IN DIESEM KAPITEL

- Von Menschen und ihren Softwareproblemen
- Mehr als nur Fehler finden
- Schnelldurchlauf durch das Testen von Software
- Unterstützung durch Werkzeuge

Kapitel 1
Mal eben schnell was testen?!

Vor mehr als 20 Jahren bekam ich von einem Projektleiter den Vorwurf zu hören, dass mein Team und ich »schuld sind, dass das Projekt den Meilenstein nicht halten kann, weil zu viele Fehler gefunden wurden« und die klare Anweisung »nicht mehr so intensiv zu testen, damit das Projekt nicht weiter in Zeitverzug gerät«. Es handelte sich um eines der vielen Projekte, die nie beendet, sondern nach etlichen herausgeworfenen Geldern und Hunderttausenden von Arbeitsstunden gestoppt wurden. Die Firma existierte wenige Jahre darauf nicht mehr.

Damals haben wir getestet, um die Kunden und Endnutzer des Systems vor allzu gravierenden Fehlern in der Software zu bewahren, und in der Hoffnung, dass das Projekt zu einem guten Abschluss kommt. Die Kunden und Geldgeber verlangten einen Nachweis einer ausreichenden Qualität und machten die Bezahlung zu den jeweiligen Meilenstein-Terminen von den Ergebnissen bestimmter Tests abhängig. Letztlich wollten die Kunden sichergestellt wissen, dass das Produkt möglichst keine Fehler enthält, den Anforderungen genügt und seinen Zweck erfüllt.

Warum getestet wird

Neben den Kunden und Endnutzern beziehungsweise den Geldgebern verlangen auch andere ausreichende Qualität, die durch Tests nachgewiesen werden muss. Das Projektteam möchte möglichst frühzeitig wissen, wie gut und vollständig Anforderungen und Design in dem System umgesetzt wurden – ein *Testziel* ist häufig neben der Abdeckung der Anforderungen auch die Überprüfung von Arbeitsergebnissen des Teams. Behörden fordern den Nachweis der Einhaltung von Vorschriften, Verordnungen, Gesetzen oder anderen Regularien sowie von Normen und Standards wie den folgenden:

- ✔ Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie-Informationstechnik-Verordnung – BITV 2.0)

- ✔ Markets in Financial Instruments Directive (MiFID II) für Banken
- ✔ EU-Medizinprodukteverordnung/Medical Device Regulation (MDR) in der Medizintechnik
- ✔ Grundsätze ordnungsgemäßer DV-gestützter Buchführungssysteme (GoBS)
- ✔ ISO 26262 Road vehicles – Functional safety für Automobilbau

Diese Liste könnten Sie sicher mit den für Ihren Bereich geltenden Normen, Standards und Gesetzen noch ergänzen.

Testen beinhaltet die Verifikation der spezifischen Anforderungen an das getestete System und die Validation, ob alles so funktioniert, wie es von den Endanwendern und anderen Stakeholdern erwartet wird. Das Testteam nutzt verschiedenste Techniken, um entsprechende Fehler zu provozieren und somit das Risiko einer ungenügenden Qualität des Systems zu reduzieren.

Beispielsweise können Tester verifizieren, ob die Software gemäß den Anforderungen in der Testbasis korrekterweise ein Skonto von 4 % auf Artikel gewährt, die innerhalb von drei Tagen bezahlt werden. Sie können aber auch validieren, ob zu jedem Bezahlvorgang sichtbar ist, ob ein Skonto in Anspruch genommen wurde. Dass diese Information direkt ermittelbar ist, dass für die Anzeige beispielsweise eine eigene Zeile auf einer Bildschirmseite existiert und dass eindeutig klar ist, dass es sich hier um ein Skonto und nicht um einen anderen Rabatt handelt, sind Erwartungen, die häufig nicht dokumentiert werden. Andererseits kann die Software unbrauchbar sein, wenn diese Informationen nicht sichtbar sind – der Wert für den Kunden ist dann gering, obwohl die Software korrekt rechnet.

Verifikation: Bestätigung durch Bereitstellung eines objektiven Nachweises, dass festgelegte Anforderungen erfüllt worden sind.

Validierung: Bestätigung durch Überprüfung, dass ein Arbeitsergebnis den Bedürfnissen eines Stakeholders entspricht.

Quelle: ISTQB-Glossar

Weitere wichtige Testziele sind die Vermeidung von (Folge-)Fehlern und das Schaffen von Vertrauen in das getestete System. Wie in der Geschichte oben sollen zudem Stakeholder ausreichende Informationen für ihre Entscheidungen erhalten.

Abhängig von dem Kontext, in dem Sie arbeiten und testen, von der Anwendungsdomäne und auch von dem Zeitpunkt der Entwicklung beziehungsweise der Teststufe ändern sich die Testziele. Je später in der Entwicklung des Systems der Test durchgeführt wird und je eher mehr Kunden oder Endanwender in den Test einbezogen werden, desto weniger geht es um das Entdecken von Fehlern und umso mehr geht es darum, Vertrauen in das System aufzubauen. *Beta-Tester* sind häufig ausgewählte Kunden, die ein System vor allen anderen möglichen Kunden testen dürfen und dafür beispielsweise nichts oder einen geringeren Kaufpreis zahlen und andere Vergünstigungen wie einen besseren Support erhalten. Im Gegenzug erhofft sich der Hersteller des Systems positive Berichte in den Medien. Sollten doch noch Fehler vorhanden sein (und das ist meistens der Fall), dann werden diese besser früh bei wohlgesonnenen Kunden gefunden. Zudem wird bei solchen Tests überprüft, wie gut das getestete System zu den Konfigurationen und realen Systemumgebungen des Kunden passt.

Beim Testen von einzelnen Software-Komponenten liegt der Fokus dagegen auf dem Auffinden und Beseitigen von Fehlern und dem Nachweis, dass die Anforderungen umgesetzt wurden und das Design zweckmäßig ist. Genau wie man die vollständige Umsetzung von Anforderungen häufig als Testziel definiert, gibt es hier vergleichbare Überdeckungen der einzelnen Testobjekte.

Das Testen gehört zur *Qualitätssteuerung*, auch *analytische Qualitätssicherung* genannt, und prüft »im Nachhinein«, wie gut die Qualität des Produkts ist. Das gewünschte Qualitätsniveau kann mithilfe der Qualitätssteuerung durch anschließend erfolgte Korrekturen erreicht werden. Neben dem Testen zählen formale Methoden (wie Prüfung von Modellen und Korrektheitsbeweise), das Ermitteln von Kennzahlen, Simulationen und Prototyping zur Qualitätssteuerung.

Der Begriff *Qualitätssicherung* beinhaltet die Erwartung, dass mit der Einhaltung von Prozessen auch ausgereifte Produkte hoher Qualität entstehen. Es geht hierbei um das Vorbeugen mangelhafter Qualität durch die Definition von Dokumentvorlagen, Handlungsanleitungen und der Vorgabe von Prozessschritten. Damit wird versucht, Qualität in ein Produkt zu konstruieren (daher auch *konstruktive Qualitätssicherung* genannt). Testergebnisse können dabei helfen, zu erkennen, wo und mit welchen Methoden diese Prozesse verbessert werden müssen, um Qualität »von Anfang an« zu ermöglichen.

Und wer ist für die Qualität verantwortlich? Mein erster Arbeitgeber verteilte damals einen Gegenstand, auf dessen Vorderseite diese Frage gedruckt war – und auf dessen Rückseite sich ein Spiegel befand.

Ein Unternehmen gilt als umso reifer,

- ✔ je weniger das Testen mit dem bloßen Ziel der Suche nach Fehlern im System durchgeführt wird,
- ✔ je mehr Testen als Möglichkeit gesehen wird, die Effektivität der Maßnahmen der konstruktiven Qualitätssicherung nachzuweisen und
- ✔ je systematischer Grundursachen von Fehlern ermittelt werden, um Fehler dauerhaft zu vermeiden und insgesamt die Prozesse kontinuierlich zu verbessern.

Warum testen Sie?

Was haben Sie an Regularien, Gesetzen oder Normen zu beachten?

Wo sind die Testziele dokumentiert?

Was beim Testen an Fehlern & Co. herauskommt

Haben Sie schon einmal erlebt, dass Entwicklerinnen oder Entwickler empfindlich auf das Wort »Fehler« oder »Bug« reagiert haben? Es kann an Ihrem Auftreten und an Ihrem Tonfall liegen und natürlich an der jeweiligen Person, aber es liegt häufig daran, dass zum Zeitpunkt

des Testens oft noch gar nicht feststeht, ob es sich wirklich um einen Fehler handelt oder um etwas anderes, wie beispielsweise einen Änderungswunsch oder einen Irrtum des Testers.

Rund um das Testen gibt es viele Fehlerbegriffe, die im Rahmen des ISTQB® Certified Tester definiert wurden, um ein einheitliches Verständnis zu schaffen. Alles beginnt mit einer *Fehlhandlung* (*error*), also damit, dass

- ✔ jemand etwas Falsches macht, zum Beispiel eine widersprüchliche oder unrealistische Anforderung definiert oder eine Codezeile fehlerhaft programmiert,
- ✔ oder dass jemand etwas nicht tut, zum Beispiel eine Anforderung nicht im Design berücksichtigt und diese damit dann auch im Code fehlt oder
- ✔ dass jemand etwas zu viel macht, zum Beispiel zusätzliche Funktionalitäten programmiert, die vom Kunden gar nicht gewünscht sind.

Kurz gesagt: Durch eine Fehlhandlung werden Fehlerzustände (Bugs oder Ungeziefer) in den Code eingefügt (siehe Abbildung 1.1).

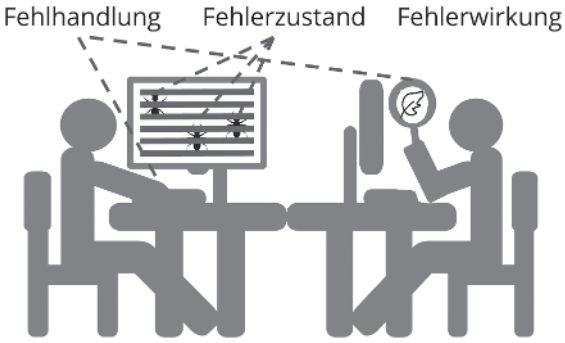

Abbildung 1.1: Fehlhandlung – Fehlerzustand – Fehlerwirkung

Die Fehlhandlung führt zu einem *Fehlerzustand* (*defect*) in einem Arbeitsergebnis, zum Beispiel in den Anforderungen, in einem Designdokument, in Codezeilen oder in einem Testfall. In Abbildung 1.1 handelt es sich sogar um drei Fehlerzustände, wobei der Tester rechts außen offensichtlich nur eine *Fehlerwirkung* (*failure*) (ein angefressenes Blatt) sieht. Manchmal werden keine oder nicht alle Fehlerwirkungen sichtbar. Dies kann aus mehreren Gründen passieren:

- ✔ weil der Fehlerzustand zwar noch im Dokument (beispielsweise den Anforderungen) enthalten ist, aber in einem Folgedokument (beispielsweise dem Design und dem Code) keinen Folge-Fehlerzustand erzeugt hat (vielleicht weil jemand gut aufgepasst hat)
- ✔ weil es keinen Test gab, der dazu geeignet war, die Fehlerwirkung zu provozieren
- ✔ weil ein weiterer Fehlerzustand den anderen Fehlerzustand *maskiert*, die Fehlerwirkung also unsichtbar macht

  Beispielsweise könnte eine fehlerhafte Berechnung maskiert sein, sodass aufgrund eines anderen Fehlerzustands die Ergebnisse gar nicht angezeigt werden. Schlimmstenfalls fällt die fehlende Anzeige dem Tester nicht auf.

Testen Sie, dann finden Sie erst mal Abweichungen zwischen dem von Ihnen erwarteten Soll-Verhalten des Systems und dem tatsächlichen Ist-Verhalten. Das erwartete Soll-Verhalten ist das sogenannte *erwartete Ergebnis*, also das Verhalten des Getesteten, das Sie zum Beispiel auf Basis einer Spezifikation vorhergesagt haben. Außer der Spezifikation können Sie auch ein anderes System für Vorhersagen nutzen, zum Beispiel ein abzulösendes Altsystem oder das Produkt eines Mitbewerbers. Sie nutzen dazu vielleicht auch ein Nutzungshandbuch oder die Ergebnisse aus Interviews mit Experten oder Ihre eigene Erfahrung. Bei diesen Informationsquellen handelt es sich um *Testorakel*. Das denkbar schlechteste Testorakel für den Test von Code ist der Code selbst – die Wahrscheinlichkeit, dennoch Fehlerwirkungen aufzudecken, ist eher gering und sollte daher nicht genutzt werden.

Vermutlich müssen Sie gar nicht lange suchen, um einen »Fehler« in Software zu finden. Vielleicht ist auch, während Sie dieses lesen, gerade wieder etwas ganz aktuell in den Medien. Versuchen Sie doch mal, die Nachricht mithilfe der oben genannten Begriffe nachzuerzählen. Wie viel erfahren Sie von dem tatsächlichen Fehlerzustand in den Medien? Wie ist das in Ihrem Unternehmen?

Wie Testen funktioniert

Wenn Sie schon seit vielen Jahren als Testerin oder Tester arbeiten, kann es natürlich sein, dass Sie Software vorgelegt bekommen, sofort ein paar Eingaben machen und gleich auch ein paar Fehlerwirkungen feststellen, die schon ausreichend sind, um das Ganze wieder an das Entwicklungsteam zurückzugeben. Das Testen war damit erfolgreich, das Projekt wohl weniger.

In den allermeisten Fällen funktioniert das Testen aber geplant und mit verschiedenen einzelnen Aktivitäten:

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

Diese Aktivitäten können überlappend oder auch parallel durchgeführt werden und sind je nach dem jeweiligen Projekt und der jeweiligen Anwendungsdomäne unterschiedlich formal. In den nachfolgenden Abschnitten erhalten Sie eine erste Übersicht über die einzelnen Aktivitäten.

Testplanung

Häufig ist es das Qualitäts- oder Testmanagement, das die *Testplanung* (und Teststeuerung) durchführt. Dabei wird dokumentiert,

- ✔ wer testet (Rollen, Verantwortlichkeiten) sowie
- ✔ was (Testziele, Testobjekte),
- ✔ wann (zeitliche Planung),
- ✔ womit (Testumgebung, Werkzeuge, sonstige Hilfsmittel, räumliche, personelle, finanzielle Ressourcen) und
- ✔ mit welcher Vorgehensweise (strategische Überlegungen, Voraussetzungen für das Testen, Grundlagen des Testens, Methodiken) getestet wird.

In Abbildung 1.2 ist dies die erste Aktivität. Sie wird vom Testmanager ausgeführt.

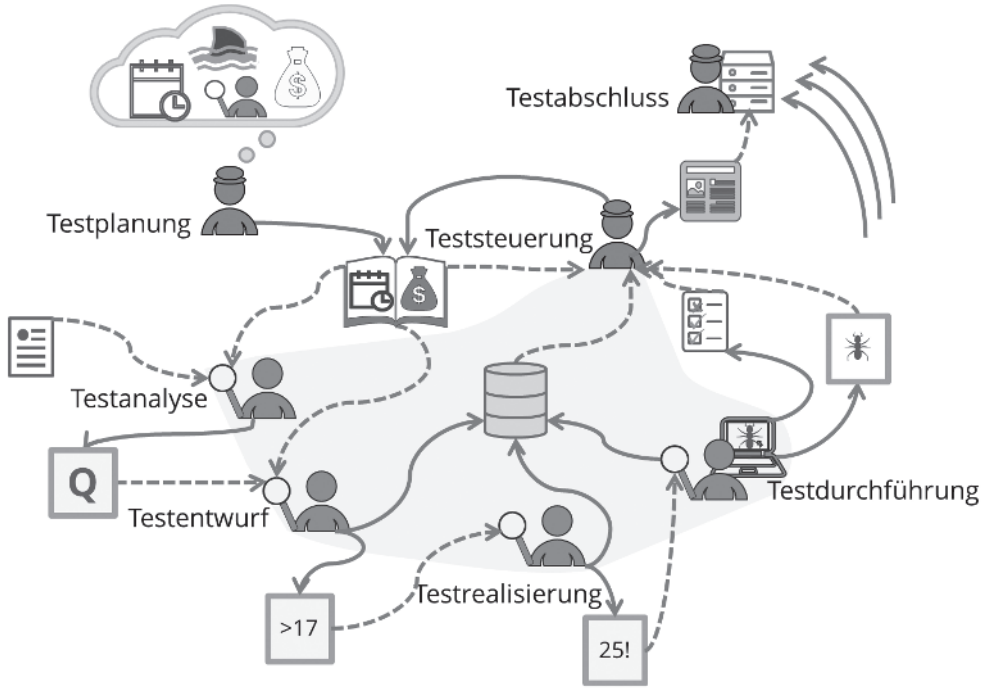

Abbildung 1.2: Aktivitäten im Testprozess

Die Ergebnisse dieser Planungen werden in einem *Testkonzept* zusammengefasst. Ähnlich wie im Projektmanagement enthält es alle Informationen zur Planung des Testprojekts und berücksichtigt dabei Budget-, Personal- und Zeitvorgaben sowie die spezifischen Risiken des Testprojekts. Ein Testkonzept wird durch das Testmanagement in Absprache mit allen Stakeholdern erstellt und von der Entwicklungs- oder Projektleitung unterschrieben. Damit entsteht eine Art Vertrag, in der alle Seiten vereinbaren, was wer zum Testen beiträgt.

Das Testkonzept ist damit wie in Abbildung 1.2 gezeigt das zentrale Dokument, das als Grundlage (gestrichelte Pfeile) für alle anderen Aktivitäten im Testprozess dient. Der Übersichtlichkeit halber sind allerdings nur die wichtigsten dargestellt.

Meistens orientieren sich die Vorgaben der Unternehmen an Standards: häufig noch an dem mittlerweile veralteten IEEE-Standard for Software and System Test Documentation IEEE Std 829™-1998, aber immer öfter an dem aktuellen ISO/IEC/IEEE 29119, der in diesem Buch genutzt wird. In der agilen Softwareentwicklung (siehe Abschnitt »Testen im agilen Kontext« in Kapitel 3) ist dagegen eher ein kurzer Abriss mit den wichtigsten offenen und vereinbarten Punkten zu sehen, oft genug gibt es kein explizites Testplanungsdokument.

Verwechseln Sie das Testkonzept nicht mit dem Testplan. Der *Testplan* (engl. test schedule) enthält die zeitliche Planung und gegebenenfalls noch Zuordnungen von Personen zu Arbeitspaketen und Aufgaben. Der Testausführungsplan beinhaltet wiederum nur die zeitliche Planung der Testdurchführung. Das Testkonzept (engl. test plan) ist dagegen ein umfassendes Dokument, das den Testplan beinhalten kann. Werden häufiger zu ändernde Teile wie beispielsweise Testplan und das Risikomanagement in andere Dokumente ausgelagert, ist das Testkonzept ein sehr stabiles Dokument.

Falls Sie in einem Unternehmen arbeiten, das für das Testkonzept (und natürlich auch alle anderen nachfolgend genannten Dokumente) andere Begriffe benutzt, erstellen Sie sich am besten ein persönliches Glossar mit den passenden »Übersetzungen«. Notieren Sie dann auch, wo es für diese Dokumente Unterschiede zu den hier benutzten Definitionen gibt.

Zur Testplanung gehört die Berücksichtigung vieler einzelner Aspekte, über die Sie sich im Abschnitt »Wer beim Planen scheitert« in Kapitel 12 detailliert informieren können.

Testüberwachung und -steuerung

Wenn eine Testmanagerin den Testprozess beobachtet und mit ihrer ursprünglichen Planung vergleicht, bedeutet es, dass sie eine Reihe von Kennzahlen erfasst oder dass Sie als Tester diese bereitstellen. Solche Kennzahlen sind beispielsweise die Anzahl der geplanten Testfälle und der durchgeführten Testfälle. Alle Kennzahlen werden von ihr ausgewertet und entsprechend zur *Testüberwachung und -steuerung* genutzt. Außerdem wird sie immer wieder genau hinsehen, um drohende Probleme frühzeitig zu erkennen und entsprechend zu handeln (also Risiken zu managen). Tut sie das nicht, dann wird die Testmanagerin von den Problemen gemanagt und kann nur noch Feuerlöscher spielen. Eine gute Testmanagerin wird Sie als Tester optimal einbinden. Sie sind also nicht nur Lieferant von Kennzahlen, sondern werden die Analyseergebnisse mit interpretieren, Sie werden auf Risiken hinweisen und diese aktiv durch Ihre Testaufgaben vermeiden oder zumindest den möglichen Schaden reduzieren, indem Sie beispielsweise Workarounds vorschlagen.

Die Testüberwachung und -steuerung wird, wie in Abbildung 1.2 gezeigt, während des gesamten Testprozesses über alle anderen Aktivitäten hinweg durchgeführt.

Testanalyse

Der Begriff kann leider in die Irre leiten: Nicht der Test wird analysiert, sondern die *Testbasis* – die Aktivität *Testanalyse* bestimmt also, »was zu testen ist«.

Testbasis: Alle Informationen, die als Basis für die Testanalyse und den Testentwurf verwendet werden können.

Quelle: ISTQB-Glossar

Beispiele für die Testbasis sind:

- ✔ Dokumente mit Anforderungsspezifikationen
  - Funktionale und nicht-funktionale Anforderungen, wie beispielsweise die Performanz (mehr dazu in Kapitel 11)
  - Epics und User Storys (siehe Abschnitt »Testen im agilen Kontext« in Kapitel 3)
  - Anwendungsfälle (Use Cases), Geschäftsvorfälle, Business-Modelle
  - Verträge, Normen, Standards, Gesetze und andere Regularien
- ✔ System- oder Softwarearchitekturdokumente
  - Architekturdiagramme/-dokumente
  - Spezifikationen an Entwurf und Komponenten (zum Beispiel Diagramme der Unified Modeling Language (UML) oder Entity-Relationship-Diagramme)
  - Spezifikationen an Schnittstellen
- ✔ Erstellte Module oder Systemteile sowie das ganze System
  - Code
  - Benutzerschnittstellen (Masken, Bedienoberflächen)
  - Datenbank-Metadaten
  - Datenbankabfragen (Queries) und Berichte
  - Schnittstellen zu anderen Systemteilen oder Systemen
- ✔ Marketingdokumente
  - Flyer und andere dokumentierte Marketing-Versprechen
  - (Messe-)Prototypen
- ✔ Projektmanagementergebnisse
  - Projektanträge (also die Versprechen der Projektleitung an das Management)
  - Risikoanalysen (vor allem soweit sie funktionale und nicht-funktionale Aspekte des Produkts betreffen)

Gute Tester nutzen die Testbasis nicht einfach, sondern bewerten sie auch hinsichtlich ihrer Qualität (siehe Kapitel 5).

Das Testkonzept unterstützt – wie in Abbildung 1.2 gezeigt – diese Aufgabe, in dem es unter anderem die Vorgehensweise und die anzuwendenden Testverfahren darstellt. Das Ergebnis der Testanalyse sind die *Testbedingungen* für jedes *Testobjekt*.

Wenn Sie jetzt raten, was eine Testbedingung ist, liegen Sie vermutlich nicht richtig – tut mir leid. Eine Testbedingung ist nämlich nicht die Vorbedingung für den Test (obwohl es ganz ähnlich klingt), sondern beispielsweise eine zu testende Funktion (Beispiele sind »Benutzer anlegen«, »Rechnung bezahlen« oder »Position abfragen«) oder ein zu prüfendes Qualitätsmerkmal (Beispiele sind »Korrektheit« oder »Performanz«). Die Testbedingung wird in Abbildung 1.2 als Dokument mit einem großen Q dargestellt. Nur sehr selten handelt es sich aber um separate Dokumente, meist sind es Auflistungen der zu testenden Merkmale oder auch einfach die ersten Entwürfe der Testfälle. Dann stellen die Kurzbeschreibungen der Testfälle die Testbedingungen dar und müssen später mit der genauen Beschreibung, den erwarteten Ergebnissen und weiteren Details ergänzt werden.

Testbedingung: Ein testbarer Aspekt einer Komponente oder eines Systems, der als Grundlage für das Testen identifiziert wurde.

Testobjekt: Das zu testende Arbeitsergebnis.

Quelle: ISTQB-Glossar

Halten Sie doch mal einen Moment inne und überlegen Sie, welche Dokumente in Ihrem Projekt oder Ihrem Unternehmen als Testbasis geeignet wären. Was würden Sie sich wünschen? Worauf haben Sie (keinen) Zugriff? Was ist zwar vorhanden, aber für Sie noch nicht gut genug? Was fehlt noch?

Überlegen Sie, was Sie daran ändern könnten.

Kennen Sie schon Testtechniken (wie im Kapitel 7 beschrieben)? Was würde Ihnen helfen, um die Testbasis für diese Testtechniken zu verbessern?

Testentwurf

Im Testentwurf nutzen Sie die Testbedingungen je Testobjekt und die detaillierten Testverfahren und erzeugen *abstrakte Testfälle*, das heißt solche, in denen die genutzten Testdaten beschrieben, aber noch nicht genau genannt werden. Beispielsweise notieren Sie im abstrakten Testfall, dass eine Person erwachsen sein soll, der Testwert wird im abstrakten Testfall also »> 17« lauten. Mehr dazu erfahren Sie in Kapitel 6.

Testfall: Eine Menge von Vorbedingungen, Eingaben, Aktionen (falls anwendbar), erwarteten Ergebnissen und Nachbedingungen, die auf Basis von Testbedingungen entwickelt wurden.

abstrakter Testfall: Ein Testfall mit abstrakten Vorbedingungen, Eingabedaten, erwarteten Ergebnissen, Nachbedingungen und Aktionen (falls anwendbar).

Quelle: ISTQB-Glossar

Die erzeugten Testfälle werden priorisiert und zusammengestellt. Dabei werden, wie auch schon bei der Testanalyse, häufig weitere Probleme in der Testbasis entdeckt, die dann frühzeitig behoben werden können.

Abbildung 1.2 zeigt, dass hierbei erste Kennzahlen ermittelt und zumeist in einer Datenbank gesammelt werden wie beispielsweise die Anzahl von Testfällen, sodass das Testmanagement den Fortschritt bei der Erstellung beurteilen kann.

Testrealisierung

Jetzt wird es konkret: Die abstrakten Testfälle werden mit den genauen Daten gefüllt und zu *konkreten Testfällen*. Beispielsweise wählen Sie im konkreten Testfall für einen Testwert »> 17« nun den Wert »25«. Zur *Testrealisierung* gehört auch das Erzeugen von *Testskripten*, die dann automatisiert ausgeführt werden. In Abbildung 1.2 erkennen Sie, dass auch die Testrealisierung Kennzahlen erzeugt, wie beispielsweise der Anteil der automatisierten zu den manuellen Tests.

Ob manuell oder automatisiert: Tests werden entsprechend ihrer Priorisierung und einer sinnvollen zeitlichen Abfolge zusammengestellt. Diese sogenannten *Testsuiten* sorgen vor allem dann für eine reibungslosere Testdurchführung, wenn sie abhängig von der Analyse der jeweiligen Risiken (siehe Kapitel 14) erarbeitet wurden. Wenn es nicht so klappt wie optimistisch von dem Projektmanagement oder dem Testmanagement geplant, werden durch die risikoorientierte Priorisierung die kritischsten Tests zuerst durchgeführt und die weniger kritischen zur Not fallen gelassen.

konkreter Testfall: Ein Testfall mit konkreten Werten für Vorbedingungen, Eingaben, erwartete Ergebnisse und Nachbedingungen sowie eine detaillierte Beschreibung der Aktionen (falls anwendbar).

Testsuite: Eine Menge von Testskripten oder Testabläufen, die in einem bestimmten Testlauf ausgeführt werden sollen.

Quelle: ISTQB-Glossar

Testdurchführung

Je nachdem, in welchem Umfeld Sie testen, werden Sie während der *Testdurchführung* mehr oder weniger protokollieren. Dazu gehören möglicherweise das Erzeugen von Nachweisen (*evidence*), meist Bildschirm-Ausdrucke, die das tatsächliche Testen belegen, und darüber hinaus auch das Erzeugen von Kennzahlen. Außerdem werden Sie beim Testen alle Auffälligkeiten und Abweichungen des erwarteten zum tatsächlichen Ergebnis dokumentieren und alle Informationen notieren, die den Test nachvollziehbar, reproduzierbar und damit den Fehlerzustand auffindbar und behebbar machen.

In Abbildung 1.2 sehen Sie, wie der Testmanager nach der Durchführung der Tests sowohl die Kennzahlen als auch direkt Fehlerberichte und Testprotokolle nutzt, um damit Entscheidungen über die Fortführung der Tests treffen zu können.

Testergebnis: Das Ergebnis und die Konsequenz der Durchführung eines Tests.

erwartetes Ergebnis: Das beobachtbare vorausgesagte Verhalten eines Testelements unter bestimmten Bedingungen, basierend auf seiner Testbasis.

Istergebnis: Im Test beobachtetes/erzeugtes Verhalten einer Komponente oder eines Systems unter festgelegten Bedingungen.

Quelle: ISTQB-Glossar

Testabschluss

Im *Testabschluss* werden die Ergebnisse des Testens an die Stakeholder berichtet und – wie beim Abschluss des gesamten Projekts – alle benötigten Materialien (Testwerkzeuge, Testfälle, Testprotokolle und andere Testdokumente) entweder archiviert oder für die erneute Nutzung in einem nächsten Release aufbereitet. Wie in Abbildung 1.2 zu sehen, ist dies wieder eine Aufgabe der Testmanagerin.

Zudem sollten die gemachten Erfahrungen gesammelt und aufbereitet werden, sodass das nächste Testprojekt oder Software-Release davon profitieren kann.

Wie Werkzeuge das Testen unterstützen

In allen Testphasen können Sie unterschiedlich spezialisierte Werkzeuge nutzen. Immer noch sehr häufig setzen Unternehmen eine Textverarbeitung für das Testkonzept und alle Berichte ein sowie eine Tabellenkalkulation zur Ermittlung von Testfällen und zur Dokumentation der Testdurchführung. Wesentlich weniger aufwendig, weniger fehleranfällig und auch angenehmer in der Nutzung sind spezielle Testmanagementtools, die die Eingabe über unterschiedliche Formular-Ansichten ermöglichen und ausgefeilte Berichtsmöglichkeiten bieten. Solche Tools gibt es als freie Software und auch von namhaften Firmen in den unteren Preisklassen. Zur automatisierten Testdurchführung und für verschiedene Testmethoden vor allem nicht-funktionaler Tests gibt es Spezialwerkzeuge. Eine detaillierte Übersicht über Testwerkzeuge finden Sie in Kapitel 18, »Werkzeuge des Testens«.