zusammengefasst. Das Thema wurde vereinfacht, da der funktionale Test bereits weitgehend im Rahmen der Testverfahren in Kapitel 3 beschrieben ist.
- Einige Themen aus Kapitel 6 (Testwerkzeuge und Testautomatisierung) wurden in den Abschnitt 1.3 verschoben und stärker auf praktische Fragen konzentriert. Das alte K3-Lernziel TA-6.2.1 "Für ein bestimmtes Szenario die Aktivitäten des Test Analysten in einem schlüsselwortgetriebenen Testautomatisierungsprojekt bestimmen" wurde leicht geändert in das K3-Lernziel TA-1.3.6 "den schlüsselwortgetriebenen Test zur Entwicklung von Testskripten nutzen". Das alte K2-Lernziel TA-6.3.1 "Die Verwendung von verschiedenen Arten von Testwerkzeugen erläutern, die beim Testentwurf, der Testdatenvorbereitung und bei der Testdurchführung eingesetzt werden" wurde leicht geändert in das K2-Lernziel TA-1.3.7 "die Werkzeugarten zur Verwaltung von Testmitteln zusammenfassen".
- Zwei ähnliche Lernziele über Reviews (alte K3-Lernziele TA-5.2.1 und TA-5.2.2) wurden zu einem K3-Lernziel TA-5.2.2 zusammengefasst.

Neue Themen

- Qualitätskriterien für Testfälle (K2-Lernziel TA-1.3.2)
- Anforderungen an die Testumgebung (K2-Lernziel TA-1.3.3)
- Bestimmung von Testorakeln (K2-Lernziel TA-1.3.4)
- Anforderungen an die Testdaten (K2-Lernziel TA-1.3.5)

- Zufallstest (K2-Lernziel TA-3.1.3)
- CRUD-Test (K2-Lernziel TA-3.2.1)
- Metamorpher Test (K3-Lernziel TA-3.3.2)
- Massentest (K2-Lernziel TA-3.4.3)
- Vorteile und Risiken der Automatisierung des Testentwurfs (K2-Lernziel TA-3.5.2)
- Beiträge des Testanalysten zur Fehlerprävention (K2-Lernziel TA-5.1.1)
- Verwendung von Modellen zur Fehlerfindung in Spezifikationen (K3-Lernziel TA-5.2.1)
- Analyse von Testergebnissen zur Verbesserung der Fehlerfindung (K4-Lernziel TA-5.3.1)
- Unterstützung der Grundursachenanalyse durch Fehlerklassifikation (K2-Lernziel TA-5.3.2)

Aktualisierung von Normen

Änderungen in den neuesten Versionen der internationalen Normen für Softwarequalität (*ISO/IEC 25010*, 2023) und Testverfahren (*ISO/IEC/IEEE 29119-4*, 2021) haben zu einer Anpassung der entsprechenden Inhalte des Lehrplans geführt.

10 Anhang D – Liste der Abkürzungen

Abkürzung	Bedeutung
BPMN	Business Process Model and Notation (Geschäftsprozessmodell und Notation)
CRUD	create, read, update and delete (erstellen, lesen, aktualisieren und löschen)
CSV	comma-separated values (Komma-getrennte Werte)
DDP	defect detection percentage (Fehlerfindungsanteil)
DRE	defect removal efficiency (Wirksamkeit der Fehlerbehebung)
DSGVO	Datenschutz-Grundverordnung
JSON	JavaScript Object Notation
KI	Künstliche Intelligenz
MBT	Modellbasiertes Testen
MR	Metamorphe Relation
MT	Metamorpher Test
PCE	phase containment effectiveness (Effektivität der Fehlereindämmung innerhalb einer Phase)
RCA	root cause analysis (Grundursachenanalyse)
SDLC	software development lifecylce (Softwareentwicklungslebenszyklus)
TA	Testanalyst
TTA	technischer Testanalyst
UML	Unified Modeling Language
XML	Extensible Markup Language

11 Anhang E – Bereichsspezifische Begriffe

Begriff	Definition
CRUD-Matrix	Eine Matrix, die die Aktionstypen der Funktionen auf die Entitäten innerhalb eines Systems angibt. *!--MFWCRansOriginalSehrschwierigzuverstehen. EsgehtumOperationsartenderFunktionenenundnichtactiontypesMH:CRgestellt.--*
Datensemantik	Die Bedeutung und Interpretation von Daten.
5-Why-Methode	Eine iterative Fragetechnik, die zur Ermittlung der Grundursache eines Fehlerzustands oder Problems verwendet wird, indem die Frage "Warum?" fünfmal wiederholt wird, wobei jedes Mal das aktuelle "Warum" auf die Antwort des vorherigen "Warum" ausgerichtet wird.
Pareto-Analyse	Ein Ansatz zur Ermittlung von Problembereichen oder Aufgaben, die den größten Nutzen bringen.
User-Journey-Karte	Dokumentation zur Visualisierung des Benutzererlebnisses, die die Schritte aufzeigt, die ein Benutzer in einem Prozess durchläuft, um ein Ziel zu erreichen.
Benutzerstudie	Eine Disziplin, die sich mit den Bedürfnissen und Denkprozessen von Benutzern befasst, indem sie untersucht, wie sie Aufgaben ausführen, beobachtet, wie sie mit einer Komponente oder einem System interagieren, oder indem sie Daten analysiert und interpretiert.

12 Anhang F – Softwarequalitätsmodell

Die folgende Tabelle zeigt die Qualitätsmerkmale des ISO/IEC-25010-Produktqualitätsmodells (*ISO/IEC 25010*, 2023). Sie gibt an, welche Merkmale/Untermerkmale in diesem Lehrplan (Test Analyst (TA)) behandelt werden und welche in anderen ISTQB®-Lehrplänen (Technical Test Analyst (TTA), Performance Testing (PT), Usability Testing (UT), Automotive Software Tester (AuT), und Security Tester (SEC)) behandelt werden. Wenn mehrere Lehrpläne ein Qualitätsmerkmal überdecken, wird derjenige zuerst aufgeführt, der es am ausführlichsten erläutert. Die Tabelle vergleicht auch das aktuelle Modell der ISO/IEC 25010 mit der Version 2011, die in der vorherigen Version dieses Lehrplans verwendet wurde.

ISO/IEC 25010:2023 (aktuell)	**ISO/IEC 25010:2011 (früher)**	**Hinweise**	**ISTQB®-Lehrpläne**
Funktionale Eignung	**Funktionale Eignung**		**TA**
Funktionale Vollständigkeit	Funktionale Vollständigkeit		
Funktionale Korrektheit	Funktionale Korrektheit		
Funktionale Angemessenheit	Funktionale Angemessenheit		
Performanz	**Performanz**		PT, TTA
Zeitverhalten	Zeitverhalten		
Ressourcennutzung	Ressourcennutzung		
Kapazität	Kapazität		
Kompatibilität	**Kompatibilität**		**TA**, TTA
Koexistenz	Koexistenz		TTA
Interoperabilität	Interoperabilität		**TA**
Interaktionsfähigkeit	**Gebrauchstauglichkeit**	**Umbenannt**	UT, **TA**
Erkennbare Angemessenheit	Erkennbare Angemessenheit		
Erlernbarkeit	Erlernbarkeit		
Operabilität	Operabilität		
Benutzerfehlerschutz	Benutzerfehlerschutz		
Benutzerbindung	Ästhetik der Benutzungsschnittstelle	Umbenannt	
Inklusivität	Zugänglichkeit	Aufgeteilt und umbenannt	
Benutzerunterstützung			
Selbsterklärungsfähigkeit		Neu	

ISO/IEC 25010:2023 (aktuell)	ISO/IEC 25010:2011 (früher)	Hinweise	ISTQB®-Lehrpläne
Zuverlässigkeit	**Zuverlässigkeit**		TTA
Fehlerfreiheit	Reife	Umbenannt	
Verfügbarkeit	Verfügbarkeit		
Fehlertoleranz	Fehlertoleranz		
Wiederherstellbarkeit	Wiederherstellbarkeit		
Sicherheit (security)	**IT-Sicherheit**	in Deutsch umbenannt	SEC, TTA
Vertraulichkeit	Vertraulichkeit		
Integrität	Integrität		
Nichtabstreitbarkeit	Nichtabstreitbarkeit		
Zurechenbarkeit	Zurechenbarkeit		
Authentizität	Authentifizierung		
Widerstandsfähigkeit		Neu	
Wartbarkeit	**Wartbarkeit**		TTA
Modularität	Modularität		
Wiederverwendbarkeit	Wiederverwendbarkeit		
Analysierbarkeit	Analysierbarkeit		
Modifizierbarkeit	Modifizierbarkeit		
Testbarkeit	Testbarkeit		
Flexibilität	**Übertragbarkeit**	**Umbenannt**	TTA, **TA**, PT
Anpassbarkeit	Anpassbarkeit		**TA**, TTA
Skalierbarkeit		Neu	PT
Installierbarkeit	Installierbarkeit		**TA**, TTA
Austauschbarkeit	Austauschbarkeit		TTA
Sicherheit (safety)		**Neu**	AuT
betriebliche Einschränkung		Neu	
Risikoidentifizierung		Neu	
Fail-Safe		Neu	
Gefahrenwarnung		Neu	
sichere Integration		Neu	

13 Anhang G – Markenzeichen

ISTQB® ist eine eingetragene Marke des International Software Testing Qualifications Board.

UML® ist ein eingetragenes Warenzeichen der Object Management Group (OMG).

BPMN™ ist eine Marke der Object Management Group (OMG).

14 Index