



Frank Witte

Strategie, Planung und Organisation von Testprozessen

Basis für erfolgreiche
Projektentwicklung im Softwaretest

EBOOK INSIDE



Springer Vieweg

Strategie, Planung und Organisation von Testprozessen

Frank Witte

Strategie, Planung und Organisation von Testprozessen

Basis für erfolgreiche Projektabwicklung
im Softwaretest



Springer Vieweg

Frank Witte
Landshut, Deutschland

ISBN 978-3-658-31227-5 ISBN 978-3-658-31228-2 (eBook)
<https://doi.org/10.1007/978-3-658-31228-2>

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Springer Vieweg

© Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2020

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von allgemein beschreibenden Bezeichnungen, Marken, Unternehmensnamen etc. in diesem Werk bedeutet nicht, dass diese frei durch jedermann benutzt werden dürfen. Die Berechtigung zur Benutzung unterliegt, auch ohne gesonderten Hinweis hierzu, den Regeln des Markenrechts. Die Rechte des jeweiligen Zeicheninhabers sind zu beachten.

Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag, noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen. Der Verlag bleibt im Hinblick auf geografische Zuordnungen und Gebietsbezeichnungen in veröffentlichten Karten und Institutionsadressen neutral.

Planung: Petra Steinmüller

Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer Fachmedien Wiesbaden GmbH und ist ein Teil von Springer Nature.

Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Vorwort

Mit meinem ersten Buch „Softwaretest und Testmanagement“ hatte ich mich mit einem Überblick über den gesamten Testprozess beschäftigt und mit meinem Buch „Metriken für das Testreporting“ Analysen und geeignete Verfahren vorgestellt, die während und am Ende des Testprozesses zur Steuerung und Kontrolle Anwendung finden. Mit dem vorliegenden Buch möchte ich mich verstärkt mit dem Anfang des Testprozesses beschäftigen, ich beschreibe vor allem eine Testphase, bei der noch nicht unbedingt ein Testobjekt vorhanden ist und bei der es um die Teststrategie, die Testkonzeption und die Planung geht. Die Organisation und Planung von Projekten erstreckt sich grundsätzlich über den gesamten Projektzeitraum, da sie ständig an neue Gegebenheiten angepasst werden muss. Ein Schwerpunkt liegt jedoch in der Konzeptphase zu Anfang des Projektes.

In früheren Zeiten gab es immer noch Unternehmen, die ernsthaft die Notwendigkeit von Software-Systemtests infrage stellten. Das hat sich immerhin flächendeckend im Bewusstsein gewandelt, ob Kleinunternehmen oder Konzern. Jeder professionelle Software-Entwickler lernt bereits im Studium seine Software zu testen. Abhängig davon, ob es sich um ein erstes Release oder ein Update handelt, gilt es die Software in Gänze oder im Rahmen von Regressionstests so weit wie sinnvoll zu testen. Das nicht zu tun, ist so ähnlich, wie wenn ein Chirurg diskutiert, ob das Desinfizieren der Hände vor der OP nun wirklich notwendig sei.

Im Projektmanagement gilt die Regel, dass 25–35 % der voraussichtlichen Projektgesamtdauer für Planungen aufgewendet werden soll. Denn je klarer Ziele, Rahmenbedingungen und Anforderungen beschrieben sind, desto einfacher und schneller lässt sich das Projekt später umsetzen. So wie Projektplanung die Hauptaufgabe des Projektmanagements ist, ist auch eine gut durchdachte und fundierte Testplanung ein Schlüssel für ein erfolgreiches Testprojekt.

Das folgende Buch behandelt Strategie, Planung und Organisation von Testprozessen.

Unter Strategie wurde ursprünglich die Kunst der Heeresführung verstanden. Clausewitz entwickelte den militärischen Begriff politisch und wirtschaftlich weiter. In der Wirtschaft werden die geplanten Verhaltensweisen der Unternehmen zur Erreichung ihrer Ziele als Strategie verstanden. Mit einer Teststrategie werden folglich Rahmenbedingungen

und Vorgehensweisen, aber auch notwendige Einzelaktivitäten für ein Projekt definiert, die zur erfolgreichen Umsetzung des Ziels erforderlich sind.

Planung will die Zukunft gestalten und soll berücksichtigen, mit welchen Mitteln welches Ziel erreicht werden kann. Das bedeutet für den Softwaretest, dass Ziele, Ressourcen und Prozesse definiert und dokumentiert werden müssen. Mit Hilfe der Planung werden Abläufe festgelegt, die im betrieblichen Umfeld strukturiert ablaufen. Die Ergebnisse der Planung werden in der Organisation laufend umgesetzt. Das bedeutet, dass Testprozesse laufend gesteuert werden müssen, also den aktuellen Testfortschritt mit der Planung permanent abgeglichen werden müssen und ggf. Korrekturmaßnahmen einzuleiten sind.

Für die Planung von Testaktivitäten in einem Softwareprojekt wird ein Testkonzept erstellt. Das Testkonzept kann man mit einem Bauplan vergleichen. Die Struktur des Buches orientiert sich daher am Aufbau eines Testkonzeptes nach **Norm IEE829**. Zusätzlich gebe ich Hinweise zur Teststrategie, zur Projekt- und Testplanung und Methoden zur Bewertung. Neue Trends wie das agile oder kognitive Testen wirken sich ebenfalls auf die Testorganisation aus.

Die Definition „IEEE 829 Standard for Software Test Documentation“ ist ein vom IEEE (Institute of Electrical and Electronic Engineers) veröffentlichter Standard, der einen Satz von acht Basis-Dokumenten zur Dokumentation von Softwaretests beschreibt. Die aktuelle Version ist die IEEE 829-2008. Die Struktur nach IEEE 829 kann man anpassen und erweitern, nicht alle Teile sind in gleichem Umfang relevant. Es ist aber sinnvoll, sie als grobe Leitlinie und Kompass zu verwenden.

Das Testkonzept bestimmt Abgrenzung, Vorgehensweise, eingesetzte Mittel und enthält einen Ablaufplan der Testaktivitäten. Das Testkonzept identifiziert die Testobjekte, die zu testenden Features und die Testaufgaben. Es bildet die Grundlage, auf der die Testorganisation und die Testinfrastruktur bereitgestellt und die Tests durchgeführt werden.

In früheren Zeiten wurden Testaktivitäten meist erst zu einer späteren Projektphase gestartet. Dadurch kam die Testabteilung von Anfang an in Terminkonflikte. Inzwischen hat sich auf breiter Front die Erkenntnis durchgesetzt, dass der Testmanager schon von Anfang an im Projekt mit eingebunden sein soll, also von Anfang an den Entwicklungsprozess mit begleitet, die Planungsphase folglich am Beginn des Projektes steht. Daher soll auch das Testkonzept erstellt werden, bevor bzw. während die Systementwicklung in Arbeit ist und die Projektphase der Definition geeigneter Testfälle und die Phase der Testdurchführung noch nicht begonnen haben.

Für die Testplanung gilt sinngemäß dasselbe wie für die Gesamtplanung von Projekten: für eine effiziente und effektive Projektplanung empfiehlt sich eine „**rollierende**“ **Vorgehensweise**. Das bedeutet, dass zu Beginn des Projektes eine grobe Planung für das gesamte Projekt vorgenommen wird, welche jeweils für die einzelne anstehende Projektphase weiter detailliert wird. Vorteile dieser Vorgehensweise sind, dass im Laufe der Untersuchung gewonnene Erkenntnisse in die Planung einfließen können und der Planungsaufwand vertretbar bleibt [ORGH2019].

Im Folgenden möchte ich aber nicht nur die Inhalte des Testkonzeptes lehrbuchartig aufzeigen, sondern aus Erfahrung auf die Fallstricke und Besonderheiten in der Praxis

hinweisen. Bereits bei der Erstellung des Testkonzeptes kann man, bei geübter Betrachtung, im Projektverlauf entstehende Risiken im Vorfeld erkennen und mögliche Gegenmaßnahmen von Anfang an definieren. Man kann die Schwächen im Prozess aufzeigen und die Rahmenbedingungen transparent machen, um bei eventuell auftretenden Problemen von vornherein gerüstet zu sein. Es geht darum, als Testmanager nicht als Getriebener im Reaktionsmodus zu sein, sondern von Anfang an aktiv agieren zu können.

Jedes Testprojekt ist in einem bestimmten Umfeld eingebunden, und ich empfehle jedem Testmanager, sich erst die Rahmenbedingungen und die besonderen Probleme zu vergegenwärtigen:

Da ist zum einen das Thema Personal: Stehen überhaupt genügend Tester zur Verfügung? Sind die Tester grundsätzlich motiviert? Sind gewisse Ressourcen erfahrungsgemäß überlastet? Wie ist die Auslastung im Zeitverlauf? Wenn ein Mitarbeiter mit 40 Arbeitstagen von 1.3. bis 30.4. geplant ist, aber sein Arbeitspaket Voraussetzungen hat, die erst am 15.3. vorhanden sind, und der Mitarbeiter am 15.4. seine Testergebnisse bereits liefern muss, ist er zeitweise massiv überlastet, obwohl die Gesamtverfügbarkeit stimmt. Gibt es (evtl. aus der Vergangenheit) persönliche Differenzen im Team oder Koalitionen? Stehen die Mitarbeiter dem Projekt exklusiv zur Verfügung, oder greifen unterschiedliche Projekte auf die einzelnen Tester zu? Dabei ist zu bedenken, dass jeder Stakeholder mehr die Kommunikation komplizierter macht: Bei nur 10 Beteiligten gibt es rein rechnerisch 45 Kommunikationswege zwischen allen Beteiligten, bei 20 bereits 190. Vor allem der „Flur Funk“ kann in größeren Organisationseinheiten viel zerstören.

Wer wird den Test durchführen, sind die Tester erfahren oder neu im Projekt? Sind sie motiviert oder eher desillusioniert, machen sie vielleicht sogar schlechte Stimmung gegen das Projekt? Gibt es persönliche Differenzen innerhalb der Stakeholder oder Allianzen im Unternehmen, zum Beispiel Entwickler gegen Tester?

Wie steht es um das Projekt, ist es politisch gewollt, hat es hohe Priorität? Hat das Projekt Förderer im Unternehmen, oder kann es sein, dass es aufgrund anderer Prioritäten ausgebremst, geblockt, hintangestellt wird? Gibt es andere Projekte, die eine höhere Priorität haben, Voraussetzung für das anstehende Projekt haben oder vielleicht sogar im Widerspruch dazu stehen?

Gibt es technische Risiken, Neuerungen, unbekannte Nebeneffekte? Eine Umstellung der Server oder der Datenhaltung in der Cloud hat meist nicht direkt mit der zu testenden Software zu tun, aber in vielen Fällen gibt es Auswirkungen. Ähnlich verhält es sich zum Beispiel bei einer Änderung der Datenbanken. Laufen parallel andere Projekte, die einen technischen oder fachlichen Einfluss auf das eigene Projekt haben, oder ist das betrachtete Projekt sogar auf Gedeih und Verderb ihnen ausgeliefert? Welche Voraussetzungen müssen vorhanden sein, um gewisse Aktivitäten überhaupt zu starten bzw. erfolgreich durchführen zu können? Sind andere Projekte vom eigenen Projekt abhängig?

Wie liefen Projekte in der Vergangenheit ab? Gibt es unternehmensspezifische oder branchenspezifische Probleme? Gibt es dazu „Lessons-Learned-Protokolle“?

Ein Testmanager, der von außen (z. B. extern) in ein Projekt kommt, kennt oft nicht genügend die technischen und fachlichen Besonderheiten und die Stärken und Schwächen

der einzelnen Mitarbeiter. Jemand, der schon lange dabei ist, kennt das alles vielleicht, kann aber wiederum wegen Erfahrungen aus früheren Projekten vorbelastet sein. Wie ist die Stellung der Testabteilung im Unternehmen, ist sie stark oder wird der Test eher als notwendiges Übel betrachtet, der ein Schattendasein fristet?

In der Vergangenheit wurde das Testen von Anwendungen vielfach als notwendiges, aber wenig nutzbringendes Anhängsel klassischer Software-Entwicklungsprojekte gesehen. Diese Betrachtungsweise hat sich nicht zuletzt vor dem Hintergrund geändert, dass Testen, über den Lebenszyklus einer Applikation betrachtet, zwischen 25 und 50 % der anfallenden Gesamtkosten in Anspruch nimmt. Auch Software-Qualitätsprobleme können schwerwiegende Konsequenzen nach sich ziehen, wie beispielsweise eine qualitätsbedingte Projektverzögerung, die oftmals die Time-to-Market verlängert. Größte kommerzielle und materielle Schäden können entstehen, wenn beispielsweise eine Buchungsplattform aus softwaretechnischen Gründen ausfällt, wichtige Daten verloren gehen bzw. verfälscht werden oder Sicherheitslücken im System zum Diebstahl von Kunden- oder Kreditkarteninformationen führen. Darüber hinaus steigen durch die immer engere Verzahnung von Business und IT die Anforderungen an Time-to-Market und Qualität, was den Leistungsdruck auf die Testorganisationen erhöht. Zugleich werden Systeme und IT-Landschaften immer komplexer und die Software-Entwicklungs- und -Lebenszyklen schneller. Hinzu kommen erweiterte regulatorische und sicherheitsspezifische Anforderungen. Insofern ist es nur konsequent, dass seit einigen Jahren das Testen von Anwendungen zu den aktuellen Fokusthemen der IT zählt [IODT2019].

Aus meiner Sicht sind vor allem unklare Aufgabenstellungen ein wesentlicher Problempunkt in zahlreichen Testprojekten. Viele der Kommunikationsprobleme entzündeten sich daran, dass man von Anfang an nicht exakt definiert hat, wie das Testobjekt auszusehen hat, wie es abgegrenzt wird und was die Erwartungshaltung an den Softwaretest ist. Anforderungen, die unvollständig, interpretierbar oder widersprüchliche Inhalte mit sich führen, sind erfahrungsgemäß leider eher die Regel als die Ausnahme. Diese Probleme führen dann zu Fehlern bei der Entwicklung und Rückfragen und Fehlern im Testprozess.

Welche Erwartungen werden vom Management an das Projekt gestellt? Sind diese Erwartungen realistisch hinsichtlich Terminsituation, Kosten und Qualität? Wo liegt die Kostenverantwortung? Dabei muss gerade der Testmanager, der die Qualität der Software sichern soll, von Anfang an vor übertrieben optimistischen Erwartungen warnen.

Erst wenn man entsprechende Aussagen mit Zahlen hinterlegt, werden sie auch konkret greifbar.

Für die Bewertung empfehle ich, die vorhandenen Risiken immer in echte EUR umzurechnen. Eine Aussage wie „das Risiko beträgt 750.000,- EUR wegen mangelhafter Testabdeckung“ ist wesentlich eindringlicher als eine Aussage wie „wahrscheinlich können 5 % der geplanten Testfälle nicht termingerecht getestet werden“.

Viele Fragestellungen kann man sich im Vorfeld stellen und an geeigneter Stelle im Testkonzept berücksichtigen. Typische Problemfälle will ich auf den folgenden Kapiteln dabei benennen. Teilweise empfiehlt es sich, entsprechende „Hilfsdokumente“ dafür zu erstellen.

Dieses vorliegende Buch soll dabei helfen, das Testkonzept mit Leben zu füllen.

Manchmal wird das Testkonzept lediglich aus Gründen der Revisionssicherheit erstellt. Manchmal ist es als theoretisches Dokument vorhanden, spiegelt aber nicht den tatsächlichen Zustand der Testaktivitäten wider. Dadurch wird die Chance vertan, den Testprozess mit dem notwendigen Pragmatismus individuell zu gestalten und begründet zu dokumentieren. Ein Testkonzept mit Alibifunktion erfüllt seinen Zweck nicht.

Die dabei freigegebene Version ist ein Plan, mit dem man startet. Es ist normal, dass man im Laufe der Testaktivitäten neue Erkenntnisse gewinnt und dadurch nicht mehr strikt nach dem vorgezeichneten Wege testen kann. Man muss so viel Flexibilität besitzen, pragmatisch vorzugehen und es kann immer gute Gründe geben, vom ursprünglichen Plan abweichen zu müssen. Es ist dann aber auch notwendig, das Testkonzept nachträglich anzupassen, damit das tatsächliche Testvorgehen daraus hervorgeht. Wenn man bei einem neuen Haus auf der Baustelle eine Wand nachträglich einzieht oder eine Tür versetzt, muss auch der Bauplan geändert werden. Genauso verhält es sich mit dem Testkonzept.

Je mehr die einzelnen Mitarbeiter im Team bei offenen Fragen rund um den Softwaretest das Testkonzept heranziehen, je mehr sie sich damit beschäftigen und kritische Anmerkungen geben, desto eher kann auch das gesamte Testprojekt in die richtige Richtung gelenkt werden. Gerade dem Testmanager obliegt hier die Verantwortung, darauf zu achten, dass das Testkonzept bei Bedarf geändert wird, wenn aktuelle Entwicklungen den Plan überholen. Oft erfährt man eher nebenbei, dass bestimmte Annahmen, die im Testkonzept definiert wurden, gar nicht zutreffend sind.

Das Testkonzept ist kein Produkt, das am grünen Tisch entsteht. Ein Testmanager sitzt nicht wie ein musikalisches Genie im stillen Kämmerlein und entwirft dabei sein großes Werk. Ein Testkonzept entsteht schrittweise im Dialog mit Experten und zukünftig Betroffenen im Projekt. Bei kleinen Vorhaben reicht dafür möglicherweise schon wenige Arbeitstage (fairer Weise sei hier erwähnt), dass eine Testkonzeption im Mittel zwischen 5 bis 20 PT Aufwand benötigt, bei besonders umfangreichen Vorhaben (z. B. Multiprojekten oder großen Softwarepaketen) auch deutlich mehr. Um ein Testkonzept zu erarbeiten, muss eine Vielzahl nötiger Fragen mit den richtigen Experten geklärt werden. Sicher sind im Anschluss noch einzelne Details auszuarbeiten, aber das wichtigste sollte angerissen und einmal gemeinsam durchdacht sein. Alternativ kann ein Testmanager, Testkoordinator, Testverantwortlicher (oder wie auch immer man ihn nennen mag) in Einzelinterviews die Fragen klären und alles nach und nach zusammentragen [PERF2017].

In agilen Teams, in denen iterativ gearbeitet wird, müssen nicht alle Fragen vorweg beantwortet sein. Es reicht auch, sie anzureißen und ihre Lösungen nach Prioritäten der Reihe nach anzugehen. So ist nicht einzusehen, warum ein Team von Anfang an genau definieren muss, welche Testartefakte (Testfälle, Testprotokolle, Berichte, ...) es führen will. Wenn aber ein Akzeptanztest durch den Fachbereich zum nächsten Release erforderlich ist, sollte man die Zielgruppe schon jetzt lieber buchen (und sich damit unweigerlich festlegen). Die o. g. Norm in Form einer Taskliste kann die nötigen Klärungen gut visualisieren und dafür sorgen, dass man im Laufe der Iterationen nichts zu spät löst bzw.

entscheidet. Die zunehmende Agilisierung und künftige Trends führen zu neuen Umweltbedingungen, mit denen sich auch das Testkonzept anpassen muss.

Als zentraler Punkt gilt bei allen Ausarbeitungen und Festlegungen: Man braucht die unbedingte Zustimmung der Beteiligten! Nur wenn ein Konsens im Unternehmen besteht, ist es möglich, die Ziele der Qualitätssicherung, die mit Hilfe von Tests umgesetzt werden, überhaupt zu erreichen.

Besonders wichtig ist in diesem Zusammenhang, das Testkonzept einem Review zu unterziehen und den Review Prozess exakt zu definieren. Dafür sind Review Sitzungen mit allen Beteiligten unumgänglich. Ein Testkonzept nur in die Runde zu senden und auf Feedback zu warten, ist nicht zielführend. Nicht jeder Beteiligte kann zu allen Punkten im Testkonzept etwas sagen, wenn man aber nur die Rückmeldungen einholt läuft man Gefahr, dass einzelne Seiten gar nicht angeschaut werden, weil ein eingeladener Reviewer für bestimmte Kapitel jemand anderen für die Beurteilung des Inhalts als Verantwortlichen betrachtet. Papier ist eben geduldig. Gerade bei Reviews habe ich schon oft beobachtet, dass man meint, keine Zeit zu haben und deswegen notwendige Aktivitäten abkürzen oder unterlassen wollen. Man verbraucht dadurch aber am Ende noch mehr Zeit – frei nach dem Motto „wir haben Zeit die Sachen dreimal falsch zu machen aber nicht um sie einmal richtig durchzuführen“.

Ein Review-Zyklus dient also nicht der Fehlerfindung im Konzept allein, sondern vor allem der Zustimmung. Diese muss man sich aktiv einholen. Viele machen hier den Fehler, ein Konzept auf den Tisch zu knallen und dann davon auszugehen, dass sich alle (gefälligst) daran halten. Systemisch liegt dadurch die Verantwortung bei dem Ersteller des Testkonzeptes und er wird sich um alles selbst kümmern müssen, was selten sinnvoll ist [PERF2017].

Jedes eigene Projekt sollte ein eigenes Testkonzept haben, je nach Projektumfang unterschiedlich umfangreich.

Ein **Projekt** ist durch **Norm DIN 69901** folgendermaßen definiert: Es ist ein Vorhaben, das im Wesentlichen durch die Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist: Für das Projekt existiert eine konkrete Zielvorgabe, es ist zeitlich begrenzt, es benötigt spezifische Ressourcen (z. B. finanziell, personell), es existiert eine eigenständige Ablauforganisation, die sich von der Standardorganisation des Unternehmens abgrenzt, es werden nicht routinemäßige Aufgaben behandelt, es besteht aus mindestens 3 Teilnehmern und hat eine Mindestdauer von 4 Wochen.

Ein Projekt hat einen Anfang und ein Ende, im Gegensatz zu einem Prozess, der fortlaufend ist. Ein Projektstrukturplan (PDF) ist in Arbeitspakete gegliedert, in denen die einzelnen Vorgänge, also auszuführende Schritte, beschrieben sind. Häufig wird am Ende eines Arbeitspakts ein Meilenstein gesetzt – ein Zeitpunkt zu dem der Fortschritt des Projektes analysiert und gegebenenfalls den Auftraggebern in einem Meeting vorgestellt wird. Die Überwachung des Ablaufs führt der Projektmanager durch, alle Mitglieder des Projektteams müssen ihm zuarbeiten und sind für die ihnen zugewiesenen Vorgänge verantwortlich. Auch das Krisenmanagement und das Erstellen einer Dokumentation fallen in die Aufgaben des Managers [KARB2019].

Der Testmanager ist für das Projektmanagement aller Teststufen verantwortlich, koordiniert und überwacht die Testaktivitäten. Eine gute Zusammenarbeit und enge regelmäßige Abstimmung mit dem Projektleiter sollte deshalb gegeben sein, sonst wird es erfahrungsgemäß schwierig bis unmöglich gute Fortschritte und Ergebnisse beim Testprozess zu erreichen. Differenzen mit dem Projektleiter sollten gar nicht erst entstehen bzw. müssen sofort ausgeräumt werden. Wenn Projektleiter und Testmanager nicht an einem Strang ziehen, werden das andere Stakeholder schnell bemerken, die Produktivität im Projekt wird darunter leiden oder sogar das gesamte Projekt scheitern.

Man kann aber zusätzlich ein generisches Testkonzept für das gesamte Unternehmen bzw. die gesamte zu betrachtende Organisationseinheit erstellen, vor allem wenn es sich um wiederkehrende Tätigkeiten handelt.

Die IEEE 829 dient dabei als ein Leitfaden, es ist aber notwendig, das Testkonzept an die betrieblichen Gegebenheiten und individuellen Besonderheiten anzupassen. Ich habe bereits eine Reihe von Testkonzepten erstellt, die sich zwar in der Grobstruktur ähneln aber im Schwerpunkt je nach Branche, Unternehmen und Testobjekt unterscheiden.

Erfahrungsgemäß kommt es dabei zu zahlreichen Fragen, die sich während der Erstellung des Testkonzeptes bereits klären lassen. Je stärker man die Stakeholder dabei schon in der Planungsphase einbezieht, umso mehr Aufmerksamkeit und Vertrauen werden bereits im Vorfeld erzeugt, bevor es mit der eigentlichen Testdurchführung losgeht.

Bei den offiziellen Vorgaben und Normen geht man in der Regel von einem optimalen Arbeitsumfeld aus. Dies ist oftmals nicht gegeben, sondern man hat von Anfang an Rahmenbedingungen, die das Projekt negativ beeinflussen, wie z. B. unklare Aufgabenstellungen, Terminverzug, mangelndes Know-how im Testteam; dafür sind meist Kommunikationsprobleme verantwortlich. Die Beispiele in vielen Lehrbüchern gehen von idealen Umgebungsvoraussetzungen aus, die in der Praxis so nicht gegeben sind. Ich werde im Folgenden im Detail darauf eingehen, wie man gerade bei schwierigen Randbedingungen den Softwaretest doch noch zum Erfolg führen bzw. das Schlimmste verhindern kann. Viele Projekte erreichen ihr Ziel nicht in Time and Budget, oft, weil die Planung von Anfang an zu optimistisch war und/oder zu wenig Parameter bei der Betrachtung eingeflossen sind. Man kann aber mit geeigneten Methoden darauf hinwirken, dass Terminverzug, Kostenüberschreitung oder Qualitätsmängel immerhin deutlich minimiert werden. Man muss dazu den Königsweg zwischen einem geplanten, strukturierten Vorgehen und der notwendigen Flexibilität beschreiten. Ein pragmatischer Umgang mit den vorhandenen Realitäten ist genauso wichtig wie die Einhaltung von Regeln und Normen.

Der Umfang eines Testkonzeptes hängt sicherlich vom Umfang des Projektes ab. Aber selbst bei kleinen Projekten muss man einen Plan haben, wie man beim Testen vorgeht und die entscheidenden Rahmendaten dokumentieren.

Falls Informationen bereits existieren oder an anderer Stelle gepflegt werden, ist zu vermeiden, sie redundant zu beschreiben. Eine Stakeholder Liste mit Namen, Abteilungsbezeichnungen und Telefonnummern muss bei einer betrieblichen Umorganisation jeweils aktualisiert werden. Wenn man jedoch nur die Rolle und den Namen beschreibt und dabei auf ein Dokument verweist, das dafür ohnehin gepflegt wird, kann das gerade bei großen

Projekten besser sein. Das gilt auch für Terminpläne. Eine redundante Datenpflege beinhaltet immer das Risiko, dass unterschiedliche und damit widersprüchliche Informationen vorgehalten werden, auf denen aufgebaut wird.

Es empfiehlt sich für eine Organisationseinheit, die häufiger Tests durchführen soll, ein Template für das Testkonzept vorzubereiten. Wenn einzelne Teile nicht relevant sein sollten, kann man das dann innerhalb dieser Vorlage an geeigneter Stelle anmerken. Dadurch wird nichts vergessen, und man bekommt bei der Erstellung des Testkonzeptes bereits Hinweise, an was man denken sollte.

Man kann sich an ähnlich gelagerten Projekten aus der Vergangenheit orientieren und Schwachstellen, die damals auftraten, bereits zu Projektbeginn untersuchen: wenn man schon weiß, dass die Bereitstellung der Testumgebung bei früheren Releases mehrfach verzögert war oder man schon ahnt, dass es bei der Integration bestimmter Komponenten immer wieder zu Fehlern kommt, kann man bereits während der Planungsphase ausreichend Puffer einplanen und entsprechende Risiken schon zu Projektbeginn aufzeigen.

Wenn im Folgenden von Testmanagern, Testkoordinatoren, Testern usw. die Rede ist, meine ich immer auch die weibliche Form. Die männliche Form wird nur aufgrund der Lesbarkeit verwendet.

Zu bedenken ist, dass eine gute Testorganisation und ein durchdachtes Testkonzept Zeit benötigen. Die Schätzungen für den dafür benötigten Aufwand sollen sich an Erfahrungsschatz, Projektumfang und dem Einsatzzweck der Anwendung orientieren. Hier gilt es für den Testmanager, hinter seinen Zahlen zu stehen, diesen Aktivitäten die nötige Wichtigkeit im Unternehmen zu verschaffen und manchen übertriebenen Optimismus zu bremsen.

Literatur

- [KARB2019]: <https://karrierebibel.de/projektarbeit>, zugegriffen am 09.05.2020
- [PERF2017]: <https://www.informatik-aktuell.de/entwicklung/methoden/das-perfekte-testkonzept-in-6-schritten.html>, zugegriffen am 09.05.2020
- [IODT2019]: <https://www.iot-design.de/allgemein/transformation-von-testorganisationen/>, zugegriffen am 09.05.2020
- [ORGH2019]: https://www.orghandbuch.de/OHB/DE/Organisationshandbuch/2_Vorgehensmodell/21_Projektvorbereitung/212_Projektplanung/projektplanung-node.html, zugegriffen am 09.05.2020

Inhaltsverzeichnis

- 1 Testdokumente nach IEEE 829** 1
 - 1.1 Normen 1
 - 1.2 Testkonzept 2
 - 1.3 Testspezifikation („test specification“) 3
 - 1.4 Testbericht („test reporting“) 4
 - 1.5 Integritätslevel 6
 - Literatur. 8
- 2 Teststrategie** 9
 - 2.1 Merkmale der Teststrategie 9
 - 2.2 Integrationsstrategie 13
 - 2.3 Testspezifikationen 13
 - 2.4 Einsatz von Tools 14
 - 2.5 Fragen zur Umsetzung der Teststrategie 15
 - 2.6 Balanced Scorecard 16
 - Literatur. 19
- 3 Testziele** 21
 - 3.1 Bedeutung von Testzielen 21
 - 3.2 Testkenngrößen 23
 - 3.3 Test-Reporting 24
 - Literatur. 27
- 4 Testplanung** 29
 - 4.1 Inhalte der Testplanung 29
 - 4.2 Teilpläne für IT-Projekte 30
 - 4.3 Testvoraussetzungen 32
 - 4.4 Rahmenbedingungen der Testplanung 33
 - 4.5 Testwerkzeuge 34
 - 4.6 Testaufwände und Testautomatisierung 34
 - Literatur. 35

5	Bezeichnung des Testkonzeptes und Einleitung	37
5.1	Bezeichnung und Ablage des Testkonzeptes	37
5.2	Testkonzepte für einen längeren Zeitraum	38
5.3	Formale Grundlagen	39
5.4	Einleitung Testkonzept	40
5.5	Im Testkonzept referenzierte Dokumente	42
5.6	Testrahmen mit Testvoraussetzungen, Fehlerklassen und Startbedingungen	46
	Literatur	47
6	Testorganisation	49
6.1	Wesentliche Kriterien der Testorganisation	49
6.2	Testpersonal und Testereinsatzplanung	52
6.3	Rechenzeit	53
6.4	Datenbereitstellung	54
6.5	Benötigte Hardware	54
6.6	Projektorganisation	54
6.7	Stakeholder und Rollen in der Projektorganisation	55
6.8	Organisationsstruktur	55
6.9	Projektlenkungsausschuss	55
	Literatur	57
7	Prozessbeschreibung	59
7.1	Testprozess nach ISTQB	59
7.2	Planung und Steuerung	60
7.3	Meetings zu Projektstart	61
7.4	Analyse und Design	63
7.5	Realisierung und Durchführung	63
7.6	Auswertung und Bericht	64
7.7	Testabschluss	64
7.8	Dokumente als Testvoraussetzung	65
	Literatur	66
8	Testobjekte und Testphasen	67
8.1	Testobjekte	67
8.2	Testphasen	68
8.3	Einflussfaktoren für die Ermittlung des Testaufwands	70
	Literatur	72
9	Teststufen	73
9.1	Klassische Teststufen	73
9.2	Exploratives Testen	76
9.3	Teststufen in einem agilen Projekt	77
9.4	Integration der Testaktivitäten mit anderen Projektaktivitäten	81
	Literatur	82

10	Zu testende Leistungsmerkmale	83
10.1	Arten von Systemtests	83
10.2	Spezielle Testverfahren	86
	Literatur.	87
11	Leistungsmerkmale, die nicht getestet werden	89
11.1	Testabdeckung durch Testverfahren.	89
11.2	Nachweise durch andere Methoden.	90
	Literatur.	91
12	Priorisierung von Testfällen	93
12.1	Testumfang.	93
12.2	Priorisierung beim Systemtest und beim Abnahmetest	94
12.3	Priorisierung beim Modultest und Integrationstest.	94
12.4	Priorisierung nach unterschiedlichen Testarten	96
12.5	Risikobasiertes Testen	98
	Literatur.	99
13	Permanente Testorganisation	101
13.1	Wesen der permanenten Testorganisation	101
13.2	Bedingungen für eine permanente Testorganisation	103
13.3	Kritische Punkte für permanente Testorganisationen	104
13.4	Testrichtlinien	104
	Literatur.	105
14	Abnahmekriterien	107
14.1	Merkmale von Abnahmekriterien	107
14.2	Abnahmekriterienkatalog.	108
14.3	Organisation der Abnahme.	110
14.4	Definition geeigneter Akzeptanzkriterien	111
14.5	Eigenschaften von Akzeptanztests.	113
	Literatur.	114
15	Kriterien für Testabbruch und Testfortsetzung	115
15.1	Fälle für Testabbrüche	115
15.2	Wiederanlauf und Testfortsetzung nach Testabbruch	116
16	Testrisiken	119
16.1	Projekterfolg	119
16.2	Bewertung von Risiken	120
16.3	Mögliche Testrisiken	121
	Literatur.	127
17	Testdaten	129
17.1	Testdatenmanagement	129

17.2	Ansätze zur Testdatenerstellung	131
17.3	Werkzeuge zur Testdatenerstellung	133
17.4	Testdatentypen	134
	Literatur	134
18	Testdokumentation	135
18.1	Ziele der Testdokumentation und Integritätslevel	135
18.2	Struktur der Testdokumentation	137
18.3	Testskript	137
18.4	Testfall	138
18.5	Testprotokoll	138
18.6	Zusammenfassung der Testdokumentation	139
18.7	Vernachlässigung der Testdokumentation	139
18.8	Testabschlussbericht	141
18.9	Metriken	141
18.10	Verfügbarkeit des Testsystems	143
	Literatur	145
19	Testaufgaben	147
19.1	Arten von Testaufgaben	147
19.2	Rollenbeschreibungen	148
19.3	Norm IEC 62034	150
19.4	Problemlösungsprozess	151
19.5	Testaufgaben im Regressionstest	152
19.6	Testtools	153
19.7	Aufgaben für die einzelnen Stakeholder	154
	Literatur	154
20	Testumgebung	157
20.1	Notwendigkeit von Testumgebungen	157
20.2	Angemessenheit der Testumgebung	158
20.3	Testumgebungsmanagement	158
20.4	Externe Testumgebungen	160
20.5	Testumgebung in der Cloud	161
	Literatur	164
21	Verantwortlichkeiten, Zuständigkeiten und Kommunikation	165
21.1	Festlegung der Verantwortungsbereiche	165
21.2	Kommunikation im Projekt	166
21.3	Definition von Zuständigkeiten	167
21.4	Kommunikationsmatrix	168
21.5	Auswahl der Informationsempfänger	170
21.6	Definition von Kommunikationszielen	172
21.7	Festlegung von Informationsinhalten	172

21.8	Vorteile der Kommunikationsmatrix	173
21.9	Anforderungen an die Projektkommunikation	173
21.10	RACI-Methode	174
21.11	Häufige Fehlerquellen	176
	Literatur	176
22	Personal, Einarbeitung, Ausbildung	177
22.1	Testpersonal	177
22.2	Konfliktmanagement	178
23	Zeitplan/Arbeitsplan	181
23.1	Ermittlung der Testdauer	181
23.2	Detaillierte Arbeitsplanung	183
23.3	Kostenschätzung	184
23.4	Planung der einzelnen Testphasen	184
23.5	Personalauslastungsplan	187
23.6	Detaillierte Testaufwandsplanung	188
23.7	Schätzung der Testproduktivität mit der COCOMO-II-Methode	190
	Literatur	192
24	Planungsrisiken und Unvorhersehbares	193
24.1	Unvorhersehbare Risiken	193
24.2	Zielbeiträge der Handlungsalternativen sichtbar machen	194
24.3	Methoden und Werkzeuge als Entscheidungstechniken	194
24.4	Methoden der Investitionsrechnung	195
24.5	Argumentenbilanz	195
24.6	SWOT-Analyse	196
24.7	Konsequenztabelle	197
24.8	Entscheidungsbaum	198
24.9	Nutzwertanalyse	199
24.10	Risikoanalyse	201
24.11	Methoden und Werkzeuge zur Entscheidung auswählen und nutzen	201
24.12	Entscheidungsprobleme, Handlungsalternativen und mögliche Folgen aufzeigen	202
25	Genehmigung und Freigabe	205
25.1	Freigabeempfehlung	205
	Literatur	207
26	Projektorganisation	209
26.1	Stablinien-Projektorganisation	209
26.2	Reine Projektorganisation	211
26.3	Matrixorganisation	212

26.4	Ausbalancierte Matrixorganisation	213
26.5	Befugnisse und Zieleverantwortung des Projektleiters.	213
26.6	Formeln zur Bewertung des Projektfortschritts	214
	Literatur.	215
27	Testmethoden.	217
27.1	Einsatz von Testmethoden	217
27.2	Geeignete Testmethoden und Teststrategie	218
27.3	Testing Maturity Model (TMMI))	219
27.4	TMap (Test Management Approach).	221
27.5	Vorteile von TMap	224
	Literatur.	224
28	Reifegrad des Testmanagements nach TPI Next	225
28.1	Ermittlung des Reifegrads des Testprozesses	225
28.2	Testorganisation im Reifeprozess nach TPI Next	227
28.3	Einteilung der Reifegrade	228
28.4	Reifeprozess für die Testorganisation	228
28.5	Reifegrad „Initial“	229
28.6	Reifegrad „Kontrolliert“	229
28.7	Reifegrad „Effizient“	230
28.8	Reifegrad „Optimierend“	231
28.9	Unterschiede TMMI und TPI.	232
	Literatur.	233
29	Besonderheiten der Testorganisation in agilen Projekten.	235
29.1	Agile Projekte	235
29.2	Agile Methoden	236
29.3	Organisation und Ziele agiler Projekte	238
29.4	Agile Projekte und traditionelle Organisation	239
29.5	Notwendige Voraussetzungen zur Testorganisation in agilen Projekten	239
29.6	Grundsätze für agiles Testen	240
	Literatur.	241
30	Künstliche Intelligenz und kognitives Testen	243
30.1	Künstliche Intelligenz	243
30.2	Anwendungsbereiche künstlicher Intelligenz	244
	Literatur.	245
	Nachwort	247
	Stichwortverzeichnis.	253



Zusammenfassung

Der IEEE 829 definiert Basis-Dokumente für den Softwaretest. Das Testkonzept beschreibt die Ergebnisse der Testplanung. Unterschiedliche Arten von Testdokumenten dokumentieren die Testinhalte, den Testfortschritt und die Ergebnisse der Testdurchführung.

1.1 Normen

Normen helfen bei der Standardisierung von Geschäftsprozessen und Abläufen. Eine Norm ist ein Dokument, das Anforderungen an Produkte, Dienstleistungen oder Verfahren festlegt. Sie schafft somit Klarheit über deren Eigenschaften, erleichtert den freien Warenverkehr und fördert den Export. Sie unterstützt die Rationalisierung und Qualitätssicherung in Wirtschaft, Technik, Wissenschaft und Verwaltung. Sie dient der Sicherheit von Menschen und Sachen sowie der Qualitätsverbesserung in allen Lebensbereichen [DIN2019].

Der Standardisierungsprozess im Softwaretest hat wesentlich später eingesetzt als in anderen Industrien. So wurde z. B. der Zertifizierungsprozess im Bereich von Softwaretestern erst ab Mitte der 1990er-Jahre in Großbritannien begonnen, woraus im Jahre 1997 das ISEB-Zertifizierungsprogramm erwuchs (ISEB = „Information System Examination Board“).

Im Jahre 1998 wurde eine erste Variante des IEEE 829 beschrieben, um damit bestimmte Testdokumente überhaupt zu etablieren. Diese Variante wurde im Jahr 2008 erweitert und aktualisiert.

Die Definition **IEEE 829 Standard for Software Test Documentation** ist ein vom IEEE (Institute of Electrical and Electronics Engineers) veröffentlichter Standard, der ei-

nen Satz von acht Basis-Dokumenten zur Dokumentation von Softwaretests beschreibt. Die aktuelle Version ist die IEEE 829-2008. Der Standard beschreibt Form und Inhalt der jeweiligen Dokumente. Er schreibt jedoch nicht vor, welche der jeweiligen Dokumente zwingend verwendet werden müssen.

Dieser Standard kann für traditionelle, nach dem Wasserfallmodell ablaufende Softwareprojekte genauso wie für agile Vorhaben angewendet werden.

Im September 2013 wurden die ersten Teile des internationalen Standards **ISO/IEC/IEEE 29119** veröffentlicht, der den IEEE 829 Standard for Software Test Documentation international ersetzt.

Der Standard beschreibt acht Dokumente, die sich wie folgt in drei Kategorien unterteilen lassen.

1.2 Testkonzept

Testkonzept (Test Plan): Das Testkonzept bestimmt Abgrenzung, Vorgehensweise, Mittel und Ablaufplan der Testaktivitäten. Es bestimmt die Elemente und Produktfunktionen, die getestet werden sollen, die Testaufgaben, die durchgeführt werden müssen, das verantwortliche Personal für jede Aufgabe und das Risiko, das mit dem Konzept verbunden ist.

Das Testkonzept dient als Leitfaden, um Testaktivitäten zu planen und zu organisieren.

Im Folgenden ist eine Struktur angegeben, die sich für die Erstellung von Testkonzepten als sinnvoll erwiesen hat.

1. Einführung
 - 1.1. Identifikation des Testkonzeptes
 - 1.2. Geltungsbereich und Umfang
 - 1.3. Referenzen
 - 1.3.1. Externe Referenzen
 - 1.3.2. Interne Referenzen
 - 1.4. Zu testendes System und Testobjekt
 - 1.5. Überblick über die Testaufgaben
 - 1.5.1. Organisation
 - 1.5.2. Projekttestplan
 - 1.5.3. Integrationsstufen
 - 1.5.4. Ressourcenübersicht
 - 1.5.5. Zuständigkeiten
 - 1.5.6. Werkzeuge, Techniken, Methoden, Metriken
2. Details
 - 2.1. Testprozess und Teststufen
 - 2.2. Dokumente
 - 2.3. Abweichungs- und Änderungsmanagement
 - 2.4. Berichtswesen

3. Allgemeines

3.1. Glossar

3.2. Änderungsdienst und Historie [QYTE2019]

Das Testkonzept definiert Gültigkeitsbereich, Vorgehensweise, Zeitplanung und die benötigten Ressourcen der beabsichtigen Testaktivitäten. Dabei werden Testobjekte, zu testende Features und die einzelnen Testaufgaben identifiziert.

Eine Zuordnung der Testaufgaben auf einzelne Mitarbeiter kann ebenfalls vorgenommen werden. Manchmal wird dafür auch auf ein entsprechendes eigenes Planungsdokument verwiesen.

Der Test soll unabhängig von der Entwicklung stattfinden. Nur wenn der Test eine neutrale, eigene Instanz ist kann er diese Unabhängigkeit gewährleisten. Die daraus folgende organisatorische Festlegung wird ebenfalls im Testkonzept beschrieben.

Darüber hinaus werden die Testumgebung, das Testentwurfssverfahren und die anzuwendenden Verfahren zur Messung der Testaktivitäten und Testergebnisse beschrieben und deren Auswahl begründet. Das Testkonzept enthält ein Kapitel über Risiken und mögliche Gegenmaßnahmen, wenn diese Risiken eintreten.

Das Testkonzept dokumentiert somit die Ergebnisse des Testplanungsprozesses.

1.3 Testspezifikation („test specification“)

Die Testspezifikation enthält die Dokumente Testentwurfsspezifikation, Testfallspezifikation und Testablaufspezifikation.

Die **Testentwurfsspezifikation** („test design specification“) verfeinert die Beschreibung der Vorgehensweise für das Testen der Software. Sie identifiziert Features und Funktionen, die von definierten entsprechend zugeordneten logischen Testfällen abgedeckt werden müssen. Sie beschreibt außerdem die Testfälle und Testabläufe, die benötigt werden, um die Testfälle als erfolgreich durchgeführt betrachten zu können. Die Kriterien, nach denen Tests bestanden oder nicht bestanden sind, werden dabei pro Funktion definiert [SETH2019].

Die **Testfallspezifikation** („test case specification“) definiert die zu verwendenden Eingabewerte und die der zu erwartenden Ausgabewerte. Außerdem werden in der Testfallspezifikation die dazu notwendigen Vor- und Nachbedingungen, die Ziele und Testaktionen beschrieben. Die Priorität und Dauer der Testdurchführung sollen, vor allem bei manuell durchzuführenden Testfällen, ebenfalls dokumentiert werden. Bei Bedarf ist auch zu beschreiben, wie nach der Testdurchführung das System auf einen Anfangszustand zurückgesetzt werden muss.

In der Norm IEEE 829 sind Testfälle vom Testdesign getrennt. Das erlaubt die Verwendung der Testfälle in unterschiedlichen Designs und die Wiederverwendung in anderen Situationen. In der Praxis sind häufig die Testentwurfsspezifikation und Testfallspezifikation zusammen in einem Dokument gespeichert und bestehende Testfälle werden dabei für

ein neues Projekt nur kopiert. Der modulare Aufbau in unterschiedlichen Dokumenten führt aber zu einer höheren Qualität, auch wenn es anfangs einen Mehraufwand darstellt.

Die **Testablaufspezifikation** („test procedure specification“) ist die Beschreibung aller Schritte zur Durchführung der spezifizierten Testfälle und Implementierung des zugehörigen Test-Designs. Bei vielen Testfällen sind zunächst die Voraussetzungen herzustellen (z. B. mehrere Buchungsschritte), bevor mit dem eigentlichen Testfall überhaupt begonnen werden kann. Manche Tests bestehen aus einer Reihe von Schritten. Die Testablaufspezifikation wird manchmal auch als Testprozedurbeschreibung, Testskript oder Testdrehbuch bezeichnet. Oft werden auch Mischformen von Testfallspezifikation und Testablaufspezifikation verwendet.

In vielen professionellen Testmanagement-Tools werden der Testablauf und die Trennung von Testfall- und Testablaufspezifikation mittels verschiedener Bereiche abgebildet.

1.4 Testbericht („test reporting“)

Das Berichtswesen soll die Ergebnisse aller Testläufe einer Version zusammenfassen.

Der **Testobjektübergabebericht** („test item transmittal report“) beschreibt die Übergabe der Testfälle für den Fall, dass getrennte Entwicklungs- und Testteams eingebunden sind oder für den Fall, dass ein offizieller Zeitpunkt für den Beginn einer Testausführung erwünscht ist. Im Testobjektübergabebericht wird eine definierte Version an ein Testteam übergeben, und die testbaren Features sind dort aufgeführt.

Das **Testprotokoll** („test log“) dient zur chronologischen Aufzeichnung der Ereignisse während einer Testausführung. Bei jeder Testdurchführung ist zu definieren, welche Logfiles von der Entwicklung benötigt werden, um Fehlverhalten reproduzieren zu können. Das ist gerade bei automatisierten Tests, die über Nacht oder am Wochenende laufen, wichtig. Zu beachten ist aber, dass ein hoher Log Level nicht nur viele Informationen aufzeichnet und daher viel Speicherplatz benötigt, sondern auch, dass durch das verstärkte Logging die Antwortzeit und die Systemlast derart ansteigen kann, dass es zu Fehlern kommt, die unter Normalbetrieb gar nicht auftreten würden.

Der **Testabweichungsbericht** („test incident report“) beschreibt alle Ereignisse, die während einer Testausführung auftreten und weitere Nachprüfungen erfordern. Eine Abweichung ist gemäß Norm IEEE 108 (IEEE Standard for Software Unit Testing) ein Ereignis, das während des Testens auftritt und weiterer Untersuchungen bedarf.

Der **Testabschlussbericht** („test summary report“) fasst die Testaktivitäten zusammen, die mit einer oder mehreren Testentwurfsspezifikationen zusammenhängen. Wenn z. B. der Systemtest oder der Integrationstest einer definierten Software komplett abgeschlossen ist, wird der Testabschlussbericht erstellt, um die nächste Teststufe starten zu können. Wenn es dabei z. B. beim Systemtest nur eine bedingte Freigabe gibt, kann die Software zwar im Abnahmetest weiter getestet werden, aber man muss bestimmte Einschränkungen beachten. Manchmal sind auch Fehler am Ende der Testaktivitäten einer Teststufe noch nicht behoben bzw. noch nicht erfolgreich nachgetestet, so dass man zwar mit diesen Einschrän-

kungen die nächsthöhere Teststufe durchführen, das Produkt aber noch nicht in Produktion ausliefern kann solange diese Fehler noch bestehen. Der Testabweichungsbericht wird in der Praxis häufig mit dem Testabweichungsbericht in einem Dokument zusammengefasst.

Der Testabschlussbericht soll die beim Test aufgetretenen Fehler mit ihrem aktuellen Status dokumentieren, also eine Aussage darüber treffen, welche Fehler festgestellt und während der Testphase behoben wurden. Der Testabschlussbericht wird teilweise auch als **Testbewertungsbericht** bezeichnet. Dieses Dokument enthält auch eine Bewertung des Testprozesses und einen Erfahrungsbericht.

Bei umfangreicheren Projekten dauern einzelne Testphasen über mehrere Wochen bzw. Monate. Daher ist es ratsam, im wöchentlichen Abstand einen aktuellen **Statusbericht** zu erstellen. Bereits im Rahmen der Testvorbereitung sollte man sich dabei überlegen, welche Aussagen man darstellen will und welche Metriken man dazu einsetzen will. Es ist zu empfehlen, das regelmäßige Reporting im Vorfeld der Testphasen zu definieren und abzustimmen. Idealerweise werden Metriken, die nur noch für das laufende Projekt konfiguriert werden müssen, bereitgestellt. Wenn man Fragen nach dem Testfortschritt während der Testdurchführungsphase beantworten soll und dann nicht vorbereitet ist, hat man zusätzlich zu stressigen Projektphasen noch das Problem, Ereignisse ansprechend zu präsentieren und die benötigten Informationen (nicht zu viel aber auch nicht zu wenig) zu sammeln und aufzubereiten.

In diesem Zusammenhang sollte auch der Rhythmus für das Reporting bereits vor Beginn des Projektes geprüft werden: wie aktuell sind die Ereignisse, die in den Testreport einfließen? Wenn am Mittwoch der Report für die Vorwoche erst erstellt und unternehmensweit verteilt wird, oder – noch schlimmer – ein Testreport erst am 10. des Folgemonats für den laufenden Monat präsentiert wird, sind viele Informationen längst veraltet. Ein Reporting über Fortschritt, aber auch Probleme, muss immer zeitnah, also tagesaktuell erfolgen und damit auch zeitnah erstellt werden können.

Bei allen Berichten sollte man die Zeit für die Aufbereitung und Präsentation der Ergebnisse nicht unterschätzen. In aller Regel ist es schwierig, komplexe Zusammenhänge auf wenigen Seiten Reporting zusammenzufassen und grafisch ansprechend darzustellen. Das ist aber unbedingt notwendig für ein umfassendes Berichtswesen gegenüber dem Management und zuletzt für das Eigenmarketing als Testmanager. Die Aufbereitung von Fakten für das Reporting kann auch sehr zeitaufwändig werden. Meiner Erfahrung nach ist in vielen Projekten das Reporting nicht von Anfang an so definiert, dass man sinnvoll mit standardisierten Berichten, die für das individuelle Projekt nur noch konfiguriert werden müssen, darauf aufsetzen kann. Ein klares Reporting ohne Beschönigung, ohne Panikmache und ohne Schuldzuweisungen, sondern als Aufzeigen der Ist-Situation mit einer realistischen Einschätzung verbessert die Kommunikation im Projekt erheblich.

Abb. 1.1 zeigt den Zusammenhang der einzelnen Dokumente im Überblick.

Die Eingangsdokumente „Proj Doc“ (**Projektdokumentationen**) und „Item Doc“ (**Projektprodukt dokumentationen**), sowie die „Test Item“ (Testgegenstand/Software)

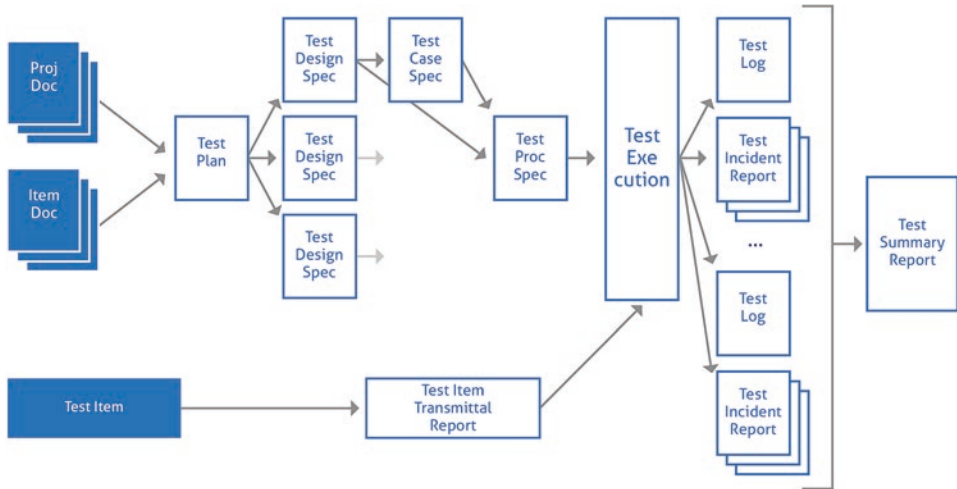


Abb. 1.1 Testdokumente und Ergebnistypen im Test

sind nicht Gegenstand des Standards, sondern werden von den im vorigen Kapitel benachbarten Prozessen definiert und erstellt.

Daneben gibt es weitere typische Dokumente, welche jedoch nicht im IEEE 829 hinterlegt sind. Dazu gehört beispielsweise der **Testfortschrittsbericht** („test progress report“), der **Teststufensteckbrief** etc. Außerdem gibt es weitere Dokumente, welche in der Regel durch das Projektmanagement definiert werden (beispielsweise Steckbrief des Teilprojektes Test, Risikoliste, Offene Punkte Liste, etc.) [PMQS2019].

In diesem Zusammenhang sind auch **Release Notes** zu betrachten. Release Notes sind Dokumente, die im Rahmen der Übergabe von der Entwicklung zum Test zu Beginn der Testdurchführung die Testobjekte identifizieren, ihre Konfiguration, ihren aktuellen Status und weitere Informationen zur eingesetzten Software enthalten.

Bei allen erstellten Dokumenten ist darauf zu achten, die Identifizierung eindeutig vorzunehmen, eine Versionierung und den Dokumentenstatus zu kennzeichnen (z. B. „in Arbeit“, „in Review“, „fertiggestellt“, „abgenommen“). Dazu sollte ein Ablaufplan über den jeweiligen Dokumentenstatus als Basisdokument dem Testprozess zugrunde gelegt werden.

1.5 Integritätslevel

IEEE 829 soll für ein agiles, ebenso wie für ein konventionelles systematisches Vorgehen geeignet sein und bezieht sich genauso auf Systeme wie Software. Mit IEEE 829 wurde aus einem bisher dokumentenorientierten Prinzip „Dokumente folgen Aktivitäten“ nun ein Dokumentationsprozess (konform zu **ISO 12207**, **ISO 15288**) mit Input und Output (aufgabenorientiert).

Mit IEEE 829 werden 4 **Integritätslevel**, je nach Fehlertoleranz des Systems, definiert. Für jedes Level werden bestimmte Dokumente vorgeschlagen:

4 – Katastrophal

Master Test Plan (statisch), Level Test Plan (dynamisch und detailliert), Level Test Design, Level Test Case, Level Test Protocol, Level Test Log, Anomaly Report, Level Integration Test Status Report, Level Test Report, Master Test Report

3 – Kritisch

Master Test Plan (statisch), Level Test Plan (dynamisch und detailliert), Level Test Design, Level Test Case, Level Test Protocol, Level Test Log, Anomaly Report, Level Integration Test Status Report, Level Test Report, Master Test Report

2 – Marginal

Level Test Plan (dynamisch und detailliert), Level Test Design, Level Test Case, Level Test Protocol, Level Test Log, Anomaly Report, Level Integration Test Status Report, Level Test Report

1 – Vernachlässigbar

Level Test Plan (dynamisch und detailliert), Level Test Design, Level Test Case, Level Test Protocol, Level Test Log, Anomaly Report, Level Test Report

In Anlehnung an das V-Modell sind für den Test mehrere Stufen vorgesehen, die Dokumente Test Design, Test Case, Test Protokoll, Test Log, Interim Test Status Report und Anomaly Report werden daher bei Bedarf jeweils für die Stufen Component, Component Integration, System und Acceptance Test erstellt.

Zur Unterstützung des Testmanagers existiert eine Vorlage für einen **Master Test Plan Metrics Report**, dieser schließt Metriken nach **ISO 9126** ein.

Eine Anpassung des Testdokumentationsprozesses an die eigene Organisation (Projekt bzw. Unternehmen) ist dabei wie bei anderen Normen ausdrücklich gewünscht, solange die Anforderungen an eine nachvollziehbare Testdokumentation erfüllt werden. Die Norm soll als Richtschnur und Orientierung dienen, die Ausgestaltung und Umsetzung ist individuell unterschiedlich. Dabei kommt es immer wieder darauf an, keine starren Vorgaben zu definieren, sondern den etablierten Abläufen und Umweltvoraussetzungen so weit wie möglich entgegenzukommen, ohne die begründeten Basisanforderungen zu vernachlässigen.

Über die Testdokumentation hinaus befasst sich der Standard IEEE 829 auch mit der Einbindung des Tests in die Organisation. Er definiert und beschreibt folgende Formen: (ISTQB konform)

- eingegliedert: komplett integriert im Entwicklerteam
- intern: Tester im Entwicklerteam
- integriert: eigenes Team parallel zu den Entwicklern in derselben Organisationseinheit

- eigenständig: eigene selbstständige Organisationseinheit innerhalb des Unternehmens
- unabhängig: komplett unabhängig, außerhalb des Unternehmens, extern vergeben [ROBU2019]

Dabei werden auch Mischformen angetroffen, etwa wenn der Integrationstest einer zugelieferten Komponente zwar an eine externe Organisation ausgelagert ist, der Systemtest und Abnahmetest aber von einer bestimmten Abteilung innerhalb des Unternehmens organisiert und durchgeführt wird.

Literatur

- [DIN2019] : <https://www.din.de/de/ueber-normen-und-standards/basiswissen>, zugegriffen am 09.05.2020
- [QYTE2019] : <https://www.qytera.de/blog/testkonzept-ieee-829-testmanagement>, zugegriffen am 09.05.2020
- [SETH2019] : http://www.sebastianthiem.de/?page=Workshop/004_SoftwareTest.html, zugegriffen am 09.05.2020
- [PMQS2019] : <https://pmqs.de/index.php/testmanagement/prozesse-tm/32-04-ergebnistypen-im-test>, zugegriffen am 09.05.2020
- [ROBU2019] : <http://robert.bullinger.over-blog.com/article-testdokumentation-nach-ieee-829-20510038.html>, zugegriffen am 09.05.2020

Zusammenfassung

Eine Teststrategie stellt die Voraussetzung für planmäßige Handlung und die Grundlage zur Definition der richtigen Testprozesse und Festlegung der Organisation dar. Eine Teststrategie gilt unternehmensweit oder für eine Organisation im Unternehmen und nicht nur für ein einzelnes Projekt. Abweichungen von der Teststrategie im eigenen Projekt sind zu begründen. Die Teststrategie ist im Zusammenhang mit der IT-Strategie zu betrachten. Als Instrument zur Steuerung für die Umsetzung von Projektstrategien empfiehlt sich die Verwendung von Balanced Scorecards.

2.1 Merkmale der Teststrategie

Strategien entwickeln, Pläne schmieden und Ziele umsetzen – das ist das Kerngeschäft des Managements in Unternehmen. Eine Voraussetzung dafür ist strategisches Denken. Strategisch zu denken bedeutet nicht unbedingt, Entscheidungen mit Folgen für die gesamte Organisation zu treffen oder über die Vergabe knapper Etats zu bestimmen: Es reicht aus, wenn man selbst die kleinsten Entscheidungen im Kontext der weitergehenden strategischen Ziele der eigenen Organisation betrachtet. Zum Beispiel ist es hochgradig strategisch, eine Beziehung zu pflegen, wenn sie besondere Erkenntnisse über einen Zulieferer, einen Kunden oder einen Wettbewerber verspricht. Jeder hat also Gelegenheiten, strategischer zu sein [HARV2014].

Die **Teststrategie** ist definiert als eine Dokumentation, die die generischen Anforderungen an das Testen in einem oder mehreren Projekten innerhalb einer Organisation beschreibt, einschließlich der Details darüber, wie das Testen durchgeführt werden soll. Die Teststrategie ist an der **Testrichtlinie** ausgerichtet. Die Testrichtlinie bezeichnet ein Doku-

ment, in dem auf hohem Abstraktionsniveau die Prinzipien, der Ansatz und die wichtigsten Ziele einer Organisation in Bezug auf das Testen zusammenfasst [GLOS2019].

Die Testrichtlinie gilt also in der Regel für das gesamte Unternehmen und nicht für ein einzelnes Projekt. Aufgrund der Testrichtlinie wird auch das Template für Testdokumente, wie z. B. ein Testkonzept, erzeugt. Wenn aber von der unternehmensweiten Teststrategie in einem Projekt aus unterschiedlichen Gründen von der Testrichtlinie abgewichen wird, so ist dies im Testkonzept des entsprechenden Projektes zu vermerken. Man kann dann z. B. die Teile, die vorgesehen sind, als „nicht relevant“ kennzeichnen (das ist besser als sie zu entfernen, denn damit dokumentiert man, dass man an diese Aspekte bei der Konzeptionsphase durchaus gedacht hat und nichts vergessen hat).

Das gilt vor allem für kleinere Projekte, bei denen der Integrationsaufwand geringer ist und die daher keinen kompletten Integrationstest oder Teilintegrationstest erfordern. Man kann auch das Testkonzept klein halten, indem man auf ein anderes Projekt verweist und nur die Abweichungen und Besonderheiten im eigenen Projekt beschreibt. Aber selbst bei geringem Testaufwand von wenigen Tagen, wie es z. B. für zahlreiche Apps typisch ist, sollte man auf die Testrichtlinie und ein generisches Testkonzept verweisen und im projektspezifischen Testkonzept das Projekt beschreiben und auf die Anforderungen referenzieren, Stakeholder, Teststufen und Termine definieren. Ein gewisser Mindestumfang muss in jedem Fall gegeben sein.

Bei der Definition der Teststrategie ist ein wesentlicher Aspekt die Festlegung, wie intensiv auf welcher Stufe getestet werden soll. Das hängt von unterschiedlichen Faktoren ab, beispielsweise dem verfügbaren Budget für das Testen, dem Risiko, der Komplexität des zu entwickelnden Systems, der Menge der Teams oder auch dem Wissen über Testaktivitäten. Der Testmanager muss möglichst viele dieser Faktoren ermitteln und berücksichtigen, oftmals in enger Abstimmung mit weiteren Rollen wie einem Projektleiter oder dem **Product Owner**. Eine Strategie kann sein, dass auf der Modulebene jedes Modul mit Test Driven Design entwickelt werden muss, bevor es integriert wird und den Integrationstest durchläuft und in der Folge auf Systemtestebene alle User Stories manuell durchlaufen werden müssen. Auf jeder Teststufe kann die Strategie auch weiter verfeinert werden. Beispielsweise kann auf Systemtestebene jede als kritisch eingestufte Userstory von mindestens zwei Testern sowohl erfahrungsbasiert als auch systematisch getestet werden, während für weniger kritische User Stories nur einer der beiden Tester Tests durchführt. Der Entwicklertest auf Unit-Ebene kann sowohl auf White-Box- als auch auf Black-Box-Basis stattfinden und wird normalerweise von den Entwicklern selbst durchgeführt. Dabei geht es nicht nur um die korrekte Funktionalität bei gültigen Eingaben, sondern auch um das korrekte Abfangen ungültiger Eingabewerte. Jede „kleinste Einheit“ („Unit“) wird möglichst isoliert getestet und geprüft. Bei agilen Entwicklungslebenszyklen hat sich der Einsatz von Test-First- oder Test-Driven-Development etabliert. Im Integrationstest wiederum werden die Schnittstellen zwischen Einheiten geprüft und somit das Zusammenspiel mehrerer Einheiten getestet. Auch wenn jede Einheit für sich korrekt arbeitet, kann das Zusammenspiel der Komponenten trotz allem zu fehlerhaften Ergebnissen führen. Der Integrationstest kann in mehrere Phasen aufgeteilt werden, um von „kleinen“ Integrations-

schritten zu „großen“ Integrationsschritten alles abzudecken. Auf dieser Teststufe spielt insbesondere bei iterativer Entwicklung der Regressionstest eine große Rolle. Mit der Prüfung nichtfunktionaler Aspekte wie Performanz oder Security kann bereits im Integrationstest begonnen werden. Der Systemtest wiederum wird auf Black-Box-Basis durchgeführt und umfasst die Prüfung des kompletten Systems, in der Regel von der Benutzeroberfläche aus. Geprüft werden soll, ob auf Basis der Eingaben die Ziele der Benutzer mit der Anwendung erreicht werden können. Auf dieser Teststufe finden auch nichtfunktionale Tests statt. Alle diese Teststufen finden in der Umgebung der Entwicklung statt. Erst der Akzeptanztest findet in der Umgebung der künftigen Anwender statt und wiederholt Testfälle, die im Systemtest bereits durchlaufen wurden. Der Modultest wird im Rahmen des agilen Vorgehens täglich ausgeführt, der Integrationstest am Ende des Sprints. Systemtests werden nach einigen Sprints begonnen, wobei komplexe Test-szenarien getestet werden. Der letzte Sprint vor dem Release wird dann noch einmal explizit für die Qualitätssicherung genutzt, und Akzeptanztests werden durchgeführt.

Im Rahmen der Nachbereitung und Auswertung wird fortlaufend geprüft, ob die initiale Teststrategie sinnvoll ist und ggf. angepasst werden muss. Stellt sich zum Beispiel während der Entwicklung heraus, dass eine bisher nicht als risikoreich eingeschätzte Komponente risikoreich ist, sollten Testaufwände hier zusätzlich konzentriert werden, beispielsweise durch weitere Modul- oder Integrationstests. Fehler, die während der Tests auf den unterschiedlichen Stufen gefunden werden, sollten detailliert (z. B. in einem Bug-tracking-Werkzeug) dokumentiert und analysiert werden, um die zukünftige Qualitätssicherung weiter zu optimieren und ggf. zu entscheiden, ob Fehler schon früher gefunden werden können bzw. gänzlich vermieden werden können [PQ4A2015].

Je besser die Teststrategie ausgefeilt ist, desto mehr kann man im einzelnen Kleinprojekt darauf verzichten, alle Überlegungen noch einmal zu beschreiben. Ein modularer Aufbau des Testkonzeptes ist daher in jedem Fall zu empfehlen.

Strategien müssen sich im langfristig an neue Trends, neue Verhaltensweisen, neue Technologien und ein verändertes Unternehmensumfeld anpassen. Das gilt auch für die Teststrategie. Strategien sollen, wie Scheuklappen für Pferde, dabei helfen, sich auf einer Geraden zu bewegen und ein bestimmtes Ziel zu verfolgen, verhindern aber die Nutzung der peripheren Sehkraft. Dadurch, dass sie die Bemühungen und Aufmerksamkeit aller Teile einer Organisation fokussieren, läuft die Organisation Gefahr, ihre Strategie dann nicht zu ändern, wenn sie dies tun sollte. Das führt dazu, dass vor allem Großunternehmen an starren überkommenen Strukturen festhalten, obwohl sich das technologische und gesellschaftliche Umfeld längst geändert hat. Dadurch werden notwendige Innovationen verzögert und verhindert. Oft ist es besser, sich langsam vorwärts zu bewegen, immer nur ein kleines bisschen, nicht zu weit nach vorne schauend – aber sehr wachsam! –, sodass das Verhalten innerhalb kürzester Zeit verändert werden kann [BUSW2019].

Die Teststrategie betrifft mehrere Teststufen. Je nach Aufgabenstellung werden auch softwarespezifische Themen für den Integrationstest (vor allem in Embedded-Systemen von Software und Hardware), Komponententest und Systemtest betrachtet.

Die Teststrategie betrachtet dabei unterschiedliche Aspekte des Testens:

- Kunden und firmeninterne Qualitätsvorgaben stellen Anforderungen an den Test und beeinflussen Art und Umfang der Tests. Idealerweise sind dazu messbare Ziele definiert.
- Vor Entwicklung der Testfälle müssen die Testvoraussetzungen geklärt sein, die Testplanung durchgeführt werden, die Integrationsstrategie festgelegt sein und der Testumfang genau bestimmt werden. Struktur und Inhalt der Testspezifikationen und Testprotokolle werden dabei festgelegt.
- Die Testumgebung für das Testobjekt sowie die Testwerkzeuge werden definiert. Ein Prüfverfahren wird festgelegt, um die zuverlässige und fehlerfreie Funktionalität zu gewährleisten.
- Zur Testdurchführung gehören auch individuelle Testvorbereitungen und Kriterien für das Testende bzw. Abbruch der Tests, die Prüfung der Testergebnisse und Freigabe der Software.

Beim Test von Embedded-Software sind dabei spezielle Aspekte zusätzlich zu rein softwarebasierten Anwendungen zu prüfen:

- Hardwareabhängigkeiten,
- Echtzeit-Randbedingungen,
- Einschränkungen der Testumgebung:
 - Funktionalität,
 - Verfügbarkeit,
- Störungen:
 - Bitfehler in digitalen Signalen,
 - Unterbrechung von Signalen.

Häufig werden bereits vom Kunden definierte Anforderungen an die Software gestellt. Dies ist oft bei Embedded-Software, die in Investitionsgüter oder Massenartikel eingebaut wird, der Fall. Dabei geht es dem Kunden auch darum, eigene Testaufwände zu sparen, indem er von seinen Lieferanten, die ihm Systeme oder Komponenten zuliefern, die Erfüllung spezifizierter Qualitätsziele erwartet. Oft gibt es dabei vertraglich festgelegte Forderungen, Testprotokolle als Teil der Produktdokumentation mit auszuliefern.

Auch die Verifikationsmaßnahmen und die Testprozesse werden dabei in vielen Fällen vom Kunden mitbestimmt. Die Prozessvorgaben orientieren sich meist an Normen und Standards, z. B. den Sicherheitsnormen **IEC 61508**, **ISO 26262** oder **EN 50128**, **SPICE** und **CMMI**. Wenn diese Prozessvorgaben in der betrachteten Organisationseinheit nur mangelhaft umgesetzt sind, besteht ein Projektrisiko, weil dann ggf. Testergebnisse am Ende vom Kunden nicht akzeptiert werden. Sollte eine Zertifizierung nach einer Norm Bestandteil des Projektes sein und bereits zur Konzeptphase bestimmte Schwächen in der Organisation bekannt sein, die zu Mehraufwand führen, um diese Zertifizierung überhaupt zu bestehen, ist auch dieser Punkt als Risiko im Testkonzept des Projektes aufzuführen.

Qualitative Vorgaben an den Test betreffen die Erstellung der Testdokumentation und die Vorgehensweise bei der Testfallimplementierung. Eine gewisse Testabdeckung und ein erwarteter Umfang an Regressionstests werden dabei ebenfalls vorgegeben [EISE2012].

Es kann vorkommen, dass man aus bestimmten Gründen von der Teststrategie, die für das gesamte Unternehmen bzw. die gesamte Organisation festgelegt wurde, abgewichen werden muss, etwa, wenn zusätzliche Integrationsstufen erforderlich sind oder bestimmte Rollen neu definiert werden müssen. Sollte dies der Fall sein, sind diese Abweichungen im Testkonzept für das betroffene Projekt zu begründen und die Auswirkungen zu bewerten.

2.2 Integrationsstrategie

Zur Teststrategie gehört eine Integrationsstrategie, weil sich die Software in der Regel aus mehreren Modulen zusammensetzt.

Modellhaft testet man die einzelnen Softwaremodule erst einzeln, integriert sie dann zu Subsystemen und am Ende zu einem Gesamtsystem. Dabei ist zu prüfen, ob die einzelnen Module richtig zusammenarbeiten. Nun ist es aber häufig so, dass einzelne Module noch nicht fertig sind und man nur Platzhalter (Mocks, Stubs) verwenden kann. In der Regel sind auch die Projekttermine zu eng, um die einzelnen Modultests komplett abschließen zu können, bevor man in die nächste Teststufe kommt.

Zu bedenken ist auch, dass der **Testaufwand** der einzelnen Module unterschiedlich hoch ist. So ist z. B. bei sicherheitskritischen Modulen im Vergleich zu nichtsicherheitsrelevanter Software ein höherer Testaufwand erforderlich. Teilweise müssen auch zusätzliche Testwerkzeuge zum Nachweis der Testabdeckung eingesetzt werden. Bei Softwarekomponenten, die von anderen Unternehmen zugeliefert werden, sind ebenfalls Integrationstests vorzusehen. Hardwarenahe Software ist meist auf dem Zielsystem sinnvoll zu testen.

Die Integrationsstrategie bestimmt, in welcher Reihenfolge die einzelnen Module zu einem Gesamtsystem integriert werden. Es empfiehlt sich, die für die Verarbeitung wichtigsten Module dabei am Anfang zu integrieren und dabei die Eingabemodule vor den Ausgabemodulen. Eine Parallelisierung von Integrationsschritten (z. B. Applikationssoftware und hardwarenahe Basisfunktionen) ist wegen terminlich enger Projektpläne in aller Regel ohnehin vorgegeben.

2.3 Testspezifikationen

Zur Teststrategie gehört es, die Erstellung von Testspezifikationen einzuplanen. Die Testspezifikationen sollen aus Start der Testdurchführung im Wesentlichen fertiggestellt und einem Review unterzogen worden sein. Zwar muss man auch während der Testdurchführung die Testspezifikation prüfen und detaillieren und um Testfälle, die dem Tester beim explorativen Test auffallen, ergänzen, die Struktur und der Rahmen soll aber in einer frühen Phase der Testaktivitäten vorhanden sein und die Beschreibung grundsätzlich vorliegen.

2.4 Einsatz von Tools

Die Teststrategie definiert, welches Tool eingesetzt wird und wie der Aufbau der einzelnen Testfälle strukturiert ist. Wichtig ist, dass man sich im gesamten Unternehmensbereich auf ein definiertes Tool einigt und nicht mehrere Tools nebeneinander einsetzt. Der Aufwand für Vergleich und Benchmarking der einzelnen Bereiche steigt ansonsten gewaltig an und es besteht ein Risiko für den Testmanager, umfassend aussagefähig zu sein.

Folgende Informationen sollen im Tool auf jeden Fall vorhanden sein:

1. **Testfall-ID:** Eindeutige Identifikationsnummer, sodass der Testfall jederzeit wieder auffindbar ist und archiviert wird.
2. **Testgegenstand/Testobjekt:** Die zu testende Anwendung wird beschrieben. Eine kurze Beschreibung sollte bereits aus der Überschrift hervorgehen.
3. **Testkonfiguration/Testvoraussetzungen/Testdaten:** Welche Hardware- und Softwarekonfiguration und welche Testumgebung müssen vorhanden sein? Wie sind die Umgebungsparameter? Müssen andere Schritte ausgeführt werden, bevor mit dem eigentlichen Testfall begonnen werden kann? Ggf. kann man auf eine exakt bezeichnete Testumgebung referenzieren, die in einem Dokument des Konfigurationsmanagements beschrieben ist.
4. **Testbeschreibung:** Genaue Beschreibung des Testfalls und der Testprozeduren. Die Schritte, die bei der Ausführung nacheinander durchgeführt werden, sind zu dokumentieren.
5. **Referenzen:** Bezug zum Pflichtenheft oder zu anderen Testfällen. Unbedingt empfehlenswert ist, dabei eine Matrix einzusetzen, in der Anforderungen aus dem Pflichtenheft und zugeordneter Testfall gegenübergestellt werden. So ist man jederzeit auskunftsfähig, welche Anforderung mit welchem Testfall nachgewiesen werden kann, und – in anderer Richtung – warum ein bestimmter Testfall erforderlich ist. Das ist auch Voraussetzung für eine erfolgreiche Zertifizierung z. B. nach SPICE.
6. **Priorität:** Ist der Testfall unbedingt erforderlich oder eher nachgeordnet? Gerade bei Regressionstests kann man den Testfall dann evtl. weglassen oder auch bei kurzfristig hintereinander eintreffenden Lieferungen. Allerdings sollten auch Testfälle niedriger Priorität mindestens einmal für eine erfolgreiche Abnahme mit der finalen Version getestet werden. Sollten Testfälle aus Zeitgründen nicht getestet werden können, ist das als Risiko bei der Abnahme zu dokumentieren.
7. **Detailfelder oder Kommentarfelder** für Ergänzungen und spezielle Hinweise sind vorzusehen.
8. **Soll-Ergebnis bzw. erwartetes Ergebnis:** Welches Ergebnis ist zu erwarten und soll laut Spezifikation erzielt werden? Dabei ist darauf zu achten, dass das Ziel messbar formuliert ist. Die Beschreibung muss daher so ausführlich sein, dass sie keinen Interpretationsspielraum offenlässt.
9. **Ist-Ergebnis:** Beschreibung des tatsächlichen Ergebnisses.

10. **Bewertung des Ergebnisses:** War der Test erfolgreich, also „OK“ oder „NOK“.
11. **Referenz zum Fehler:** Falls ein Fehler vorliegt, sollte die Nummer hier vermerkt sein. Es sollte auch aufgeführt werden, in welcher Phase der Fehler entstanden ist (Grobentwurf, Feinentwurf/Design, Implementierung, Test), um Rückschlüsse für den gesamten Entwicklungs- und Testprozess zu gewinnen.
12. **Kommentare und Anmerkungen** zu den Ergebnissen (freier Text) sind aufzunehmen.
13. **Tester**, der den Test durchgeführt hat, bzw. bei automatischen Skripts eine Referenz zum Programmlauf sind festzuhalten.
14. **Datum und Uhrzeit** der Testdurchführung sind zu notieren.
15. **Testdauer:** Die Testdauer ist vor allem für manuelle Testdurchführung zu erheben, um gerade bei knapper Zeit und Ressourcen die möglichst durchzuführenden Tests zu definieren.

Je nach Besonderheiten der Testumgebung und des betrieblichen Umfelds können weitere Angaben erforderlich sein. Professionelle Testtools bieten im Allgemeinen eine Auswahl an möglichen Feldern an, die bei Einsatz im Unternehmen bzw. im Projekt individuell konfiguriert werden können.

2.5 Fragen zur Umsetzung der Teststrategie

Im Zusammenhang mit der Teststrategie empfiehlt es sich, einige Fragen zu beantworten, die auch für die IT-Strategie gelten:

- Wozu ist das Projekt gut, was soll der Test bewirken?
- Was wird gemessen und wie wird gesteuert? (Betrachtungsobjekt – Produkt/Prozess/Ressource/Rahmenbedingung)
- Welche Aspekte gilt es zu berücksichtigen? (Führungsgrößen/Eigenschaften/Metriken)
- Mit welchem Maß erfolgt die Messung? (Maßeinheit/Messgröße zur Messung der Ausprägung)
- Welche Zielwerte werden verfolgt? (Ausgewählte Zielwerte)
- Wie wird gemessen? (Messmethode)
- Wann wird gemessen? (Zeitpunkt und Zeitraum der Messung)
- Wo wird gemessen? (Ort/Stelle)
- Wer führt die Messung durch? (Person/Gerät)
- Wie werden die Messergebnisse dokumentiert? (Qualitätsdaten)
- Wie werden Messergebnisse ausgewertet? (Vergleich gegen Zielwerte)
- Welche Maßnahmen werden ggf. zur nachhaltigen Verringerung der Abweichung bzw. Streuungen ergriffen und wer kontrolliert deren Wirkung?
- Welche Maßnahmen werden ggf. zur Vorbeugung ergriffen?

Zu Beginn der 1990er-Jahre führten Robert S. Kaplan und David P. Norton eine Studie bei zwölf großen amerikanischen Unternehmen durch. Ziel war es zu ermitteln, wie diese Unternehmen ihre Leistung messen und welche Defizite die jeweils angewendeten Methoden haben. Es ging um das wichtige Thema Performance Measurement (Leistungsmessung). Die Studie fand unter anderem heraus, dass es folgende zentrale Probleme bei der Leistungsmessung gibt:

- Die Messsysteme sind unglaublich kompliziert und differenziert. Vor lauter Messzahlen und komplexen Berechnungssystemen gehen die wesentlichen Informationen verloren.
- Es wird viel im operativen Bereich gemessen, ohne dass es überschaubare, zentrale und strategische Informationen gibt.
- Die meisten Kennzahlen, die für die Leistungsmessung genutzt werden, bilden nur die Vergangenheit ab. Sie sind wenig geeignet, um Entwicklungen in der Zukunft abzuschätzen.
- Die meisten Kennzahlen sind ausschließlich auf finanzielle Aspekte ausgerichtet, sie bilden letztlich den Geldfluss ab. Andere Faktoren, die nicht unmittelbar mit Geld zu tun haben, werden nicht betrachtet. Das bedeutet, dass entscheidende Elemente der Unternehmensstrategie gar nicht in Kennzahlen abgebildet werden [BUSW2020].

2.6 Balanced Scorecard

Daher wurden aus den Befunden der Studie die **Balance Scorecards** (BSC) entwickelt, die sich inzwischen als bewährtes Instrument zur Steuerung der Umsetzung von Strategien etabliert haben. Die Konzentration auf die strategischen Ziele erfordert eine Konzentration auf ca. 20 Ziele. Um die Zielerreichung planen und verfolgen zu können, werden die einzelnen Ziele mit monetären und nichtmonetären Messgrößen beschrieben und deren Sollwerte beschrieben. Strategische Aktionen zu den einzelnen Zielen stellen die Zielerreichung sicher. Jede strategische Aktion erhält Termin- und Budgetvorgaben sowie einen Verantwortlichen. Aufgrund dieser Stringenz wurde der BSC der Untertitel „From Strategy to Action“ beigelegt. Der BSC-Ansatz ist dadurch charakterisiert, dass die strategischen Ziele jeweils einer konkreten Betrachtungsweise – der sog. Perspektive – zugeordnet werden. Diese Zuordnung verhindert einseitiges Denken. Durch das Prinzip, alle – auch die nicht monetären – Sichtweisen zu berücksichtigen, die für die strategische Positionierung des Unternehmens wichtig sind, und alle Sichtweisen zu verknüpfen, macht die BSC komplexe Zusammenhänge transparent.

Die Verknüpfung erfolgt mithilfe von Ursache-Wirkung-Ketten. Dabei wird in der Planung nach dem Top-down- und in der Umsetzung nach dem Bottom-up-Ansatz gearbeitet. Im Rahmen der BSC werden vier Perspektiven unterschieden:

- **Finanzperspektive:** Welche Zielsetzungen werden aus den finanziellen Erwägungen der Kapitalgeber abgeleitet?
- **Kundenperspektive:** Welche Ziele sind hinsichtlich Struktur und Anforderungen unserer Kunden zu setzen, um die finanziellen Ziele zu erreichen?
- **Prozessperspektive:** Welche Ziele sind hinsichtlich unserer Prozessqualität zu setzen, um die Ziele der Finanz- und Kundenperspektive zu erreichen?
- **Potenzialperspektive:** Welche Ziele müssen hinsichtlich unserer Potenziale gesetzt werden, um den aktuellen und zukünftigen Herausforderungen gewachsen zu sein?

Die Balanced Scorecard kann in der Linienorganisation, in (großen) Projekten und im Multiprojektmanagement für folgende Zwecke eingesetzt werden:

- Planung, Entwicklung und Umsetzung von Strategien,
- Ziel- und Leistungsvereinbarungen mit Führungskräften,
- Überwachung der Strategieumsetzung,
- Präsentation der Strategie vor Stakeholdern.

Die folgende Tabelle (Tab. 2.1) zeigt die Stärken und Schwächen der Balanced Scorecard: Neben Balanced Scorecards finden auch Projekt-Scorecards Anwendung, mit denen sich die aktuelle **Projektesituation** gut darstellen lässt (Tab. 2.2).

Eine geeignete grafische Darstellung der Projektesituation zeigt Abb. 2.1.

Beim Einsatz einer Projekt-Scorecard ist darauf zu achten, dass

- eine besondere Informationskultur gefordert ist, die erst gelernt und dann gelebt werden muss;
- sie jederzeit flexibel änderbar ist;

Tab. 2.1 Vor- und Nachteile einer Balanced Scorecard [PRBS2019]

Stärken	Schwächen
• Die Balanced Scorecard ist auf die wesentlichen Steuerungsvariablen fokussiert und hilft dadurch, die kritischen Erfolgsfaktoren auf die Strategie auszurichten. • Sie liefert einen schnellen Überblick, da sie auf einer Seite die wichtigsten Kennzahlen für die Strategieumsetzung darstellt. • Sie ist leicht verständlich und lässt sich damit gut für die Kommunikation einsetzen. • Sie kann flexibel an alle geschäftlichen Anforderungen angepasst werden.	• Die Balanced Scorecard ist kein Instrument für die Strategieentwicklung, sondern ergänzt diese lediglich um den Aspekt der Kennzahlen. • Sie ersetzt nicht die Kosten- und Leistungsrechnung. Dafür ist sie zu ungenau und bildet nicht steuerliche oder handelsrechtliche Aspekte ab. • Sie gibt nicht alle Kennzahlen wieder, die für die Steuerung des Unternehmens bzw. des Projektes relevant sind. Für Teilbereiche des Unternehmens können weitere Kennzahlen erforderlich sein. • Sie kann für unternehmenspolitische Zwecke missbraucht werden.

Tab. 2.2 Darstellung der Projektesituation

Ziele		Zielerreichung	
		1.Quartal	2.Quartal
Finanzen			
	Rentabilität		
	Umsatzsteigerung		
	Gesamt		
Kunden			
	Neukunden		
	Kundenzufriedenheit		
	Gesamt		
Prozesse			
	Termintreue		
	Qualität		
	Gesamt		
Mitarbeiter			
	Mitarbeiterzufriedenheit		
	Absentismus		
	Fluktuation		
	Gesamt		
Gesamtwertung			

- sie in regelmäßigen Abständen aktualisiert wird;
- sie sich auf das Wesentliche beschränkt;
- sie ein Instrument ist, mit dem man täglich arbeitet; neue Vorschläge müssen grundsätzlich auf ihre Zielwirkung diskutiert und bewertet werden.

Für die speziellen Bedürfnisse von Testprojekten bietet es sich an, die Projekt-Scorecard entsprechend abzuändern und neue Kriterien dafür aufzunehmen.

So können z. B. Kriterien wie Testabdeckung, Automatisierungsgrad oder die Anzahl der Fehler als messbare Zielwerte definiert werden.

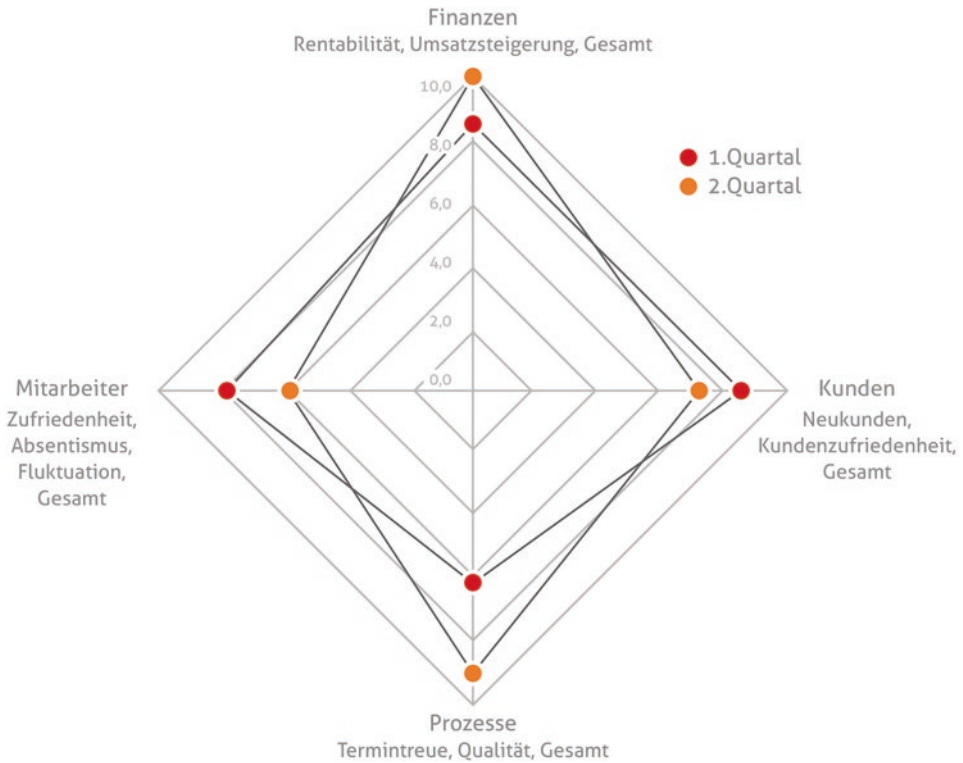


Abb. 2.1 Darstellung der Projektesituation

Literatur

- [GLOS2019] : <http://glossar.german-testing-board.info/v3.21/#t>, zugegriffen am 09.05.2020
- [EISE2012] : Thomas Eißelöffel, Embedded-Software entwickeln, dpunkt Verlag Heidelberg 2012, https://bilder.buecher.de/zusatz/34/34536/34536969_lese_1.pdf, zugegriffen am 09.05.2020
- [BUSW2019] : <https://www.business-wissen.de/hb/was-strategisches-denken-bedeutet/>, zugegriffen am 09.05.2020
- [PRBS2019] : <https://www.projektmagazin.de/methoden/balanced-scorecard-mit-kennzahlen-erstellen-beispiel>, zugegriffen am 09.05.2020
- [PQ4A2015] : <http://www.pq4agile.de/PQ4WP/wp-content/uploads/2015/02/PQ4Agile-AP-2.2-Teststrategie-festlegen-und-Teststufen-aufeinander-abstimmen-V.11.pdf>, zugegriffen am 09.05.2020
- [HARV2014] : <https://www.harvardbusinessmanager.de/blogs/management-strategisches-denken-verbessern-a-950400.html>, zugegriffen am 09.05.2020
- [BUSW2020] : <https://www.business-wissen.de/hb/balanced-scorecard-einfach-und-verstaendlich-erklart/>, zugegriffen am 26.05.2020

Zusammenfassung

Testziele sind eine Untermenge der Projektziele und mit geeigneten Methoden zu messen. Zur Bewertung der Testziele sind geeignete Metriken zu definieren. Dazu sollten Projektampeln als geeignete Bewertungsskalen eingesetzt werden, mit denen Testergebnisse zusammengefasst und beim Reporting bildhaft dargestellt werden können.

3.1 Bedeutung von Testzielen

Ein **Testziel** ist definiert als ein für den Auftraggeber testrelevantes Ziel, oft ausgedrückt durch IT-gestützte Geschäftsprozesse, realisierte Benutzeranforderungen oder Anwendungsfälle, kritische Erfolgsfaktoren, Änderungsvorschläge oder abzudeckende Risiken. Testziele sind projektbezogen eine Untermenge der Projektziele. **Projektziele** sind dabei die Aufstellung von möglichst quantifizierten Anforderungen, die erfüllt sein müssen, damit ein Projekt als erfolgreich abgeschlossen betrachtet werden kann. Um Testziele zu formulieren, muss man sich also fragen, welche Ergebnisse man mit den Softwaretestaktivitäten erreichen will.

Da das Softwaretesten aus zahlreichen Einzelmaßnahmen besteht, die i. d. R. über mehrere Teststufen hinweg und an unterschiedlichen Testobjekten ausgeführt werden, hat zum einen jeder Testfall ein individuelles Testziel. Auch die einzelnen Teststufen haben unterschiedliche Testziele, wie z. B. den Test einer Rechenfunktion, den erfolgreichen Nachweis der Inbetriebnahme oder die Erfüllung definierter Performanceziele bei einem Lasttest. Das erwartete Ergebnis des einzelnen Testfalls dokumentiert diese Testziele. Man kann daher ein Testziel auch als definierte Menge von Softwareeigenschaften, die als messbare Qualitätsziele in der Spezifikations- und Entwurfsphase festgelegt werden, beschreiben. Beim Testen müssen diese Testziele benannt werden. Sie müssen so beschrie-

ben werden, dass ihr Erreichen realistisch gemessen werden kann [ENZY2019]. Das bedeutet im Umkehrschluss, dass Testziele überhaupt erreichbar und messbar sein müssen. Das hört sich zwar selbstverständlich an, häufig sind die Testziele jedoch in der betrieblichen Praxis zu schwammig und damit interpretierbar formuliert.

Mit geeigneten Tests während der Realisierung oder Umsetzung soll die Anzahl der Fehler möglichst niedrig gehalten werden. Außerdem soll dafür gesorgt werden, dass die Entwickler bereits während des Projektes aus erkannten Fehlern lernen können. Testen ist deshalb weit mehr als eine lästige Aufgabe, die man an einen Qualitätsmanager delegieren kann. Im Entwicklungszyklus einer IT-Lösung muss die Qualitätssicherung in den Projektablauf eingebettet sein und darf sich nicht nur auf eine Testphase zum Projektabschluss beschränken.

Mit der Abnahme bestätigt der Auftraggeber, dass seine Anforderungen erfüllt sind, die in Projektauftrag bzw. Lastenheft oder Kundenspezifikation beschrieben sind. Die Abnahme ist Voraussetzung für den Projektabschluss. Sie hat darüber hinaus rechtliche Auswirkungen, z. B. für Rechnungsstellung oder Gewährleistung. Grundlage der Abnahme ist der Abnahmetest, der überprüft, ob alle Abnahmekriterien erfüllt sind. Wichtig ist, dass die Abnahmekriterien frühzeitig definiert werden und allen Beteiligten bekannt sind.

Bei IT-Projekten verbindet das Testen somit die wertschöpfenden Projektprozesse (d. h. die Erstellung) und die Projektmanagementprozesse, z. B. des Qualitätsmanagements und der Abnahme. Zum einen müssen die **Testprozesse** deshalb in den Plan des jeweiligen Projektes integriert werden, zum anderen muss eine Testorganisation eingerichtet werden, die Test- und Managementprozesse aufeinander abstimmt [PRMA2019].

Darüber hinaus gibt es globale Testziele: was will ich mit dem Test erreichen, wie weit soll mein Nachweis gehen? Dazu gehört auch eine eindeutige Abgrenzung, was der Test leisten kann und was nicht. Welchen Qualitätsstandard soll der Test erfüllen? Wann gilt der Test als bestanden oder erfolgreich? Und mit welchen Metriken und Kenngrößen kann das überhaupt nachgewiesen und dokumentiert werden?

Die Testziele unterscheiden sich je nach Teststufe: im **Komponententest** ist das Testziel das Ein- und Ausgabeverhalten des Testobjektes. Daher wird die Komponente mehreren Testfällen unterzogen, wobei jeder Testfall eine bestimmte Ein- und Ausgabekombination abdeckt. Im **Integrationstest** hingegen ist das Testziel, Schnittstellenfehler aufzudecken. Daher werden hier meist dynamische Tests durchgeführt, die Fehlerzustände im Datenaustausch bzw. in der Kommunikation zwischen den Komponenten aufdecken. Beim **Systemtest** besteht das Testziel darin zu verifizieren, ob und wie gut das fertige System die gestellten funktionalen und nichtfunktionalen Anforderungen erfüllt. Hier geht es darum, falsche, unvollständig oder widersprüchlich umgesetzte oder widersprüchlich definierte Anforderungen aus der Perspektive des Benutzers aufzudecken. Dabei werden auch nicht dokumentierte oder vergessene Anforderungen dokumentiert [SPIL2007].

3.2 Testkenngrößen

Die Verwendung von Testzielen führt zu einer Objektivierung des Testprozesses und einer operationell handhabbaren Ergebniskontrolle. Dazu werden **Testkenngrößen** definiert:

Eine **Testabdeckungskenngröße (TAK)** ist eine tätigkeitsorientierte Testkenngröße, die sich aus der Überprüfung von Systemelementen und deren Berücksichtigung im Test ergibt.

Beispiele zur Testabdeckungskenngrößen zur Spezifikation von Testzielen sind

TAK1 = Anzahl ausgeführte Verzweigungen/Anzahl vorhandener Verzweigungen sowie

TAK2 = Anzahl ausgeführter Module/Anzahl vorhandener Module.

Dabei kann für den Modultest ein konkreter Wert als messbares Testziel definiert werden, z. B. TAK1 = 90 %, TAK2 = 100 %.

Eine **Testergebniskenngröße (TEK)** bezeichnet eine ergebnisorientierte Testkenngröße, die sich aufgrund der Ergebnisse von Testaktivitäten ergibt.

Beispiele für Testergebniskenngrößen sind die mittlere Zeit zwischen gefundenen Fehlern, die benötigte Rechenzeit für einen Test, die benötigte Rechenzeit für einen Test/Anzahl gefundener Fehler, die Kosten für einen Test oder die Kosten für einen Test/Anzahl gefundener Fehler [WALL2011]. Es werden also Indexwerte gebildet, die eine Vergleichbarkeit zwischen verschiedenen Algorithmen, Anwendungen oder Projekten erlauben. Unterschiedliche Parameter können entsprechend gewichtet werden,

Oft erlebt man es, dass zu den entscheidenden Fragen zu wenig Daten verfügbar sind: Wie viele Function Points haben die Anwendungen? Wie hoch wird die Fehlerrate in etwa sein? Wie viele Testfälle sind erforderlich? Dazu kommt, dass viele dieser Parameter erst während des Projektes erhoben werden können. Die Daten liegen in der Regel auch verteilt im Unternehmen vor, um sie zusammenzutragen ist ein erheblicher Aufwand erforderlich, der sich für das einzelne Projekt gar nicht rechnet, sondern nur als übergeordnete Aufgabe der Qualitätssicherung darstellbar ist. Dabei sind oft erhebliche Defizite zu erkennen. Informationen sind versteckt und werden nicht transparent gemacht und können dadurch Probleme gar nicht ausreichend adressieren. In vielen Unternehmen erlebt man eher geschäftigen Aktionismus als zielgerichtetes Handeln aufgrund solcher Daten. Genau in diesem Bereich liegen aber mittel- bis langfristig erhebliche Optimierungspotenziale.

Dazu ist es erforderlich, geeignete Metriken zu erheben, um die Testabdeckung und das Testergebnis messen zu können. Die nachfolgende Liste erhebt keinen Anspruch auf Vollständigkeit, sondern soll lediglich das Mindestmaß an zu verwendenden Metriken ausdrücken:

- Anzahl Anforderungen,
- Anzahl dokumentierte Testfälle,
- Aufteilung der Testfälle nach Priorität,

- Anzahl getestete Testfälle,
- Anzahl erfolgreiche bzw. fehlerhaft getestete Testfälle,
- Anzahl Fehler,
- Dauer der Fehlerbehebung,
- Fehler pro Teststufe,
- Aufteilung der Fehler nach Auswirkung bzw. Priorität.

Die einzelnen Werte sind dabei in Beziehung zu setzen. Zum Beispiel wie folgt: Es sind 200 Testfälle dokumentiert, davon 100 getestet, davon 90 OK und 10 fehlerhaft. 5 der 10 Fehler haben niedrige, 3 mittlere und 2 hohe Auswirkung.

Es empfiehlt sich, für eine Detailbewertung, die aber nicht in regelmäßigen Abständen, sondern eher bei Bedarf vorgenommen wird, weitere Parameter heranzuziehen, um das Projekt zu bewerten. Besonders dann, wenn man im Vergleich mehrerer Projekte wiederkehrende Probleme feststellt, dass z. B. die Fehlerquote unternehmensweit zu hoch ist, ließe sich in diesem Beispiel überprüfen, ob die Qualität der Anforderungen ausreichend ist, ob spezielle Teststufen besonders betroffen sind, ob die Ausbildung der Entwickler gesteigert werden sollte, ob bestimmte Komponenten eine Häufung von Fehlern aufweisen oder ob es äußere Einflussfaktoren gibt. Dafür gibt es zahlreiche geeignete Parameter.

Das regelmäßige Reporting sollte jeweils mithilfe von Grafiken und Charts illustriert und untermauert werden. Ein Bild sagt mehr als 1000 Worte.

3.3 Test-Reporting

Zur Definition der Testziele gehört, sich die Bewertungsskala zu überlegen und welche Aussagen getätigt werden. Über grundlegende Anforderungen beim **Test-Reporting** sollte man sich von Anfang an Gedanken machen. Das gilt vor allem dann, wenn unterschiedliche Gruppen am Test beteiligt sind, die sonst nach unterschiedlichen Kriterien ihren Testfortschritt und ihre Qualität bewerten.

Es sollte vorab geklärt werden, wie umfangreich das Reporting gestaltet werden soll. Das hängt vor allem von der Kritikalität und Größe des Projektes ab. Allzu fein granular können die Informationen im Management nicht mehr sinnvoll verarbeitet werden, zumal das Management mehrere Projekte präsentiert bekommt. Eine rein subjektive Zusammenfassung ohne Referenz auf ermittelte Werte und Parameter zur Zielerreichung ist aber ebenfalls nicht ausreichend. Hier gilt es bereits in der Phase der Testvorbereitung einen geeigneten Kompromiss zu finden. Ein erfahrener Testmanager erkennt in der Regel bereits bei Start eines Projektes, wie zeitkritisch, komplex und herausfordernd es wahrscheinlich wird.

Zur Zusammenfassung der Einschätzung wird meist eine „**Projektampel**“ mit den Farben rot, gelb und grün verwendet. Die Ampelfarben machen die Ergebniswerte und Kennzahlen (= Indikatoren) im Berichtswesen einfacher interpretierbar.

Grün bedeutet, dass alles „im grünen Bereich“ ist. Die definierten Indikatoren zeigen keinerlei Planabweichungen an, alle Entscheidungspunkte wurden gemäß Plan oder besser erreicht. Kurz und gut, es besteht kein Entscheidungs- oder Handlungsbedarf für die übergeordnete Instanz. Wenn man allerdings seine Projekte zu oft mit Grün bewertet, kann das auch so ausgelegt werden, dass man von Anfang an zu pessimistisch geplant und zu viel Ressourcen für das Projekt beantragt und genutzt hat. Man bekommt dann im schlimmsten Fall sogar Ressourcen vom Projekt abgezogen.

Gelb bedeutet Achtung und mahnt zu Vorsicht. Die Indikatoren zeigen Planabweichungen, es besteht Handlungsbedarf. Es müssen neue und/oder zusätzliche Maßnahmen zur planmäßigen Erreichung eines Entscheidungspunkts ergriffen werden. In diesem Fall ist zusätzlich eine Liste der erforderlichen Maßnahmen mit deren Begründungen vorzulegen. Ein Entscheidungs- oder Handlungsbedarf der übergeordneten Instanz wird mit Gelb angekündigt und wird notwendig, falls die getroffenen Maßnahmen nicht wirken. Gelb führt zu einer gewissen Awareness, bedeutet aber auch, dass der Projektleiter und der Testmanager die bestehenden Probleme im Griff haben. Erfahrungsgemäß sind die meisten Projekte im Unternehmen in gelber Farbe in der Projektampel dargestellt.

Rot heißt Anhalten! Die definierten Indikatoren zeigen eindeutige Planabweichungen an. Die Maßnahmen zur planmäßigen Erreichung des Entscheidungspunkts waren nicht wirksam oder nicht möglich. Ein Entscheidungs- oder Handlungsbedarf der übergeordneten Instanz ist in diesem Fall zwingend und üblicherweise auch dringend notwendig. Man benötigt Managementsupport, es kann aber so ausgelegt werden, dass man selbst als Testmanager nicht fähig war, die Probleme zu bewältigen. Daher wird man oft versuchen, rot zu vermeiden und eher ein „tiefes dunkelorange“ noch als gelb darzustellen.

Grau bedeutet, dass die Ampel ausgeschaltet oder kaputt ist. Es bedeutet, dass der Indikatorwert nicht ermittelt werden konnte. So kann z. B. die Qualitätsbewertung gar nicht vorgenommen werden, wenn man mit der Testdurchführung, obwohl sie für den entsprechenden Zeitabschnitt ursprünglich vorgesehen war, noch nicht begonnen hat. Meistens gelten die für diesen Fall bestimmten Ersatzregelungen.

Eine detailliertere Form der Projektampel (siehe Abb. 3.1) nutzt 9 Stufen, also 1–3 für rot, 4–6 für gelb und 7–9 für grün. Nach meiner Erfahrung sind diese 9 Stufen erforderlich, da die Unterteilung nach rot – gelb – grün nicht fein granular genug ist.

Es sollte für alle Projekte der Organisationseinheit allgemein gültig dokumentiert werden, welche Kriterien im Einzelnen dazu führen, dass eine Ampel rot, gelb oder grün ist. Die Interpretationen laufen sonst innerhalb des Unternehmens zu sehr auseinander.

Testfortschritt und Qualität sind regelmäßig zu bewerten. Die Kosten sind in der Regel durch ein Budget von vornherein festgelegt. Das Testbudget wirkt sich wiederum auf die Testziele aus: wenn von Anfang an klar ist, dass mit den geplanten Kosten nur eine stark eingeschränkte Testabdeckung möglich ist und auf bestimmte Testaktivitäten daher verzichtet wird, so muss das im Testkonzept beschrieben sein.

Dazu werden beim Reporting Trendkurven (nach oben, nach unten, gleichbleibend) dokumentiert, die den Verlauf des Fortschritts oder der Qualität ausdrücken. Eine steigende Trendlinie ist dadurch bei Stufe 9 und eine fallende Kurve bei Stufe 1 nicht möglich.

1		Produktziel nicht mehr erreichbar. Gesamtprojekt gefährdet.
2		Produktziel nicht erreichbar. Auswirkungen in Teilbereichen.
3		Produktziel nicht erreichbar. Maßnahmen nicht vorhanden oder greifen nicht.
4		Produktziel kann mit eingeleiteten Maßnahmen erreicht werden. Risiken vorhanden.
5		Produktziel wird durch eingeleitete Maßnahmen erreicht. Maßnahmen überprüfen.
6		Produktziel wird mit abgesicherten Maßnahmen erreicht.
7		Produktziel wird erreicht. Maßnahmen nicht notwendig.
8		Produktziel ist erreicht.
9		Produktziel ist übertroffen.

Abb. 3.1 Projektampel

Dieses Reporting ist wöchentlich vorzunehmen. Eine tägliche Einschätzung ist zu umfangreich und zeigt zu viel Verzerrungen, eine monatliche Auswertung des Teststatus lässt zu viel Zeit verstreichen, um bei Problemen rechtzeitig genug gegensteuern zu können. Teilweise sind Lenkungskreise in monatlichem Abstand angesetzt, zu denen das Reporting zu einer dann unmittelbar vergangenen Woche präsentiert werden sollte. Auch Präsentationen vor einem größeren Gremium sind von langer Hand vorzubereiten, sodass man bei Bedarf nur noch die aktuellen Zahlen einfügen muss. Ansonsten muss man zu einem Zeitpunkt, in dem sich das Projekt sowieso schon in einer arbeitsintensiven Phase befindet, zusätzlich Überlegungen anstellen, wie man seine Ergebnisse sinnvoll vorstellt.

Das Bewertungsschema sollte unternehmensweit gelten und im Testkonzept eines einzelnen Projektes lediglich referenziert werden bzw. nur beschrieben sein, wenn davon aus bestimmten Gründen abgewichen wird.

Bei den Testzielen ist auch darauf zu achten, die Verantwortlichkeiten für die Testdurchführung und die **Qualitätsziele** exakt festzulegen. Das gilt vor allem in dem Fall, dass einzelne Komponenten von anderen Lieferanten geliefert werden: Liefert der Zulieferer einen Testbericht? Welcher Qualitätslevel ist vereinbart? Wie umfangreich waren die Tests des Zulieferers? Ist die einzelne Komponente in sich lauffähig? Welche Fehler sind beim Zulieferer bekannt? Um diese Bewertung vornehmen zu können, empfiehlt es sich, den Zulieferer ebenfalls entsprechende Indexwerte vorzulegen, um eine möglichst neutrale Bewertung vornehmen zu können. Leider enthalten vertragliche Vereinbarungen mit Lieferanten meist keine eindeutigen oder nicht ausreichenden Festlegungen über Qualitätslevel und Testumfang.

Tab. 3.1 Teststufen und Testinhalte

Thema	Teststufe	Test
Antwortzeit	Lasttest	Belastung des Systems mit 500.000 Anfragen innerhalb einer Stunde
Datenbankzugriff	Integrationstest	Test der Integration Komponente A und B mit Datenbank D
Abrechnungsdatensatz	Systemtest	Prüfung der korrekten Satzarten und Felder
Drucklayout	Systemtest	Kontrolle des Layouts der Ausgabedokumente: – Anordnung des Logos – Kopfzeile – Text (Rechtschreibung, Zeilenumbrüche, Anordnung) – Seitenumbrüche

Es empfiehlt sich, im Kapitel über Testziele die Themen und Testarten in einer Tabelle zu dokumentieren, zum Beispiel wie in Tab. 3.1.

Dabei sollte man auf die Testspezifikation bzw. das Tool, in dem sie erfasst wird, referenzieren. Die einzelnen Themen, die im Testkonzept behandelt werden, müssen sich dort wiederfinden.

Literatur

[ENZY2019] : <https://www.enzyklo.de/>, zugegriffen am 09.05.2020

[PRMA2019] : https://www.projektmagazin.de/artikel/testmanagement-it-projekten-teil-1_7180, zugegriffen am 09.05.2020

[SPIL2007] : Andreas Spillner, Tilo Linz, Basiswissen Softwaretest, 2007 dpunkt Verlag Heidelberg

[WALL2011] : Ernest Wallmüller, Software Quality Engineering, 2011 Carl Hanser Verlag München

Zusammenfassung

Die Testplanung legt fest, welche Arbeitspakete in welcher Reihenfolge von wem zu erledigen sind, plant die Testressourcen und stellt Abhängigkeiten dar. Zur Testplanung sollen professionelle Tools eingesetzt werden. Geeignete Testmethoden und eine realistische Testaufwandsschätzung, die Beachtung von Zeit- und Kostenrestriktionen und eine Bewertung der Testaufgaben sind im Rahmen der Testplanung vom Testmanager zu leisten.

4.1 Inhalte der Testplanung

Die **Testplanung** ist ein Teilbereich der Projektplanung. Innerhalb softwareentwickelnder Organisationen wird Software überwiegend in Form von Projekten entwickelt. Da auch die Testplanung und Testaufwandsermittlung Aufgaben im Rahmen der Projektdurchführung sind, erweist sich das Projekt als relevante Betrachtungsebene dieser Arbeit. Projekte zeichnen sich durch die Einmaligkeit der Bedingungen aus, unter denen die gestellte Aufgabe gelöst wird, und weisen gemäß **Norm DIN 69901** folgende Merkmale auf:

- definiertes Ziel,
- zeitliche Befristung,
- relative Neuartigkeit und Komplexität,
- Risikobehaftung und
- interdisziplinäre Aufgabenstellung [[DOWI2009](#)].

4.2 Teilpläne für IT-Projekte

Für IT-Projekte werden gewöhnlicher Weise folgende Teilpläne erstellt (Tab. 4.1).

Diese Übersicht gilt im Grunde auch für Teilprojekte für Test und Qualitätssicherung. Die einzelnen Kriterien sind dabei im Testkonzept zu berücksichtigen.

Tab. 4.1 Teilpläne des Projektplans

Planungsgegenstand/Ergebnisse	Fragestellungen, die durch die Teilpläne beantwortet werden können
Projektstruktur: Projektstrukturplan (PSP)	• Welche Aktivitäten sind für das Erreichen der Projektziele nötig? (Was ist im Einzelnen zu tun?) • Welcher Zusammenhang besteht zwischen den Projektaufgaben? (Hierarchie der Teilaufgaben bilden) • Wie lassen sich Objekte und Tätigkeiten für das Projekt in Arbeitspakete aufteilen?
Projektablauf: Projektablaufplan	• In welchen Phasen soll das Projekt abgewickelt werden? • In welcher Reihenfolge sind die Aktivitäten zu erledigen? Wo gibt es logische Abhängigkeiten? • Welche Arbeitspakete können parallel ausgeführt werden?
Projekttermine (Zeiten): Terminplan	• Wie hoch wird der Zeitaufwand für die jeweiligen Aktivitäten geschätzt? • Bis wann ist welche Aktivität zu erledigen? Sind bestimmte Meilensteine zu beachten?
Ressourcenbedarf, Ressourcenkapazität, Ressourceneinsatz: Ressourcenpläne	• Wie viel Personal wird für die einzelnen Arbeitspakete benötigt? Welcher Sachmittelbedarf ist gegeben? • Welche Kapazitätsbegrenzungen bezüglich der Ressourcen sind zu berücksichtigen (Kapazitätsplan)? • Wann ist die Verfügbarkeit der Ressourcen notwendig (Ressourcen-Einsatzplan)?
Kosten und Projektfinanzen: Kosten- und Finanzplan	• Welche Kosten(-arten) fallen bei welchen Arbeitspaketen an? Wie hoch sind die Gesamtkosten für das jeweilige Arbeitspaket? • Was wird das Projekt insgesamt und ungefähr kosten? Lässt sich die Summe durch eine Addition der Arbeitspaketkosten (sog. direkte Kosten) und der Gemeinkosten ermitteln? • Wie hoch sind die anfallenden Personalkosten? • Wie erfolgt die Freigabe der Finanzmittel? (Zeitpunkt, Voraussetzungen)
Qualität: Qualitätssicherungsplan (QS-Plan)	• Welche Ergebnisse sollen in welcher Qualität/Form verarbeitet werden? • Welche qualitätssichernden Maßnahmen sollen im Projektverlauf ergriffen werden? (Projektinterne Richtlinien, QS-Schulungsmaßnahmen, Reviews, Tests)
Risikomanagement: Risikoplan	• Welche Projektrisiken sind denkbar und wie groß ist die Wahrscheinlichkeit ihres Eintretens? • Welche Auswirkungen sind im Risikofall möglich? • Welche vorbeugenden Maßnahmen zur Risikominimierung können ergriffen werden (Prioritätsplan der Risiken)?

(Fortsetzung)

Tab. 4.1 (Fortsetzung)

Planungsgegenstand/Ergebnisse	Fragestellungen, die durch die Teilpläne beantwortet werden können
Projektorganisation: Organisationsplan, Berichtsplan	• Welche Rollen sind in dem Projekt gegeben? • Welche Aufgaben, Kompetenzen und Befugnisse sind jeder Rolle zuzuordnen und wie ist das Zusammenspiel zwischen den Projektrollen geregelt? • Wie wird die Kommunikation zwischen Team, Auftraggeber und weiteren Beteiligten geplant? • Wie soll das Berichtswesen organisiert werden? • Wie wird das Projekt dokumentiert?

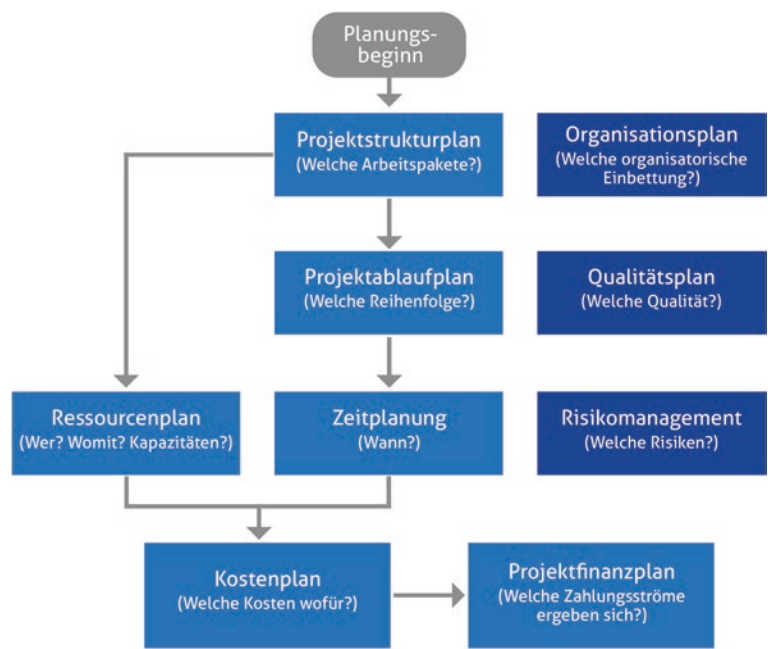


Abb. 4.1 Teilaufgaben bei der **Projektplanung**

Der Zusammenhang zwischen den Teilaufgaben bei der Planung von IT-Projekten kann durch Abb. 4.1 illustriert werden:

Die grobe Testplanung sollte mit einem professionellen Planungstool (einer Projektverwaltungssoftware wie z. B. Microsoft Project) durchgeführt werden. Leider wird eine kostenpflichtige Lizenz für derartige Tools in vielen Projekten gespart. Durch diese Art Sparpolitik steigen in der Regel die Projektkosten („wir sparen, koste es was es wolle“). Wenn man die Termine und Abhängigkeiten im Testmanagement nicht konsequent im Blick hat und bei Verzögerungen sofort gegensteuert, übersieht man schnell Probleme und erkennt Warnsignale zu spät.

Tab. 4.2 Übersicht der Stakeholder im Projekt

Verantwortung	Abteilung	Mitarbeiter	Telefon-Nummer
Projektleitung	XX1	Müller	11
		Vertreter: Meier	12
Systemtest Komponente A	XX2	Schmidt	15
		Vertreter: Huber	16
Systemtest Komponente B	XX2	Huber	16
		Vertreter: Schmidt	15
Integrationstest	XX3	Bauer	21
		Vertreter: Lehmann	22
...	...	...	...

Man kann kleinere Projekte noch mithilfe von Excel managen; dieses Vorgehen ist aber eindeutig nur sehr begrenzt möglich. Schon die Nichtverfügbarkeit von professionellen Planungstools gehört zu den ungünstigen Rahmenbedingungen, die im Testkonzept Erwähnung finden sollten. Zu beachten ist aber auch, dass das Tool konsequent eingesetzt werden sollte. Wenn man nur die Rahmendaten einmalig im Planungstool erfasst und das Tool nur zu Dokumentationszwecken einsetzt (auch das kommt in der Praxis leider immer wieder vor!), ist der Sinn des Tools ad absurdum geführt.

Für eine Übersicht der einzelnen Testphasen für die Stakeholder empfiehlt sich eine Darstellung wie in Tab. [4.2](#)

4.3 Testvoraussetzungen

Eine systematische Testplanung setzt gewisse Informationen voraus, ohne die eine genaue Planung nicht möglich ist. Wer die einzelnen Testaktivitäten genau planen will, muss wissen, welche und wie viele Testfälle zu testen sind. Diese Informationen lassen sich bei neuen Systemen aus den Anforderungen und bei bestehenden Systemen aus der Systemdokumentation bzw. der Codeanalyse ableiten. Darum ist die Anforderungsanalyse eine unbedingte Voraussetzung für die Testplanung und muss vor der Testplanung stattfinden. In der Praxis wird aber leider die Anforderungsanalyse viel zu stiefmütterlich behandelt: Die Unterlagen für das Angebot liegen erst beim Vertrieb, der Entwickler bekommt sie vielleicht zwei Tage zur Ansicht aber hat oft nur die Zeit, sie zu überfliegen und prüft gar nicht im Detail, was auf die Programmierung zukommt. Oft wird sie „nebenbei“ erledigt, weil das laufende Projekt noch nicht abgeschlossen ist und das zukünftige noch nicht terminlich drängt. Während des Projektes entstehen dabei zusätzliche Zeiten und Aufwände, sodass für die Anforderungsanalyse des Folgeprojektes wieder nicht genügend Zeit eingeplant wird. Weil viele Projekte mit Verspätung enden, sind Ressourcen gebunden und man kommt auch im übernächsten Projekt nicht dazu usw. usw. Auf einen Workshop mit Auftraggeber und Entwicklung wird leider häufig „aus Zeitgründen“ verzichtet. In der Praxis wird oft nach dem Motto „Wir haben nicht die Zeit, die Dinge einmal richtig zu machen, aber wir haben die Zeit, sie drei Mal falsch zu machen“ verfahren.

Beim Komponententest ist es erforderlich, erst den Source-Code der Komponente zu analysieren. Um die Testziele abzustecken. Vor dem Integrationstest muss erst die Systemarchitektur analysiert werden. Beim Systemtest ist es das Konzept bzw. die Anforderungsspezifikation, die vorher zu analysieren ist, um den Umfang des Systemtests abzustecken.

In den agilen Vorgehensmodellen werden Planung und Ausführung in kurzen Iterationen ausgeführt. Ein **agiles Projekt** ist per Definition ein nichtdeterministischer Prozess. Der Umfang und Inhalt können sich während der Entwicklung noch ändern, und keiner kann am Anfang genau sagen, wo die Entwicklung endet, wie lange sie dauert und wie viel sie kostet. Wenn hingegen in klassischen Projekten die Anforderungen im Detail ausgearbeitet und modelliert wurden, wird die Information für eine systematische Planung ausreichen.

Zur Definition der Testziele – des „Warum“ – werden Größenmaße des Systems, z. B. die Größe in **Function Points**, **Object Points**, **Test Points** oder Anweisungen und die Fehlerrate vergangener oder ähnlicher Projekte benötigt. Für neue Systeme müssen die Größenmaße aus der Anforderungsanalyse gewonnen werden. Für existierende Systeme, die ersetzt, saniert, migriert oder integriert werden sollen, ist es (falls keine Beschreibung vorhanden ist) notwendig, das alte System erst durch Reverse Engineering nachträglich zu dokumentieren und anschließend die Größenmaße aus der Nachdokumentation zu gewinnen. Das ist ein zeitaufwändiger und fehleranfälliger Prozess. Die Fehlerrate bisheriger Projekte lässt sich aus einer Analyse der Fehlerdatenbank ermitteln. Ohne diese Messwerte ist es kaum möglich, messbare Testziele zu definieren.

4.4 Rahmenbedingungen der Testplanung

Die Bestimmung der Testtermine hängt von den Lieferterminen ab. Die Liefertermine werden im Allgemeinen im Testplan festgelegt. Demzufolge setzt die Testplanung die allgemeine Projektplanung voraus, der Testplan baut auf dem Projektplan auf.

Die Definition der Testörtlichkeit setzt Kenntnisse der Verfügbarkeit voraus. Der Testmanager muss zum Zeitpunkt der Testplanung bereits wissen, mit welcher Hardwareausstattung getestet wird und ab wann diese Geräte verfügbar sind. Die Entscheidung, den Test zu verteilen oder zentralisiert durchzuführen, hängt u. a. von der Verfügbarkeit der Rechnerressourcen ab.

Die Wahl der richtigen **Testmethode** setzt voraus, dass der Testmanager die Art der Anwendung kennt. Er muss wissen, worauf es ankommt, wo die Risiken liegen und wo die meisten Fehler auftauchen können. Nur mit diesen Kenntnissen ist er in der Lage, die geeignete Vorgehensweise auszuwählen und diese wiederum dem Projekt anzupassen. Darüber hinaus muss der Testplaner mit den gängigen Systemtestmethoden vertraut sein. Sonst wird er nicht wissen, welche Ansätze überhaupt zur Auswahl stehen.

4.5 Testwerkzeuge

Zum zielgerichteten Einsatz von **Testwerkzeugen** muss man wissen, welche Werkzeuge zur Verfügung stehen und für welche Aufgabe diese Tools (z. B. Auswertetools zur Messung der Testabdeckung, Testfallgeneratoren) geeignet sind. Denn im Gegensatz zum Entwurf eines neuen Systems, bei dem meist ein Werkzeug ausreichend ist, braucht man für die Testdurchführung für jede Art von Testaktivitäten ein eigenes Werkzeug. Für die Testplanung werden daher Informationen zu Funktionalität und Qualität der infrage kommenden Werkzeuge und die notwendige Zeit zur Einarbeitung. Zur Verteilung der Testarbeit auf die richtigen Mitarbeiter muss man Tester und Testaufgaben genau kennen, um die verfügbaren Mitarbeiter optimal auf die Testaufgaben verteilen zu können. Auch der Test selbst verlangt unterschiedliche Schwerpunkte, z. B. auf Gebieten der Skriptprogrammierung, Methodik der Fachlichkeit. Eine gute Koordination des Testmanagers kann die Produktivität in diesem Bereich erheblich steigern.

4.6 Testaufwände und Testautomatisierung

Für die **Testaufwandsschätzung** sind meist vergangenheitsbezogene Daten erforderlich. Dabei geht es um die Fehlerrate (Anzahl gefundener Fehler in den bisherigen Projekten) und die Fehlerdichte als Anzahl der Fehler pro 1000 oder 100 Größeneinheiten, die sich als Maß auf das neue System übertragen lässt. Die Größeneinheit können Codezeilen, Anweisungen, Function Points oder auch Testfälle sein. Dieses Maß muss durch den Grad der Testabdeckung relativiert werden. Es genügt also nicht, nur die Anzahl der absoluten Fehler zu kennen, sondern man muss auch die Größe der betroffenen Software und den Grad der Testabdeckung dazu berücksichtigen.

Tester müssen möglichst viele effektive **Testfälle** entwickeln, um mögliche Fehlerquellen aufzuspüren. Wie sehr ihnen das gelingt, ist von ihrer Produktivität abhängig. Beim Systemtest ist die Produktivität ein Maß für die Anzahl der Testfälle, die ein Tester pro Zeiteinheit zu leisten in der Lage ist. Diese Zahl variiert von Projekt zu Projekt und hängt stark vom Grad der Testautomatisierung als auch von der Erfahrung und Kreativität der Tester ab. Dennoch lassen sich künftige Aufwände nur aufgrund der bisherigen Produktivität vorhersagen. Dafür benötigt man zur Testplanung entsprechende Erfahrungswerte, aus der das Verhältnis geleisteter Testertage zur Anzahl der ausgeführten Testfälle hervor geht [SNEE2012].

Zur Bewertung der **Testautomatisierung** ist vorab zu definieren, wie hoch der Automatisierungsgrad für Testfallerstellung, Testdurchführung und Testreporting ist. Oft wird nur einseitig die Testdurchführung betrachtet – gerade die Vorbereitung und Planung der Testläufe und die Auswertung der Testergebnisse benötigen aber auch bei automatisierten Tests einen Aufwand, der häufig unterschätzt wird.

Literatur

- [SNEE2012] : Sneed, Baumgartner, Seidl: Der Systemtest, Carl Hanser Verlag München 2012
[DOWI2009] : Ulrike Dowie, Testaufwandsschätzung in der Softwareentwicklung, Josef-Eul-Verlag Köln 2009

Bezeichnung des Testkonzeptes und Einleitung

5

Zusammenfassung

Für Testkonzepte, ob sie generisch für das gesamte Unternehmen oder projektbezogen erstellt werden, sind einige allgemeine Festlegungen und die korrekte Versionierung des Dokuments zu beachten. In der Einleitung zu projektbezogenen Testkonzepten ist auf die Notwendigkeit des entsprechenden Projektes einzugehen. Relevante Projektdokumente müssen im Testkonzept referenziert werden. Agile Projekte weisen einige Besonderheiten auf, die schon in der Phase der Konzepterstellung berücksichtigt werden müssen. Die Erstellung des Testkonzeptes ist am Anfang der Testphase durchzuführen.

5.1 Bezeichnung und Ablage des Testkonzeptes

Jedes Testkonzept muss eine eindeutige Bezeichnung haben (am besten eine ID), damit es auch nach den abgeschlossenen Testaktivitäten wieder auffindbar ist. Eine Versionierung ist ebenfalls vorzunehmen. Wenn das Testkonzept mithilfe einer Vorlage erstellt wurde, muss ein Deckblatt vorhanden sein, das vollständig ausgefüllt wurde.

Dabei ist auch zu untersuchen, wo das Testkonzept abgelegt wird, ob z. B. auf das entsprechend Laufwerk alle Stakeholder Zugriff haben. Die Tester sollten sich daran gewöhnen, für offene Fragen zunächst das Testkonzept zu Rate zu ziehen.

5.2 Testkonzepte für einen längeren Zeitraum

Manches Mal zieht sich ein Projekt über einen längeren Zeitraum. In diesem Fall macht es evtl. Sinn, einzelne Testkonzepte für unterschiedliche Releases oder Versionen zu erstellen. Generell sollte man bei der Erstellung von Testkonzepten einen modularen Aufbau (Abb. 5.1) wählen:

Das Testkonzept beschreibt Umfeld, Aktivitäten und Vorgehensweise zu den Tests des beschriebenen Projektes. Dabei konkretisiert es die im Dokument, das für die übergreifende Teststrategie des Gesamtunternehmens erstellt wurde, beschriebenen Prozesse und Richtlinien. Eventuelle Abweichungen werden hier beschrieben und begründet.

Das Testkonzept baut dabei auf Informationen aus anderen Bereichen auf, die zusammenwirken. Durch den modularen Aufbau (Abb. 5.1) werden Redundanzen vermieden und der Leseaufwand entspricht dem jeweiligen Informationsbedürfnis.

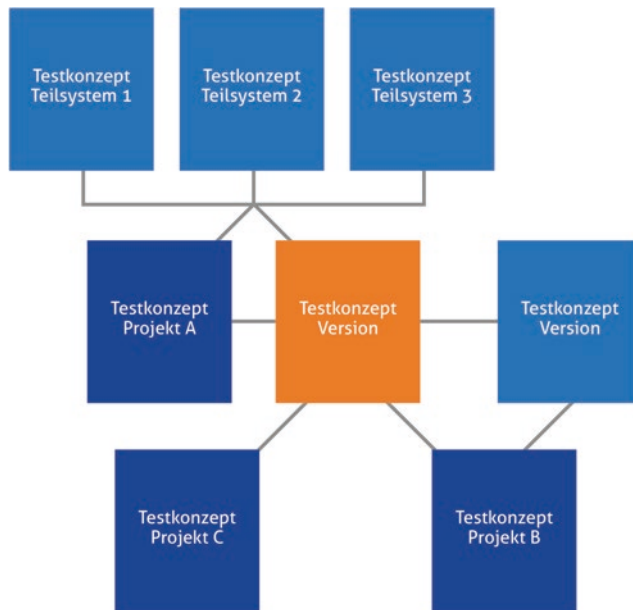


Abb. 5.1 Modularer Aufbau der Testkonzepte

5.3 Formale Grundlagen

Das **Review** des Testkonzeptes sollte idealerweise vor Beginn der Testphasen stattfinden. Dabei sollte es mit einer Version versehen werden (z. B. Version 2.1 vom 06.05.2020).

Durch diese **Vers ionierung** liegt zu Beginn der Testaktivitäten eine definierte Version vor.

Man muss (z. B. im Rahmen eines Audits) jederzeit aussagefähig sein, welche Version des Testkonzeptes zu einer definierten Testphase bzw. einem bestimmten Zeitpunkt gültig war.

Beispiel:

Version 1.0: 10.03.

Version 2.0: 03.04.

Version 2.1: 06.05.

Vom 20.04. bis 30.04. fand der Test von Release 1.0 statt, daher ist die Version 2.0 vom 03.04. das für diese Version zu beachtende Testkonzept.

Mit der Bezeichnung grenzt man bereits das Thema und den Testgegenstand ein. Im Unternehmen ggf. geltende Namenskonventionen für die Bezeichnung sind dabei zu beachten. Auf jeden Fall muss die Bezeichnung eindeutig sein und mit der Bezeichnung des Projektes korrespondieren. Dabei ist auf Details zu achten: Wenn das zu testende Projekt „Einführung 3-D Secure“ heißt, darf man das Testkonzept nicht „Testkonzept Einführung 3D-Secure“ bezeichnen, sondern eben folgerichtig „Testkonzept Einführung 3-D Secure“. Bei Begutachtungen oder Audits können solche Details sonst zu ernsthaften Problemen führen. Wie soll man einem Testmanager glauben können, dass er die Qualität sichert, wenn er in seinen Aussagen nicht 100 % exakt ist?

Bei der Erstellung des Testkonzeptes ist auf eine klare Struktur und gute, übersichtliche Gliederung zu achten. Das Testkonzept sollte Tabellen und Grafiken enthalten und für alle Stakeholder des Projektes gut verständlich sein. Das Testkonzept sollte ein Inhalts-, Abkürzungs-, Definitions- und Quellenverzeichnis enthalten. Es ist darauf zu achten, dass das Inhaltsverzeichnis mit der Gliederung des Dokuments übereinstimmt. Alle nicht allgemein bekannten Begriffe und Abkürzungen müssen definiert oder innerhalb des Textes erklärt werden. Die Seiten des Testkonzeptes sollen nummeriert sein und eindeutig als der Dokumentversion zugehörig identifiziert werden können. Das Dokument muss vollständig sein, es dürfen also keine Seiten, Textstellen oder Abbildungen fehlen, Querverweise müssen eindeutig sein und Referenzdokumente aufgelistet sein. Bei der Verwendung von Bildern und grafischen Darstellungen ist darauf zu achten, dass sie den Text sinnvoll ergänzen, übersichtlich und nicht überladen wirken, verständlich bezüglich der verwendeten Symbole sind und den entsprechenden Kapiteln oder Textstellen eindeutig zugeordnet sind. Allgemeine firmeninterne bzw. projektespezifische Richtlinien zur Unterlagenerstellung müssen eingehalten werden. Methoden, Werkzeuge und Standards sind dabei zu berücksichtigen. Vertraulichkeitsstufen und Urheberrechte müssen ausgewiesen und beachtet werden.

5.4 Einleitung Testkonzept

In der Einleitung des Testkonzeptes soll das Projekt beschrieben sein: Was ist der Testgegenstand, um den es geht? Welche Features enthalten das Produkt, und welche davon sollen getestet und welche sollen nicht getestet werden? Welche einzelnen **Testobjekte** sind enthalten?

Dazu empfiehlt sich, eine Zusammenfassung der fachlichen Beschreibung aufzuführen, also eine kurze Vorstellung des entsprechenden Projektes. Die Darstellung des Projekthintergrunds verhilft den Stakeholdern dazu, dass sie den Inhalt des Testkonzeptes besser einordnen können. Aus dieser Darstellung sollten die Hintergründe für die Aufgabenstellung und der Nutzen klar ersichtlich sein, damit sich alle Beteiligten optimal mit dem Projekt identifizieren können.

Was ist der Grund für die Notwendigkeit der Durchführung des Projektes, eine neue Technologie, eine Verbesserung des Workflows, gesetzliche Anforderungen? Mit der Bezeichnung grenzt man bereits das Thema und den Testgegenstand ein. Es soll dabei auch definiert werden, warum das Projekt durchgeführt werden soll. Begründungen für die Notwendigkeit der Projektdurchführung können sein:

- vertragliche Verpflichtungen (z. B. aus Ausschreibungen, Verträgen),
- gesetzliche Regelungen (z. B. EU-Richtlinien, Vorgaben von Aufsichtsbehörden),
- Folgen unternehmerischer Entscheidungen oder Unternehmensstrategie (z. B. Konzernstrategien, Kundenqualitätsoffensive, Kostensenkungsmaßnahmen, Vertriebsvorhaben),
- technologische Notwendigkeit (z. B. notwendige Ablösung von Altsystemen),
- Abhängigkeiten zu vorherigen Vorhaben (z. B. Projekt setzt auf Fachlichkeiten auf, die ein Vorprojekt bereitstellt),
- Vorhaben ist Voraussetzung zu Folgevorhaben (z. B. Projekt schafft Fachlichkeiten, auf die ein bereits geplantes/paralleles/späteres Projekt aufsetzt, das Projekt dient als „Enabler“ für weitere Vorhaben).

Zusätzlich sollte in diesem Kapitel beschrieben werden, wann der Test beendet ist bzw. wann er abgebrochen wird. In der Regel ist das Testende ohnehin durch Kosten und Terminrestriktionen begrenzt; man kann also nicht unbegrenzt weiter testen, um eventuell doch noch Fehler zu finden.

Die **Fehlerfindungskurve** folgt normalerweise ungefähr folgendem Verlauf (Abb. 5.2):

Man sieht dabei, dass ab einem bestimmten Punkt die Produktivität des Testteams sinkt und kaum noch neue Fehler gefunden werden. Ein erfahrener Testmanager kann diesen Punkt in etwa abschätzen, auch Erfahrungswerte von früheren Projekten kann man dabei heranziehen. Man kann aber auch detaillierte Planungen durchführen, die den Testfortschritt angesichts von der Testtiefe und der Umsetzung der Anforderungen (erworbene „**Testpunkte**“) abhängig machen. Das ist aber ein aufwändigeres Verfahren. In der Praxis

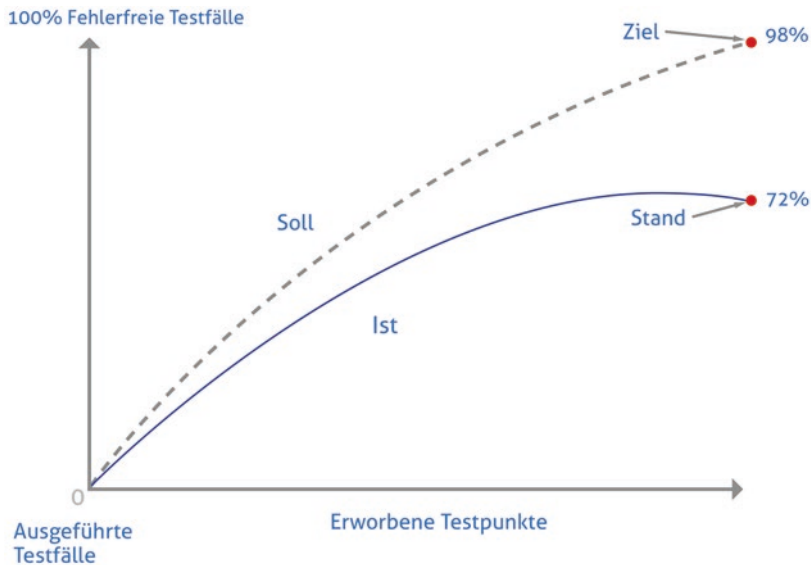


Abb. 5.2 Fehlerfindungskurve

ist die Zeit, die zum Test zur Verfügung steht, ohnehin vom Projektplan begrenzt, daher ist die Frage nach dem sinnvollen Ende der Testaktivitäten eher akademisch.

In der Einleitung des Testkonzeptes soll auf das Testziel eingegangen werden: Was soll mit dem Test erreicht werden? Welche Risiken sollen vermieden werden? Welcher Qualitätsanspruch liegt dem Projekt bzw. der Organisation zugrunde? Daraus leitet sich auch ab, in welcher Tiefe der Test durchgeführt werden muss. Welche Normen und Richtlinien sind beim Test zu beachten? Ist das Testziel eine Zertifizierung nach einer bestimmten Norm oder Richtschnur zu bestehen?

Im Testkonzept sollen in einem einleitenden Kapitel auch Kriterien für die **Testunterbrechung** und den **Testwiederanlauf** dokumentiert werden. Wenn die Anwendung gravierende Fehler enthält, z. B. eine erheblich zu große Antwortzeit oder wichtige Basisfunktionen nicht vorhanden sind, kann es sinnvoller sein, den Test zu unterbrechen und erst nach Fehlerbehebung wieder aufzunehmen. Die Bedingungen dafür (z. B. Fehler mit Priorität „Kritisch“ oder testverhindernde Fehler, also wenn zum Beispiel die zu testende Anwendung gar nicht gestartet werden kann) sind ebenfalls in der Einleitung des Testkonzeptes zu definieren. Eine Testunterbrechung kann aber dazu führen, dass man seine für das eigene Projekt vorgesehenen Ressourcen an ein anderes Projekt abgeben muss. Wenn die Software nach einiger Zeit wieder testbereit und die Unterbrechung beendet ist, kann es sein, dass man die Tester nicht mehr zurückbekommt, weil sie dann im anderen Projekt gebunden sind, und dadurch weitere Verzögerungen im Projekt einkalkulieren muss. Besser ist es also, wenn man die Zeit der Testunterbrechung eher für testbezogene Projektarbeiten wie z. B. eine Verbesserung des Grades der Testautomatisierung verwendet.

5.5 Im Testkonzept referenzierte Dokumente

An dieser Stelle im Testkonzept sollte auch auf die für den Test relevanten **Dokumente** hingewiesen werden:

- Spezifikationen, Grobkonzept, Fachfeinkonzept,
- technisches Design, Entwurfsdokumente,
- Benutzerhandbücher, Betriebshandbücher, Bedienungsanleitung sowie
- Installationshinweise.

Hier empfiehlt es sich für den Testmanager, die vorliegenden Dokumente zumindest grob zu sichten und ihre Qualität zu prüfen. Zunächst einmal ist zu prüfen, ob es die Dokumente schon gibt oder sie noch in der Entstehung sind. Weiterhin sind gerade der Detaillierungsgrad der Spezifikationen und Konzepte und die Eindeutigkeit der **Requirements** entscheidend für den Testerfolg.

Wenn man bereits an dieser Stelle bemerkt, dass Anforderungen fehlen, widersprüchlich oder unklar sind (was erfahrungsgemäß leider in den meisten Projekten der Fall ist), kann man schon hier ein erhebliches Testrisiko formulieren. Der Test kann nur so gut sein, wie die Voraussetzungen und Rahmenbedingungen es zulassen. Wenn man bei der Erstellung der Einleitung des Testkonzeptes schon bemerkt, dass dazu erheblicher Handlungsbedarf besteht, kann man hier bereits Maßnahmen proaktiv vorschlagen, um das Projekt nicht scheitern zu lassen. Sollte man nicht allzu spät mit sämtlichen Testaktivitäten begonnen haben (auch das geschieht nach wie vor viel zu oft!), kann man zumindest noch rechtzeitig warnen, welche Probleme in naher Zukunft zu erwarten sind.

Es ist auch immer wieder festzustellen, dass zu schnell mit der Entwicklung gestartet wird. Das hat auch einen guten Grund, denn man will ja möglichst kurzfristig vorzeigbare Ergebnisse präsentieren können. Das ist besonders dann der Fall, wenn ein bestehendes Vorprodukt technologisch erweitert oder geändert wird. Man investiert gerne in die Vorführung eines Prototyps, weil man sich dadurch plastisch etwas unter dem Endprodukt vorstellen kann und Kunden oder Auftraggeber einen besseren Eindruck vom Zielzustand bekommen. Meist wird dabei erst später im Projektverlauf in Testaktivitäten investiert, weil man bemerkt, dass viele Fehler im Produkt enthalten sind, die es zu verbessern gilt. Dann sind aber nach wie vor die Aufgabenstellung und Konzepte lückenhaft, und das wird beim Test nach und nach bemerkt. Häufig ist auch zu beobachten, dass die Entwicklung die Designphase längst überholt hat und die Systemarchitektur nachträglich dokumentiert, was sowieso entwickelt wurde, aber nicht die Basis von vornherein festlegt, wie es in klassischen Modellen nach dem V-Modell umgesetzt werden sollte. Der agile Ansatz hat wiederum das Problem, dass die Entwicklung noch schneller starten soll und die einzelnen Projektphasen beschleunigt werden sollen, aber dabei häufig zu wenig auf die korrekte Definition der Anforderungen geachtet wird.

Klassisch / Wasserfall



Agil / Iterativ (neu)

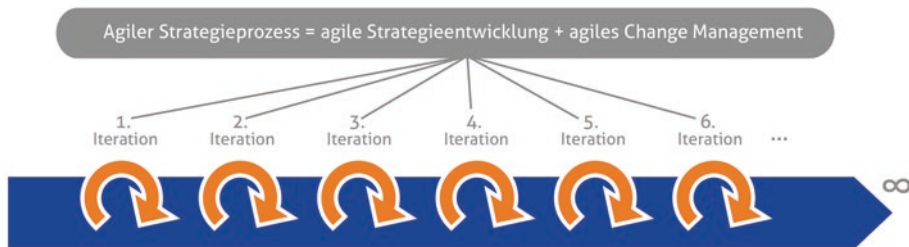


Abb. 5.3 Strategieprozesse im klassischen und im agilen Modell

Das ist kein Problem des **agilen Vorgehensmodells** an sich, sondern einer falschen Umsetzung des agilen Modells, das in vielen Organisationen beobachtet werden kann. Man schreibt sich „agil“ auf die Fahne, ohne die Bedeutung und die Konsequenzen hinreichend beachtet zu haben, und betreibt damit im Grunde Etikettenschwindel. Eine der Prinzipien agiler Prinzipien lautet, dass die besten Architekturen, Anforderungen und Designs in selbstorganisierten Teams entstehen. Es steht aber nirgends, dass man darauf verzichten kann.

Auch für agile Projekte ist ein Testmanagement entscheidend, die Rolle des Testers verändert sich, löst sich aber keinesfalls auf. Agiles Testen verlangt gute Planbarkeit, Verfolgbarkeit und Nachweisbarkeit der Testaktivitäten, eine hohe Systematik des Testentwurfs und daraus resultierend eine hohe und reproduzierbare Testabdeckung.

Die folgende Skizze (Abb. 5.3) zeigt den Unterschied der **Strategieprozesse** in klassischer und agiler Vorgehensweise:

Im **Release Planning** wird ein komplettes Release geplant. Für das einzelne Projekt wird aber nach wie vor das Testkonzept benötigt. In einem Release werden in der Regel mehrere Projekte umgesetzt. Den Übergang vom klassischen zum agilen Modell zeigt folgende Abbildung (Abb. 5.4).

Das Testkonzept wird dabei nach wie vor benötigt. Jeder Aufgabe der Rolle „Testmanager“ lassen sich Artefakte von Scrum zuweisen (Abb. 5.5).

Im Falle eines abteilungs- oder projektübergreifenden Testkonzeptes sollten strategische Ziele zur Qualitätssicherung, grundlegende Organisationsfestlegungen und die Notwendigkeit eines allgemeinen, übergreifenden Testkonzeptes erläutert werden. In diesem Fall wird die übergreifende Planung, Steuerung und Kontrolle sämtlicher Testaktivitäten des Unternehmens bzw. eines Bereichs im Unternehmen beschrieben, das generisch

Testkonzeption			Testumsetzung			Testkoordination		
Klassisch	Scrum		Klassisch	Scrum		Klassisch	Scrum	
Testkonzept			Teststufenplanung	Test-Pyramide	Planning	Projekt-/Testorganisation	Daily	Backlogs
Teststrategie	Release Planning	Sprint Planning	Testimplementierung	Sprint	Planning	Testzyklusmanagement	Sprint	Releases
Qualitätsmerkmale			Struktur-/Spezifikationsorientierte Verfahren	Test-Pyramide	Test-Automation	Risikoanalyse und -bewertung	Daily	Grooming
Testzyklen und Meilensteine	Sprint	Release			Releasetest		Burn-Down	Retro-spektive
Zeit- und Ressourcenplanung	Plannings	Daily	Komponenten-, Service- und Oberflächentests	Test-Pyramide		Testpriorisierung	Grooming	Planning
Pass-Fail-Kriterien	Definition of Done					Qualitätsgrad-bemessung	Definition of Done	Story
Infrastruktur	Grooming	Planning	Verifikation und Validierung	Story	Sprint Review	Abweichungsmanagement	ZeroBug-Policy	
Dokumentation	Story	Definition of Done				Berichtswesen/ Dokumentation	Board	BurnDown-Chart

Abb. 5.4 Agile Transition des Testmanagers

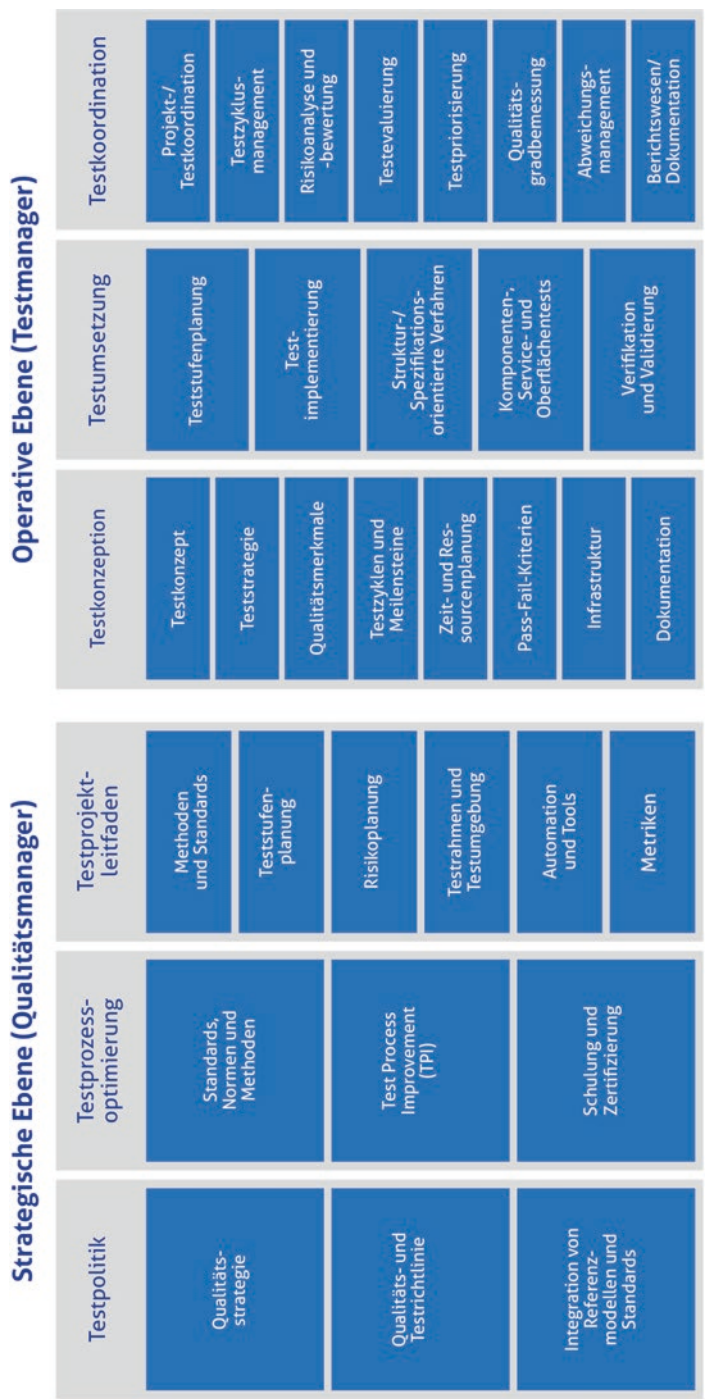


Abb. 5.5 Scrum und Testmanagement [AGIL2016]

für alle Projekte gilt, es sei denn, im einzelnen Projekt werden spezifische Festlegungen getroffen. In der Praxis erlebt man oft, dass dieses Konzept entweder gar nicht bekannt ist, veraltet oder derart allgemein gehalten ist, dass es im täglichen Ablauf nicht weiterhilft. In der Konsequenz steht für die Qualitätssicherung beauftragte Abteilung bzw. der Qualitätsverantwortliche des Projektes als eigene Instanz neben den Projekten und wird von den Entwicklungsprojekten nicht ausreichend wahrgenommen. Für eine Qualitätsabteilung ist es aber eine essenzielle Aufgabe, allgemeine Richtlinien zur Qualitätssicherung auch umzusetzen und an die betrieblichen Abläufe anzupassen. Besondere Schwierigkeiten und Diskrepanzen zwischen Theorie und Praxis werden dadurch evident und können gezielt optimiert werden.

Das Testkonzept soll erlauben, ein Softwareentwicklungsprojekt nachvollziehbar und jederzeit rückverfolgbar zu testen und eine Freigabeentscheidung zu treffen.

5.6 Testrahmen mit Testvoraussetzungen, Fehlerklassen und Startbedingungen

In der Einleitung des Testkonzeptes sollten weiterhin der grobe terminliche Rahmen und die **Testvoraussetzungen** beschrieben sein. Die Bedingungen für den Teststart sollten ebenfalls einwandfrei daraus hervorgehen.

Bei einem Softwaretest in einer entsprechenden Teststufe nach ISTQB (Komponententest, Integrationstest, Systemtest, Abnahmetest) werden viele Fehler festgestellt. Damit der Entwickler weiß, welche Probleme er mit hoher, mittlerer und niedriger Priorität lösen muss, hat es sich als sinnvoll erwiesen, **Fehlerklassen** zu definieren. Beim Abnahmetest hängen an der Fehlerklasse noch entsprechende Regelungen zwischen Auftraggeber und Auftragnehmer. Beispielsweise kann für jeden Fehler der Fehlerklasse 1 eine **Vertragsstrafe** vorgesehen sein. Außerdem ist denkbar, dass ab einer gewissen Zahl von Fehlern der Auftraggeber vom Vertrag zurücktreten kann. Dies kann frei zwischen den Parteien vertraglich ausgestaltet werden.

Im Allgemeinen werden dabei die Fehler in 3 **Fehlerklassen** unterteilt. Fehlerklasse 1 (schwerer Fehler) sind Fehler, die dazu führen, dass die Software nicht genutzt werden kann. Dazu zählen Fehler in zentralen Funktionen, die zum Abbruch der gesamten Anwendung führen und Fehler, welche die Prüfung von weiteren Teilbereichen ausschließen. Für Fehler der Fehlerklasse 2 (mittelschwere Fehler) sind Fehler in der Anwendung, die nicht zum Abbruch führen, aber gleichwohl so erheblich sind, dass ein Echtbetriebsübergang nicht zumutbar ist und nicht mit organisatorischen Mitteln (Tricks zur Umgehung der Softwarefehler = Workaround) umgangen werden kann. Leichte Fehler (Fehlerklasse 3) sind alle Softwarefehler, die nicht der Fehlerklasse 1 oder 2 zugeordnet werden können [BLMS2012].

Das Testkonzept sollte am Anfang der **Testphase** erstellt werden. Wenn es sich beim Testobjekt um eine Software handelt, die lediglich in einer neuen Version entwickelt wurde, in einer ersten Version aber schon vor einiger Zeit eingeführt wurde (etwa, wenn

die neue Version eine erhebliche Funktionserweiterung darstellt), kann man auf ein bestehenden Testkonzept aufsetzen. Es ist aber in der Regel besser, ein neues Dokument zu erstellen. Das hängt aber auch stark vom Umfang der Änderungen oder Erweiterungen ab.

Literatur

- [AGIL2016] : <https://2016.agileworld.de/sites/agileworld/files/agiletestmanagement.pdf>, zugegriffen am 09.05.2020
- [BLMS2012] : <https://blog.milsystems.de/2012/04/fehlerklassengenerator-zur-klassifikation-von-softwarefehlern/>, zugegriffen am 09.05.2020

Zusammenfassung

Für erfolgreiche Projekte ist eine professionelle Organisation erforderlich. In diesem Zusammenhang sind u. a. eine detaillierte Testplanung, die Organisation der Testressourcen, regelmäßige Statusmeetings und Abstimmungen während des Projektes zu beachten. Die Testorganisation bildet einen Teilbereich der Projektorganisation und muss die Strukturen des Projektes und der Organisationseinheit berücksichtigen.

6.1 Wesentliche Kriterien der Testorganisation

Die Professionalisierung des Testens geht mit einer Transformation der heutigen Organisation einher. Dabei kommen neue Servicekonzepte, häufig in Form von sogenannten **Test-Factory-Modellen**, zur Anwendung. Aus dem Testportfolio der Kundenorganisation werden exakt definierte Bereiche herausgeschnitten und danach in Form eines industrialisierten „**Managed-Service-Modells**“ umgesetzt. Die Ausgestaltung der **Test-Factory-Services** sind je nach den individuellen Kundenwünschen vielfältig: Breite der Testverfahren, Skalierbarkeit der Testvolumina, flexible Testinhalte und Abrechnungsmodelle auf Testfallbasis kennzeichnen diese Lösungsmodelle. Darüber hinaus zählen auf den Kundenwunsch zugeschnittene Service-Level-Vereinbarungen zu den wichtigen Bausteinen einer Test-Factory-Implementierung [IODT2019].

Die gewählte Form der **Testorganisation** ist in einem eigenen Kapitel des Testkonzeptes zu behandeln. Dabei sollten die einzelnen Stakeholder mit ihren Rollen definiert und ihre Zuständigkeiten aufgezeigt werden. Die einzelnen Mitarbeiter sollen dabei mit ihrem genauen Zuständigkeitsbereich, Telefonnummer/Adressdaten und ggf. Abteilungsnamen bezeichnet werden. Ggf. sollte dabei die Telefonliste der Organisationseinheit referenziert werden, um mehrfache Datenpflege auszuschließen.

Für einen Testmanager ist es unbedingt erforderlich, über den Status der Testaktivitäten und eventuelle Probleme jederzeit Bescheid zu wissen. Das kann man mit täglichen oder maximal wöchentlichen kurzen Abstimmungsmeetings erreichen. Die einzelnen Besprechungen dürfen kurz sein, es geht oft nur um eine „Wasserstandsmeldung“, aber die Intervalle zwischen den Abstimmungen sollten nicht zu lang sein, daher ist eine Woche Abstand erfahrungsgemäß das Maximum.

Der **Testfortschrittsbericht** mithilfe von verwendeten Tools zeigt den Testfortschritt und die Qualität in standardisierter Weise. Dieser Testfortschrittsbericht sollte auch in jedem Projekt aus nachfolgenden Gründen von Anfang an wöchentlich erstellt werden:

- Information aller Stakeholder und des Managements über Einschätzung von Fortschritt und Qualität und erhebliche Aktivitäten, Probleme, Änderungen der laufenden Periode.
- Ausblick über geplante nächste Maßnahmen.
- Aufzeigen der Strategie und notwendiger Änderungen der Strategie während der Projektlaufzeit.
- Benennung von Risiken und Maßnahmen zur Reduktion dieser Risiken.
- Allgemein bekannte Einschätzung über Fortschritt und Qualität.
- Schwerwiegende Fehler, mit denen die Zielerreichung gefährdet ist oder Testaktivitäten verhindert werden.
- Absicherung der eigenen Position und Eigenmarketing für die Wichtigkeit des Testmanagements.

Dabei ist zu empfehlen, dass der Teststatusbericht am Freitag erstellt wird und spätestens am Montag präsentiert wird. Sollten sich nämlich aufgrund des aktuellen Status Änderungen ergeben oder das Management andere Gegenmaßnahmen empfehlen, kann man relativ schnell Änderungen anweisen. Wenn ein Statusbericht der Vorwoche erst gegen Mitte der laufenden Woche präsentiert wird, so sind viele Aussagen bereits wieder überholt, weil die Dynamik in Projekten dazu im Allgemeinen zu hoch ist. In vielen Projekten wird dazu übergegangen, den Teststatusbericht nur per Mail zu versenden. Aus meiner Erfahrung reicht das nicht aus. Zumindest einmal im Monat ist ein Project Review zu empfehlen, um direkt auf bestimmte Probleme hinweisen zu können oder bestimmte Aspekte besonders hervorzuheben. Ein Project Review Meeting hat den Vorteil, dass alle Beteiligten sich gleichzeitig mit der Thematik intensiv beschäftigen, während ein schriftlicher Bericht manchmal nur überflogen und abgelegt wird, sofern der Inhalt nicht nach gravierenden Schwierigkeiten aussieht.

Problematisch wird es erfahrungsgemäß, wenn ein Teil des Testteams offshore angesiedelt ist. Das kann schon bei einer Verteilung auf unterschiedliche Standorte innerhalb weniger Kilometer schwierig werden (auch da spreche ich aus eigener Erfahrung), bei einem Testteam im Ausland kommen Sprachprobleme, kulturelle Unterschiede und ggf. unterschiedliche Zeitzonen hinzu. Es ist darum von unschätzbarem Wert, sich immerhin im Abstand von einigen Monaten selbst bei Offshore-Teams am anderen Ende der Welt regelmäßig blicken zu lassen und offene Probleme vor Ort zu besprechen. Man benötigt in

solch einem Fall ohnehin regelmäßige Telefonkonferenzen, sie senken aber im Allgemeinen die Produktivität und die Effizienz im Team erheblich. Die Videotechnik ist auch heute meistens noch nicht derart professionell, dass eine virtuelle Konferenz an den persönlichen Austausch auch nur annähernd heranreicht.

Ungünstige Rahmenbedingungen, die ein Testmanager schon von Anfang an erkennt, sollten dabei quantifiziert werden. So kann man bei einer ausgeprägten Offshore-Organisation oder Organisationen, die über mehrere Standorte verteilt sind, Planungswerte für den Testaufwand pauschal um 10–20 % höher ansetzen als bei einem Team, das ständig vor Ort zusammenarbeitet. Einzelne Homeoffice-Tage sind bei einem gut eingearbeiteten Team hingegen möglich, ohne die Produktivität wesentlich zu senken. Diese Annahmen sollte man aber immer in Stunden und folgerichtig auch in Geld umrechnen, weil nur so die Risikobewertung ausreichend transparent und bewertet wird und nicht nur als allgemeines Gejammer des Testmanagers abgetan wird. Man kann unterschiedliche Meinungen über bestimmte Bewertungsfaktoren haben, entscheidend ist aber, dass sie von Anfang an einmal überhaupt angesetzt und dokumentiert werden. Homeoffice ist gut geeignet für abgegrenzte, definierte Arbeitspakete, etwa um Testfälle zu definieren oder eine Software zu codieren. Die Möglichkeiten für verteiltes Arbeiten haben sich aufgrund des technischen Fortschritts in den letzten Jahren erheblich verbessert, der persönliche Austausch muss aber nach wie vor gewährleistet sein.

Zur schnelleren Kommunikation im Testteam ist es sinnvoll, einen **Chat** zwischen Testmanager und Testteam einzurichten. In einem Chat kann die Kommunikation beschleunigt werden, sie entwickelt aber manchmal eine Eigendynamik, die man kaum noch einfangen kann. Wenn man Testmanager ist und per Chat sein Team unterrichtet, weil sich eben manche Fakten überraschend ereignet haben oder Entscheidungen sofort allen Mitgliedern mitgeteilt werden sollen, kann es schwierig werden, die Kommunikation wieder einzufangen. Ich war einmal in der Situation, eine bestimmte Information, die das gesamte Team betraf, den Mitarbeitern per Chat mitteilen zu müssen, und hatte dann die Kommunikation mit 5 Testern, die mit der Entscheidung nicht einverstanden waren, sich darüber beschwert haben und sich gegenseitig hochschaukelten, wieder einzufangen. Die Kommunikation verselbstständigt sich schnell; es wird dabei auch plötzlich über Themen gepocht, die mit dem ursprünglichen Problem höchstens noch mittelbar zusammenhängen. Man erlebt hier also ein Phänomen, das in den sozialen Medien oder Foren generell besteht. In bezeichnetem Fall musste ich ein eigenes Präsenzmeeting einberufen, um die Gemüter wieder einigermaßen zu beruhigen und hatte am Ende einen gesamten Tag für das ganze Team verloren.

Es ist auch dringend erforderlich, von Anfang an klare Regeln für den Chat aufzustellen, wann und wie er eingesetzt wird, was Thema im Chat sein darf und was nicht. Man kann sich in einem Chat oft ungezwungener unterhalten als in einer Mail, es sollte aber nicht ausufern, weil sonst wesentliche Informationen darin unterzugehen drohen.

Es ist allerdings aus meiner Sicht immer noch besser, einen Chat einzurichten und die Dynamik zu erhöhen, als dass Gerüchte entstehen und über „Flurfunk“ verbreitet werden. In diesem Fall ist die Dynamik in der Kommunikation nämlich noch schwieriger einzufan-

gen. Es kann sehr problematisch werden, wenn in einer Unternehmenseinheit *über* das Projekt und nicht *mit* den verantwortlichen Testmanagern und Projektleitern gesprochen wird.

Zur Testorganisation gehört die Organisation der **Testressourcen**. Dabei kann es sich handeln um Testpersonal, Rechenzeit, Datenbereitstellung und für den Test benötigte Hardware.

6.2 Testpersonal und Testereinsatzplanung

Bezüglich des Testpersonals muss geprüft werden, welche Mitarbeiter in welchem Umfang für den Test zur Verfügung stehen. Wird der Mitarbeiter für weitere Tätigkeiten oder andere Projekte benötigt, die ggf. eine höhere Priorität haben als das eigene Projekt? Weiterhin ist zu beachten, wann welcher Tester für welches Arbeitspaket im Projektablauf benötigt wird.

Beispiel einer **Testereinsatzplanung** (in Stunden):

Die Testereinsatzplanung ist bis auf den einzelnen Arbeitstag herunterzubrechen und dann zu Wochen und Monaten zu verdichten.

Dabei sind die Spezialgebiete der einzelnen Tester zu berücksichtigen: Sollte ein Tester vor allem die Beschreibung der Testfälle vornehmen und ein anderer eher die automatisierten Skripte durchführen, muss ihr Einsatzplan in den entsprechenden Phasen das vorsehen. Die Testereinsatzplanung kann dabei in PT (wie in Tab. 6.1) oder auf Stundenbasis (wie in Tab. 6.2) vorgenommen werden.

Nachdem man die Testereinsatzplanung auf der Basis einzelner Arbeitstage erstellt hat kann man sie auf Wochen- oder Monatsbasis verdichten.

Oft gibt es innerhalb des Projektes aber Verschiebungen: im obigen Plan war Tester B für den März für 140 Stunden eingeplant. Wenn es hier um die Testdurchführung geht, die Software aber im März nicht termingerecht fertig ist, verschiebt sich sein Aufwand teilweise in den April. Jetzt kann es aber sein, dass er im April geplant 2 Wochen im Urlaub ist, er kann also später starten und befindet sich mit seinem Aufwand auf dem „**kritischen Pfad**“. Man muss also neu planen und prüfen, ob ein Vertreter eingesetzt werden kann und ob die Verschiebung der Lieferung eventuell einen Einfluss auf das geplante Enddatum hat. Daher empfiehlt sich für die Ressourcenplanung der Einsatz eines professionellen Planungstools. Gerade dann, wenn Tester in verschiedenen Projekten gebunden sind, kann es vorkommen, dass sie zu bestimmten Perioden gar nicht oder zu wenig ausgelastet sind, in anderen Perioden aber überplant sind. Bei der Planung der Testaufwände sind immer

Tab. 6.1 Testereinsatzplanung pro Tag in PT

Tester	Montag 15.2.	Dienstag 16.2.	Mittwoch 17.2.	Donnerstag 18.2.
A	1	1	1	1
B	0,5	1	1	0,5
C	0	0	1	1

Tab. 6.2 Verdichtete Testereinsatzplanung auf Stundenbasis

Tester	Januar	Februar	März	April
A	60	80	80	40
B	40	80	140	80
C	0	60	80	100

Zeiten mit einzukalkulieren, die für allgemeine interne Abstimmungen oder Weiterbildungen vorgesehen sind. Eine regelmäßige Kommunikation zwischen der Linienorganisation und dem Projekt- bzw. Testmanagement über Einsatzräume und Verfügbarkeiten sollte daher vorhanden sein. Gerade dieser relativ triviale Punkt wird nach meiner Erfahrung in vielen Unternehmen nicht genügend beachtet. Ich hatte schon vermehrt Tester für bestimmte Aufgaben eingeplant und dann plötzlich von der Linie kurzfristig erfahren, dass sie in diesem Zeitraum anderweitig verplant oder abwesend sind. In diesem Fall ist vom Testmanager und der gesamten Abteilung viel Flexibilität und Spontaneität gefragt.

Gerade dann, wenn die Testautomatisierung noch nicht genügend fortgeschritten ist, kann es übergangsweise notwendig werden, Testfälle manuell zu wiederholen. Man benötigt dafür nicht unbedingt ausgebildete Testanalysten, sondern kann vielleicht auch Aushilfspersonal (z. B. über ein Zeitarbeitsunternehmen) oder Studenten für die Testdurchführung rekrutieren. Dafür müssen aber die Testfälle exakt beschrieben sein und die Analyse klar sein. Die Arbeit mit Aushilfen stößt dadurch in den meisten Fällen an sehr enge Grenzen. In diesem Fall muss ebenfalls kurzfristig schnell reagiert werden können. Man muss dabei einen gewissen Aufwand für Einarbeitung und allgemeine Organisation mit einplanen. Das generelle Ziel muss aber immer bleiben, Testfälle so weit wie möglich zu automatisieren. Man muss aber auch bedenken, dass eine Testautomatisierung einen gewissen Vorlauf benötigt und nicht schon bei den ersten Testzyklen produktiv läuft, sondern sich eher mittelfristig auszahlt.

6.3 Rechenzeit

In den 1970er- oder 1980er-Jahren wurden Rechenzeiten für den Großrechner oft den einzelnen Projekten zugewiesen. Inzwischen ist die Rechenzeit für die meisten Projekte irrelevant, da die Rechenkapazität der verwendeten Hardware längst keine Beschränkung mehr darstellt, sondern in der Regel nur zu einem verschwindend geringen Bruchteil ausgenutzt wird, und hier müssen folglich keine Einschränkungen mehr beachtet werden. Nur bei sehr rechenintensiven Projekten, etwa in der Astronomie, Raumfahrt oder Meteorologie oder der Auswertung sehr großer Datenmengen ist die Rechenzeit auch heute noch ein relevanter, den Test begrenzender Faktor.

6.4 Datenbereitstellung

Jeder Tester benötigt eigene Daten, damit nicht während der Testdurchführung die Daten von anderen Usern verändert werden. Nicht immer sind aber die Daten getrennt gehalten. Noch immer kommt es vor, dass dafür die Plattenspeicherkapazität nicht ausreicht oder das Datenbanksystem nur eine Kopie lässt. Auch dieses Problem ist in den letzten Jahren aber in den meisten Organisationen deutlich geringer geworden.

6.5 Benötigte Hardware

Wenn spezielle Maschinen oder Zusatzgeräte für die Testumgebung erforderlich sind, kann es notwendig sein, sie zu buchen. Dabei kann es auch darum gehen, dass spezielle Testarbeitsplätze in einem Labor für bestimmte Projekte zur Verfügung stehen müssen. Dazu gehört zum Beispiel auch die Einsatzplanung von Testfahrzeugen für Testfahrten. Bei mehrtägigen Testfahrten ins Ausland ist neben der Konfiguration der Testfahrzeuge mit der entsprechenden Hardware und Software auch Regelungen bezüglich Lenk- und Ruhezeiten zu beachten. Zur Hardware gehört gerade bei Tests in der Fläche auch die Ausstattung der Tester mit Mobiltelefonen und ausreichend Sicherungsmedien. Allein das Kopieren von Festplatten kann sich dabei zu einer logistischen Meisterleistung entwickeln.

6.6 Projektorganisation

Jedes Unternehmen bzw. jede Organisation verfügt über eine bestimmte Struktur. Es gibt beispielsweise die Unternehmensführung, Abteilungen und die darin beschäftigten Mitarbeiter. Diese Struktur wird als Stammorganisation (oder Linienorganisation) bezeichnet.

Da Projekte nun aber komplexe und anspruchsvolle „Sondervorhaben“ sind, werden für eine bessere Umsetzung (temporär) Projektteams und Organisationsstrukturen gebildet, die sie von der „normalen“ Organisationsstruktur abgrenzt.

Es wird also zusätzlich zur Stammorganisation eine Projektorganisation gebildet, die die Zusammenstellung der Rollen, Befugnisse und Schnittstellen eines Projektes beschreibt. Und damit ist die Projektorganisation für den Projekterfolg von zentraler Bedeutung.

Denn nur, wenn alle beteiligten Personen wissen, welche Aufgaben, Verantwortungen und Befugnisse sie innerhalb eines Projektes haben (und wie sich diese zur Stammorganisation verhalten), können sie sich auch dementsprechend (richtig) verhalten.

6.7 Stakeholder und Rollen in der Projektorganisation

Jedes Projekt hat seine eigene Zusammensetzung von Projektbeteiligten. Deshalb kann sich die Zusammensetzung einer Projektorganisation auch von Fall zu Fall stark unterscheiden. Folgende Akteure sollten aber immer berücksichtigt werden (und nur mit einer guten Begründung ausgeschlossen werden):

- Der **Auftraggeber** ist die Person oder Organisation(-seinheit), die den Auftrag für die Durchführung des Projektes gegeben hat und muss deshalb auch zwingend in der Projektorganisation abgebildet werden!
- Der **Lenkungsausschuss** ist ein dem Projektleiter übergeordnetes Entscheidungsgremium und hat u. a. folgende Aufgaben: Formulierung des Projektauftrags, Ressourcenzuteilung, Freigabe von Budgets und Projektphasen usw. Es kann sinnvoll sein einen Lenkungsausschuss für ein Projekt zu bilden; zwingend ist es jedoch nicht.
- Der **Projektleiter** (bzw. die **Projektleitung**) ist zuständig für die Planung, Überwachung und Steuerung eines Projektes und muss damit auf jeden Fall in der Projektorganisation vertreten sein.
- Und last but not least: die **Projektmitarbeiter**, welche die anfallenden Aufgaben umsetzen sollen [ERFO2016].

6.8 Organisationsstruktur

Es ist unbedingt erforderlich, dass die Unternehmensführung bzw. der Auftraggeber des Projektes dafür Sorge tragen, die notwendige Organisationsstruktur (Werte und Leitlinien der Unternehmung) und die strukturellen Voraussetzungen für ganzheitliche-situative Projektarbeit zu schaffen. Wird sie dieser Aufgabe nicht gerecht, kann das Projekt schon zum Scheitern verurteilt sein, bevor es begonnen hat.

Tab. 6.3 zeigt die notwendigen Voraussetzungen.

6.9 Projektlenkungsausschuss

Vor allem bei größeren Projekten ist die Einrichtung von Lenkungsausschüssen sinnvoll. Als Entscheidungs-, Koordinations- und Kontrollgremium ist die Einrichtung eines **Projektlenkungsausschusses** (PLA) auf gesamtorganisatorischer Ebene häufig anzutreffen. Dieser Ausschuss wird teilweise auch als Entscheidungsausschuss, Entscheidungskommission, Steuerungsgruppe, Steering Board, Steering Committee, Steuerungsausschuss, Lenungskreis oder Control Board bezeichnet. Er bildet die Schnittstelle zwischen

- Projekt, Projektleitung und Projektteam auf der einen Seite und

Tab. 6.3 Unterstützung der Projektarbeit durch die Unternehmensführung

Rahmenbedingungen	Leitlinien (Projektmanagementkultur)
Ausreichende Freiräume für das Projektteam sind gewährleistet	Mindestens ein Mitglied der Unternehmensführung ist inhaltlich (als „Sponsor“) beteiligt.
Die Bedingungen für eine optimale Informationsbasis der Teammitglieder sind geschaffen.	Niemand im Unternehmen sollte Projektarbeiten als Bedrohung ansehen, sondern als Chance, innovativ tätig zu werden und Verbesserungen durchzuführen.
Ein positives Betriebsklima für die Zusammenarbeit von Projektteam und Fachabteilungen ist sichergestellt.	Kompetente Mitarbeiter sind auf entsprechenden Wunsch für die Projektarbeit freizustellen bzw. bereitzustellen. Eine laufende Fortbildung zur Förderung der Mitglieder des Projektteams ist zu gewährleisten.
Der Projektleitung und dem Projektteam wird ausreichende administrative Unterstützung gewährt.	Zwischen dem Projektmanagement im Unternehmen und den anderen Führungskräften sollte keine Rivalität bestehen, sondern die Bereitschaft, voneinander zu lernen und sich gegenseitig zu unterstützen.

- Unternehmensführung und extern Beteiligten auf der anderen Seite.

Dieses Gremium überwacht das Projekt von Beginn an bis zur Umsetzung. Es stellt das oberste Gremium für die Planung und Steuerung aller Projekte dar, die in einer Organisation realisiert werden. Hier werden in letzter Konsequenz die entscheidenden Weichen für weitere Projektfortschritte gelegt. So kann der Projektlenkungsausschuss in der Regel darüber entscheiden, ob das Projekt gestoppt oder gar abgebrochen wird, wenn sich abzeichnet, dass die Projektziele überhaupt nicht erreicht oder andere übergeordnete Ziele durch das Projekt sogar gefährdet werden.

Bezüglich der Zusammensetzung hat es sich bewährt, in dieses Gremium hochrangige Führungskräfte zu entsenden. Da in jedem Gremium Entscheidungen von erheblicher Tragweite getroffen werden, sollten in jedem Fall ein Mitglied der Geschäftsleitung sowie die IT-Leitung im Projektlenkungsausschuss vertreten sein. Die Projektleitung kann mitunter ebenfalls an den Sitzungen teilnehmen (bei unterschiedlichem Stimmrecht).

Es hat sich dabei als zweckmäßig erwiesen, dass sich die Mitglieder des Lenkungsausschusses in regelmäßigen Abständen (z. B. monatlich) treffen, darüber hinaus bei Bedarf auch außerplanmäßig. Bezüglich der Beschlussregeln gilt das „Einstimmigkeitsprinzip“, um so eine weitgehende Koordination zwischen den beteiligten Bereichen und für die spätere Umsetzung zu erzielen.

Bei der Projektmanagementmethode **PRINCE 2** (Projects in Controlled Environments) richtet sich die Zusammensetzung der Mitglieder des Projektlenkungsausschusses nach den Projektphasen, sodass sich diese regulär immer am Ende jeder Projektphase treffen, um die abgeschlossene Projektphase abzunehmen und die folgende Phase zu genehmigen. Der Projektlenkungsausschuss kann auch außerhalb der regelmäßigen Treffen bei Bedarf einberufen werden, um über sogenannte Ausnahmeberichte oder dringende Change Requests zu entscheiden.

Literatur

- [IODT2019] : <https://www.iot-design.de/allgemein/transformation-von-testorganisationen/>, zugegriffen am 09.05.2020
- [ERFO2016] : <https://erfolgreich-projekte-leiten.de/projektorganisation/>, zugegriffen am 09.05.2020

Zusammenfassung

Der Testprozess besteht aus mehreren Phasen und ist durch regelmäßige definierte Abstimmungsmeetings zu begleiten. Die Projektmitglieder im Testteam haben gewisse Aufgaben und tragen durch aktives Handeln und zielgerichtete Kommunikation zum Testerfolg bei. Für den Projekterfolg notwendige Testvoraussetzungen sind zu beachten und zu organisieren.

7.1 Testprozess nach ISTQB

Der nach dem Standard ISTQB definierte **Testprozess** besteht aus fünf Aktivitäten (Abb. 7.1):

- Planung und Steuerung,
- Analyse und Design,
- Realisierung und Durchführung,
- Auswertung und Bericht sowie
- Testabschluss.

Die einzelnen Schritte müssen dabei nicht exakt nacheinander ausgeführt werden, sondern können teilweise auch parallel ablaufen. Es kann dabei auch zu Rückkopplungen kommen, etwa wenn bei der Testdurchführung bemerkt wird dass der Testentwurf für einen bestimmten Testfall geändert werden muss.

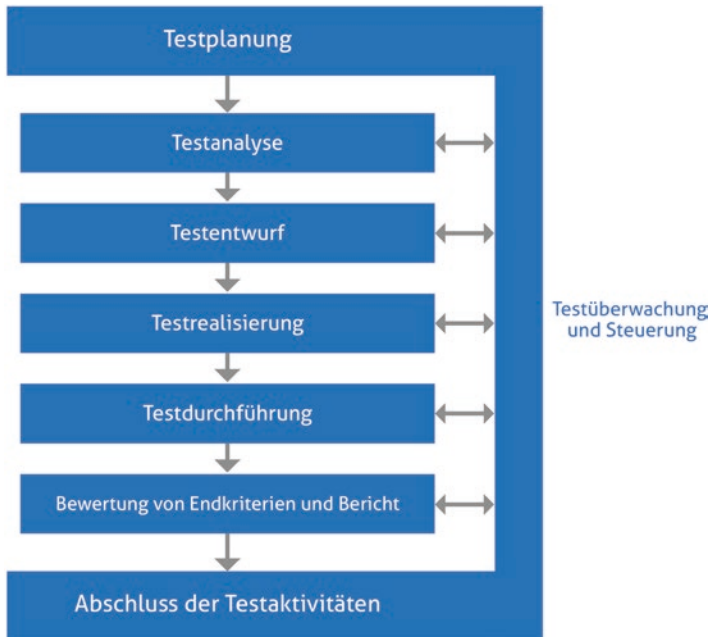


Abb. 7.1 Testprozess [BAST2015]

7.2 Planung und Steuerung

Der Testprozess muss strukturiert abgewickelt werden. Daher muss die Verfügbarkeit der erforderlichen Testressourcen exakt geplant sein. Ein Zeitplan für die einzelnen Testaktivitäten ist notwendiger Bestandteil eines Testkonzeptes.

Die Teststeuerung beinhaltet die laufende Überwachung aller Aktivitäten im Rahmen der Tests, der Entwicklungen und Korrekturen, aber auch das termingerechte und inhaltlich korrekte Testreporting.

Zur Teststeuerung werden während der Projektlaufzeit geeignete **Meetings** (Tab. 7.1) angesetzt.

Bei Bedarf können Defectmeetings auch spontan, etwa zu einem bestimmten Fehler, einberufen werden, um die weiteren Aktivitäten und notwendigen Schritte zu besprechen. Regelmäßige Defect Boards oder Change Control Boards dienen dazu, die Tragweite und Konsequenz von Defects zu bewerten und nötige Maßnahmen einzuleiten.

Während der Testdurchführung werden häufig Anforderungen geändert oder es kommen zusätzliche Anforderungen hinzu. Vor allem in agilen Projekten ist dies der Fall. Dafür muss ein Changeprozess eingeführt werden.

Changes müssen dokumentiert werden, ihre Auswirkungen auf das Gesamtsystem und auf den Test bewertet und die Kosten kalkuliert werden. Die Changes sind von definierten

Tab. 7.1 Geeignete Meetings im Testprozess

Meeting	Zweck	Teilnehmer	Rhythmus
Statusmeeting	Information der Führung über Testaktivitäten und Definition möglicher Steuerungsmaßnahmen	Projektleiter Testmanager Qualitätsmanager	Wöchentlich
Defectmeeting	Status und notwendige Aktivitäten zur Fehlerbehebung	Testmanager Releasemanager Integrator Entwickler	Wöchentlich
Testphasenabschluss	Abnahmesitzung und Entscheidung über Fortführung der Testaktivitäten in nächste Testphase	Projektleiter Testmanager SW-Integrator	Nach jeder Testphase
Reviewmeeting	Dokumentenreview	Testphase Tester	Nach Bedarf
Abnahmesitzung	Entscheidung über die Produktivsetzung nach der Abnahmephase	Lenkungskreis Projektleiter Kunde Testmanager Integrator Releasemanager	Einmalig
Releasemeeting	Information über die Aktivitäten im Release und Definition möglicher Steuerungsmaßnahmen	Testmanager Releasemanager	Wöchentlich

Stellen zu prüfen und freizugeben. Dabei müssen die Regeln und Prozesse zur Beurteilung von Changes und die Kompetenzen im Vorfeld eindeutig beschrieben sein [EPHE2016].

7.3 Meetings zu Projektstart

Gerade zum Beginn eines Projektes kommen der Motivation, der Awareness für das Projekt und der Vereinbarung von Kooperationsformen für die Mitglieder des Projektteams eine besondere Bedeutung zu. Außerdem sind alle von den Projektergebnissen künftig Betroffenen sowie die in die Projektarbeit einzubeziehenden Personen vorab in ausreichender Weise zu informieren. Es hat sich dabei bewährt, zwei Arten von **Startveranstaltungen** durchzuführen:

- einen **Start-up-Workshop** mit dem Projektteam sowie
- eine Informationsveranstaltung für alle Betroffenen (= **Kick-Off-Meeting**)

Mit einem gut vorbereiteten projektbezogenen Start-up-Workshop werden die ersten Grundlagen für ein erfolgreiches Projekt gelegt. Insbesondere kann der Start-up-Workshop dazu beitragen, dass

- sich die Mitglieder des Projektteams näher kennenlernen (auch persönliche Erfahrungen und Ziele dürfen dabei angesprochen werden),
- sich ein „Wir-Gefühl“ im Projektteam entwickelt,
- eine erste Rollenbesetzung im Projekt vorgenommen wird und
- mögliche Risiken und Konflikte für die Projektarbeit schon in der Startphase erkannt werden.

Das Kick-Off-Meeting wird in der Regel vom Testmanager bzw. Testprojektleiter vorbereitet und unterscheidet sich vom Start-up-Workshop zum einen von der Teilnehmergruppe. Neben der Projektleitung und dem Projektteam sollten daran teilnehmen:

- Vertreter des Auftraggebers (bei externen Projekten Vertreter des Kunden),
- Leitung der durchführenden Institution (etwa ein Sponsor aus der Unternehmensführung) sowie
- wesentliche Projektinteressenten und Personen, die von den Projekten betroffen sind (Stakeholder).

Stakeholder in der IT, seien es Softwareentwickler oder Tester, sprechen jeweils eine eigene Sprache und betrachten das Projekt aus völlig unterschiedlichen Perspektiven. Das Team ist in starkem Maße interdisziplinär aufgestellt, sodass das Projektmanagement und das Testmanagement die Zusammenarbeit von Menschen verschiedener Denkart ermöglichen muss.

Tab. 7.2 Aufgaben und Befugnisse von Projektmitgliedern

Typische Aufgaben	Befugnisse/Verantwortung
Selbstständige und verantwortliche Bearbeitung von Arbeitspaketen	Mitwirkung bei der Sicherstellung der Projektziele
Sachlich qualifizierte Fertigstellung der übernommenen Aufgaben im vereinbarten Terminrahmen	Termin- und kostengerechte Erledigung der Arbeitsaufträge
Regelmäßige Statusmeldungen zum Stand der übernommenen Arbeitspakete	Mitbestimmung bei Entscheidungen und Vereinbarungen zur Projektarbeit
Unverzügliche Mitteilung an den Projektleiter bei sachlichen oder zeitlichen Abweichungen von der Planung	Aktive Mitarbeit bei den Teamsitzungen und Dynamik in der Kommunikation
Unterstützung der Projektleitung durch Mitwirkung bei der Projektplanung	Rechtzeitiges Einfordern von Personal, Sachmitteln (Kapazitäten) und Finanzen bei der Projektleitung
Erarbeitung von Entscheidungsvorlagen	
Erstellen der zur Projektsteuerung notwendigen Berichte (Tätigkeitsberichte, Arbeitsaufträge und Arbeitsauftragserledigung)	

Das Projektteam sorgt in letzter Konsequenz für die Ausführung der verschiedenen Teilaufgaben im Projekt, wobei jedes Teammitglied seinen ihm gemäßen Beitrag dazu leisten muss.

Typische Aufgaben der Mitarbeiterinnen und Mitarbeiter im Projekt sowie vorhandene Befugnisse zeigt Tab. 7.2.

7.4 Analyse und Design

Die Analyse wird in der Phase der Testkonzepterstellung vorgenommen. Dabei ist zunächst zu prüfen, ob die relevanten Prozesse bekannt sind und die Testziele klar definiert werden können. Die gewünschte Testabdeckung ist dabei ebenfalls zu beschreiben.

Weiterhin muss kalkuliert werden, ob der benötigte Testumfang mit den bestehenden Ressourcen überhaupt zu leisten ist. Sollte es dabei nicht möglich sein, kurzfristig weitere Tester zu akquirieren, ist eine Risikoanalyse vorzunehmen, auf welche Testfälle man verzichten kann.

In dieser Phase muss auch bewertet werden, inwiefern das System überhaupt testbar ist. Wenn zu Testbeginn nur Teilfunktionalitäten zur Verfügung stehen, ist zu prüfen, inwiefern sich daraus eine Aussage über die Testbarkeit des Gesamtsystems ableiten lässt.

Etwas zeitversetzt zur Erstellung des Testkonzeptes wird normalerweise mit der Definition der Testfälle begonnen. Die Definition der Testvoraussetzungen, Testdaten und Verweise auf vorangestellte Tests gehören ebenfalls zum Testdesign. Abweichungen und Besonderheiten gegenüber Standardprozessen sind in dieser Phase ebenfalls zu planen. Auch die Konzeption zur Testautomatisierung im Projekt soll parallel zur Testfalldefinition starten. Jeder Testfall, der automatisiert ausgeführt werden soll, muss zumindest in einer Basisausprägung einmal manuell getestet worden sein.

7.5 Realisierung und Durchführung

Am Beginn dieser Phase steht ein Smoke-Test, der die grundsätzliche Verfügbarkeit des Testsystems prüft. Dabei kann festgestellt werden, ob bei Hauptfunktionen oder grundlegenden Features bereits Probleme bestehen.

Vor allem während manueller Testdurchführung sollten Testfälle priorisiert abgearbeitet werden. Die Funktionsfähigkeit wesentlicher Funktionen des Prüflings muss im Vordergrund stehen, um relativ zügig Aussagen über die Softwarequalität treffen zu können. Das gilt besonders dann, wenn Ressourcenengpässe oder Verzögerungen bei der Bereitstellung des Testobjektes bereits aufgetreten sind.

Bei komplexen Testszenarien, bei denen ein Schritt auf dem anderen aufbaut, sollte bei Fehlschlägen einer Funktion am Anfang der Kette nicht gleich der gesamte Testprozess abgebrochen werden, sondern der Fehler notiert aber ggf. mit einer Umgehungslösung weitergetestet werden. Dann lassen sich im weiteren Verlauf trotzdem noch sinnvolle Re-

sultate liefern. Allerdings besteht dabei die Gefahr, dass Folgefehler auftreten, die mit einer Behebung des ersten Fehlers ebenfalls nicht mehr auftreten. Hierbei sind viel Erfahrung und Systemkenntnisse des Testers gefragt. Sehr komplexe Testszenarien mit vielen Abhängigkeiten der einzelnen Schritte innerhalb der Prozedur bergen daher immer das Risiko von Verzögerungen und nicht erfolgreichen Testergebnissen.

Bei automatisierten Testfällen können solche Konstellationen ebenfalls auftreten. Die Statistik zeigt in diesen Fällen eine erhebliche Abnahme der Qualität. Deshalb ist es außerordentlich wichtig, auch bei automatisierten Tests genügend Zeit für eine zielführende Analyse der Resultate einzuplanen.

Regressionstests (Testwiederholungen) sollten immer komplett durchgeführt werden. Es kommt in der Praxis häufig vor, dass mit Behebung eines Fehlers neue Fehler als Seiteneffekt der Behebung auftreten oder auch ein Fehler durch einen anderen Fehler maskiert war und erst nach Behebung des Fehlers überhaupt gefunden werden kann. Wenn man aus Zeitgründen auf komplette Wiederholungen verzichtet, so ist immer vom Testmanager eine Risikobewertung durchzuführen und das Risiko im Abschlussbericht zu dokumentieren und zu bewerten.

Die Festlegungen zur Realisierung und Durchführung sollten im Allgemeinen im übergreifenden Testkonzept getroffen werden.

7.6 Auswertung und Bericht

Im Vorfeld sollten bereits Metriken festgelegt werden, die im Rahmen der Testdurchführung wöchentlich erhoben und berichtet werden (Kap. 1, 6, 3).

Die laufende Auswertung erlaubt es dem Testmanager, die Einhaltung des Terminplans sicherzustellen und den Zeitpunkt der Abnahme festzuhalten.

Bei besonderen Problemen oder dringendem Klärungsbedarf sollte auch außerhalb von Berichtszyklen eskaliert werden.

7.7 Testabschluss

Am Ende des Testprozesses ist ein **Abnahmetest** durchzuführen, in dessen Rahmen die korrekte Funktionsfähigkeit des Testsystems anhand ausgewählter Testfälle demonstriert werden soll.

Dazu wird mit dem Auftraggeber ein besonderer Termin vereinbart. Die Testfälle, die dabei vorgeführt werden, sollten im Vorfeld definiert und dem Auftraggeber bereits zur Kenntnis vorgelegt sein.

Man beschränkt sich dabei in der Regel auf die wesentlichen Funktionen und Features des Systems, nicht auf exotische Sonderfälle. Dazu werden ausgewählte Testszenarien definiert. Der Abnahmetest wird auf einer Umgebung, die die Produktivumgebung exakt abbildet, vorgeführt (notfalls auch auf der Produktivumgebung, wenn es nicht zu Proble-

men in der Produktion kommt). Das ist wichtig, da sonst die Ergebnisse des Abnahmetests nicht sicher übertragbar sind.

Erfahrungsgemäß sollten alle Stakeholder und auch andere Projekte, die die Abnahmeumgebung nutzen, im Vorfeld davon in Kenntnis gesetzt werden. Ich hatte selbst schon eine Kundenabnahme erlebt, die wiederholt werden musste, weil die Umgebung am selben Tag nicht wie abgestimmt konfiguriert worden war. Das ist dann außerordentlich peinlich, wenn mehrere Vertreter des Kunden extra dazu eingeladen waren und an diesem Tag aus einem anderen Ort anreisen müssen.

Zur Vorbereitung der Abnahme ist also besondere Sorgfalt und genügend Zeit einzuplanen. Die Abnahme sollte ein Tag vor Besuch des Kunden als „Generalprobe“ exakt so durchgeführt werden wie am Tag selbst, um das Risiko einer misslungenen Abnahme zu minimieren. Ansonsten ist es besser (wenn möglich) den Termin von vornherein zu verschieben, anstatt eine wiederholte Abnahme durchführen zu müssen.

7.8 Dokumente als Testvoraussetzung

Im Testkonzept sollte definiert werden, welche Angaben der Tester neben der Software von der Entwicklungsabteilung benötigt.

Folgende Dokumente sind zu empfehlen:

- **Release Notes:** Die Release Notes (Versionshinweise) sind Dokumente, die zusammen mit Softwareprodukten verteilt werden, manchmal auch, wenn sich das Produkt noch im Entwicklungs- oder Teststatus befindet. Bei Produkten, die bereits von Kunden verwendet wurden, wird die Release-Information dem Kunden zugestellt, wenn ein Update veröffentlicht wird.
- **Featureliste:** In der Featureliste stehen alle Features im Überblick. Sollten von der Software verschiedene Umfänge in den unterschiedlich lizenzierten Versionen angeboten werden (z. B. „Gold“ – „Silber“ – „Bronze“), so sind die möglichen Features pro Version markiert. Bei einer Übergabe während des Entwicklungsprojektes kann die Featureliste dazu verwendet werden, die noch nicht fertiggestellten Features zu dokumentieren.
- **Dokumentation von Einschränkungen:** Hier wird beschrieben, ob es in der Softwarelieferung bekannte Fehler oder Einschränkungen gibt, die erst in einer Folgeversion behoben werden können.
- **Requirements:** Dabei wird dokumentiert, welche Anforderungen bereits in der Software umgesetzt sind und für welche Anforderungen die Realisierung noch offen ist. Eine **Traceability Matrix** beschreibt, welche Anforderung mit welchen Testfällen abgedeckt ist und welcher Testfall sich auf welche Anforderungen bezieht. Dieses Dokument ist wichtig für die Nachvollziehbarkeit und Bewertung von neuen Anforderungen und nicht zuletzt für Audits des Testprozesses.

- **Architekturskizze:** Für Benutzer und Entwickler des Softwareprojektes gibt eine gut konstruierte Softwarearchitektur leicht ein grundlegendes Verständnis des Systems. Wichtige Faktoren, die auf die Eignung der Softwarearchitektur Einfluss nehmen, sind Projektplanung, Risikoanalyse, Organisation, Entwicklungsprozess, Arbeitsabläufe, Hardware, Qualitätssicherung und Anforderungen. Diese Softwarearchitektur wird bildhaft in einem Schema festgehalten. Eine Architekturbeschreibung umfasst etwa im Falle einer Webanwendung den Aufbau des Systems aus Datenbanken, Web-/Applicationservern, E-Mail- und Cachesystemen. Dabei muss geprüft werden, welche Informationen benötigt werden, um einen Test durchführen zu können.

Literatur

- [EPHE2016] : https://ephesos.fhsg.ch/documents/10328/103271/13983558_Fiorelli+Roger_Mas-terarbeit.pdf/17786419-b168-415f-b7fb-1258cb73024f, zugegriffen am 10.05.2020
- [BAST2015] : Bucsics, Baumgartner, Seidl, Gwihs: Basiswissen Testautomatisierung, dpunkt Verlag Heidelberg 2015

Zusammenfassung

Testfälle sind mit Testobjekten verbunden. Testobjekte bestimmen den Inhalt der Testaktivitäten. Testaktivitäten innerhalb der einzelnen Teststufen finden in mehreren Phasen statt. Für die Ermittlung des Testaufwands sind unterschiedliche Einflussfaktoren verantwortlich.

8.1 Testobjekte

Die **Testobjekte** sind alle von einem Testfall betroffenen Objekte. Ein Testfall ist einer oder mehreren Anforderungen und damit mittelbar mit einem Anwendungsfall verbunden. Ein Anwendungsfall durchläuft die Methoden mehrerer Objekte.

Beispiel: Der Testfall lautet „Fällt die Artikelmenge auf Lager unter die Mindestmenge, muss automatisch nachbestellt werden“.

In diesem Fall sind folgende Objekte betroffen:

- Auftragsverwaltung,
- Kundenstamm,
- Bestellwesen,
- Artikelstamm und
- Lieferposten.

Man spricht in diesem Zusammenhang auch vom **Wirkungsbereich (Impact Domain)** des Testfalls im **Objektmodell**. Er ist meist eine Untermenge des Wirkungsbereichs des Anwendungsfalls, denn im Anwendungsfall „Kundenauftragsbearbeitung“ werden nicht

alle Testfälle bis zum Artikelobjekt kommen, und nur dieser Testfall wird einen Lieferauftrag erzeugen.

Die einzelnen Testobjekte bzw. Komponenten sind eindeutig inkl. Angabe von Versionsnummern, Übergabemedium und Spezifikationen zu beschreiben.

Die Testobjekte definieren den Testinhalt und bestimmen dadurch, was getestet wird. In den Testobjekten ist das zu testende System beschrieben. Die Testobjekte müssen genau voneinander abgegrenzt werden. Objekte, die beim Test nicht mitgetestet werden, sondern als gegeben hingenommen werden, sind hier ebenfalls zu beschreiben. Objekte als gegeben hinzunehmen, ist aber nur dann möglich, wenn sie sich nicht während der Projektphase aus anderen Gründen ändern. Wenn im obigen Beispiel der Artikelstamm sich aufgrund der Anforderung nicht ändert und daher nicht explizit getestet werden muss, gilt das nur, solange nicht andere Anwendungen Änderungen daran vornehmen. Daher ist ein Anforderungssystem, mit dem man erkennt, welche Anforderung welches System und dadurch mittelbar welchen Testfall beeinflusst, von besonderer Wichtigkeit. Nur so bekommt man mit, dass sich das Systemverhalten bei der Testdurchführung ändern kann und der Testfall ggf. anzupassen ist. Die Angaben im Testkonzept müssen ausreichend sein, um Testfälle abzuleiten.

In diesem Zusammenhang werden die Zustände der betroffenen Objekte vor Ausführung des Testfalls als **Testfallvorzustände** und die Zustände nach der Ausführung des Testfalls als **Testfallnachzustände** bezeichnet [SNEE2012].

8.2 Testphasen

Innerhalb der einzelnen Teststufen werden immer bestimmte Aktivitäten und Phasen durchlaufen (Abb. 8.1):

Aufgrund empirischer Studien kann man davon ausgehen, dass knapp 50 % des gesamten Aufwands der Testaktivitäten auf die Testdurchführung und jeweils knapp 20 % auf die Testvorbereitung und die Testauswertung entfallen. Sofern die Testautomatisierung weit fortgeschritten ist, fällt der Anteil für die Testdurchführung, wird aber nicht null. In diesem Zusammenhang sei vor unrealistischen Erwartungen an automatisierte Tests gewarnt, die leider immer wieder in vielen Unternehmen anzutreffen sind. So wichtig es ist, die Testautomatisierung voranzutreiben, so unrealistisch ist es anzunehmen, dass alle automatisierten Tests „auf Knopfdruck“ funktionieren und dadurch kein Testaufwand mehr besteht.

Je nach Qualität der Software muss die Wiederholung der Testdurchführung unterschiedlich oft durchgeführt werden. Für Regressionstests ist immer genügend Zeit einzuplanen. Man kann sich dabei an verschiedenen Kenngrößen, zum Beispiel der Lines of Code, orientieren, und dann Indexwerte wie „erwartete Fehler pro 1000 Zeilen Code“ bilden, um Anzahl und Aufwand für Fehlerbehebung und dadurch erforderliche Regressionstests zu prognostizieren.

Vom Testaufwand abzugrenzen sind zum einen konstruktive **Qualitätssicherungsmaßnahmen**, die zur präventiven Qualitätsgestaltung eingesetzt werden. Sie setzen an der

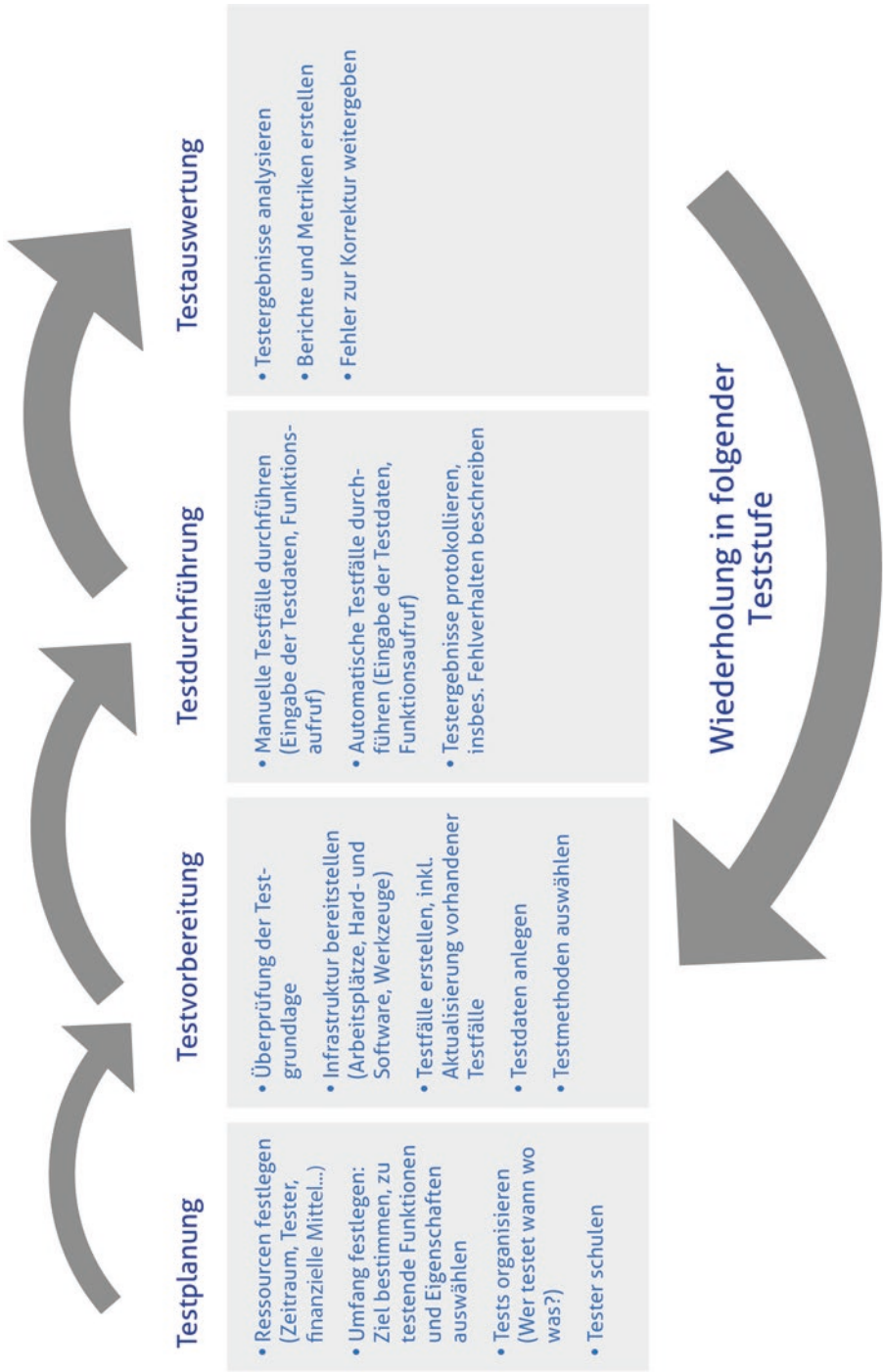


Abb. 8.1 Testaktivitäten

Prozessqualität an und tragen dadurch indirekt zur Verbesserung der Softwareproduktqualität bei. Zum anderen schließt Testen Aktivitäten der statischen analytischen Qualitätssicherung aus, mittels derer das Prüfobjekt unabhängig von seiner Umgebung analysiert wird, d. h. ohne Code auszuführen.

Der Testaufwand wird demnach wie folgt definiert:

- die Arbeitszeit
- aller am Testen Beteiligten
- für die Vorbereitung, Ausführung und Auswertung
- aller automatisierten und manuellen Testfälle in einem Projekt
- ohne Aktivitäten der Testplanung.

8.3 Einflussfaktoren für die Ermittlung des Testaufwands

Bei der Ermittlung des Testaufwands in einem Projekt sind positive oder negative Rahmenbedingungen angemessen zu berücksichtigen. Die projektespezifisch ausgeprägten Merkmale, die den Testaufwand beeinflussen, werden wie folgt gruppiert:

- Merkmale des Produktes,
- Merkmale des Teams,
- Merkmale der Ressourcen und
- Merkmale der Testorganisation.

Diese **Einflussfaktoren** wirken sich dabei auf unterschiedliche Weise und mehrfach auf den Testaufwand aus.

Merkmale des Produktes

Zu den Merkmalen des Produktes gehören nachfolgend erklärte Faktoren

- Marktbedeutung: Umsatzziele, strategische Ziele (z. B. Diversifikation, Markteintritt), die mit dem Testobjekt verfolgt werden, Anzahl Kunden des früheren Releases, Anzahl erwarteter Kunden, geplanter Produktpreis;
- Neuartigkeit: Neuentwicklung oder Redesign eines Produktes, Anzahl Vorreleases, Bekanntheit und Erfahrung mit der eingesetzten Entwicklungstechnologie;
- Dynamik der Anforderungen: Anzahl und Zeitpunkt der Änderungen während der Laufzeit des Projektes;
- Grad der Kundenindividualität: Variabilität der Nutzung des Kundenproduktes, individuelle Anpassung des Produktes an unterschiedliche Bedürfnisse;
- Kundenerwartungen an das Produkt: Wichtigkeit des Produktes aus Kundensicht (Kritikalität), Existenz von Vergleichsprodukten aus Kundensicht, Anzahl Beschwerden von Kunden des direkten Vorreleases;

- Umfang des Softwareproduktes (Umfang der Anforderungen);
- Komplexität des Softwareproduktes: Eingesetzte Entwicklungstechnologien, Anzahl externer Schnittstellen, Anzahl interner Schnittstellen zwischen Produktkomponenten im Projekt bzw. zwischen Subprojekten;
- Angestrebte Qualität des Softwareproduktes: Angestrebte Funktionalität, Zuverlässigkeit, Benutzbarkeit und Effizienz;
- Dokumentation der Anforderungen und des Entwurfs: Qualität der Spezifikation und Entwurfsdokumentation.

Merkmale des Teams

Die Merkmale des Teams gliedern sich in folgende Faktoren auf

- Erfahrung: Erfahrung der Entwickler mit dem Fachgebiet, Erfahrung der Entwickler mit ihren Aufgaben, Erfahrung der Tester mit dem Produkt (im weiteren Sinne Branchenkenntnisse), Erfahrung der Tester mit ihren Aufgaben;
- Motivation und Einstellung zur Qualität: Engagement der Tester, Engagement der Entwickler, Einstellung der Projektleitung zur Qualität;
- Teamgröße (Anzahl Projektteammitglieder);
- Zusammenarbeit: Kontinuität des Teams, Verfolgung eines gemeinsamen Ziels, Arbeitsatmosphäre;
- Kommunikation: Anzahl verschiedener Muttersprachen im Team, Anzahl der Standorte der Projektmitarbeiter;
- Rollenbesetzung: Testfallerstellung und Testdurchführung durch Tester oder Entwickler.

Merkmale der Ressourcen

Die Merkmale der Ressourcen bestehen aus nachfolgenden Faktoren

- Personelle Restriktionen: Verfügbarkeit der Teammitglieder, Abhängigkeit von Zulieferungen;
- Kundenbeteiligung beim Testen: Anzahl Kunden und Ansprechpartner, die am Test teilnehmen;
- Vorhandene Testfälle und Daten: Qualität existierender Testfälle, Wiederverwendbarkeit der Testdaten;
- Testinfrastruktur: Verfügbarkeit und Stabilität des Testsystems, Grad der Testautomatisierung;
- Zeit für die Projektdurchführung (zu Beginn des Projektes in Summe).

Merkmale der Testorganisation

Die Merkmale der Testorganisation beinhalten die Auswahl der Testfälle und die zeitliche Verteilung des Testaufwands.

Weitere Faktoren anderer Modelle

Der Testbarkeit der Software wird häufig ein starker Einfluss auf den Testaufwand zugeschrieben. Dazu tragen nicht nur der Code, sondern auch andere Artefakte wie Entwurf und Dokumentation bei.

Der Testaufwand wird weiterhin durch die Fähigkeit der Entwickler beeinflusst, weil je nach Ausprägung ihrer Fähigkeiten die Entwickler bei Anforderungsanalyse, Systemdesign, Entwurf und Coding mehr oder weniger mögliche Fehler im Vorfeld erkennen bzw. vermeiden und dadurch das entwickelte Produkt von Anfang an mehr oder weniger Fehler aufweist. Für das Management ist es in diesem Zusammenhang wichtig, die Stärken und Schwächen der Entwickler zu kennen und die einzelnen Mitarbeiter dort einzusetzen, wo ihre Stärken am besten zur Geltung kommen und ihre Schwächen am wenigsten anrichten können.

Die verwendeten Testmethoden üben ebenfalls einen Einfluss auf den Testaufwand aus. Unter einer **Testmethode** ist ein planmäßiges, auf einem Regelwerk aufbauendes Vorgehen zur Herleitung und Auswahl von Testfällen zu verstehen. Grundsätzlich wird in diesem Zusammenhang zwischen funktionsorientierten und strukturorientierten Vorgehensweisen unterschieden, wofür die Begriffe **Black-box** bzw. **White-Box-Verfahren** verwendet werden. Beim Test sind unterschiedliche Verfahren einzusetzen, um unterschiedliche Fehlerarten zu finden. Es reicht nicht aus, nur das korrekte Nutzerverhalten zu testen, sondern muss sich auch kreative Fehlerfälle ausdenken, um zu testen, wie das System in ungeplanten Situationen reagiert, also ob verständliche Fehlermeldungen ausgegeben werden oder ob es zu Systemabstürzen kommt. Aufgrund der mit den zu entdeckenden Fehlern variierender Effektivität und Effizienz der Testmethoden und ihrem unterschiedlich hohen Aufwand in der Anwendung wird der Testaufwand beeinflusst [[DOWI2009](#)].

Literatur

- [SNEE2012] : Sneed, Bauimgartner, Seidl: Der Systemtest, Carl Hanser Verlag München 2012
[DOWI2009] : Ulrike Dowie, Testaufwandsschätzung in der Softwareentwicklung, Josef-Eul-Verlag Köln 2009

Zusammenfassung

Der Testprozess findet in mehreren Teststufen statt. Das Testkonzept definiert die Inhalte der einzelnen Teststufen und grenzt sie voneinander ab. Eine gute Teststrategie sorgt für eine effiziente Testdurchführung, verhindert doppelte Aktivitäten in unterschiedlichen Teststufen und sorgt für eine möglichst hohe Testabdeckung. Für agile Projekte gelten dabei besondere Teststrategien.

9.1 Klassische Teststufen

Im Testkonzept sind die **Teststufen** mit ihren Inhalten zu definieren. #

Jede Funktion sollte in einer möglichst frühen Teststufe getestet werden, damit Fehler sich nicht in höhere Teststufen fortpflanzen.

Im Allgemeinen gibt es folgende Teststufen:

- **Komponententest:**

Einzelne Softwarebausteine werden isoliert von anderen Softwarebausteinen des Gesamtsystems einem Test unterzogen. Festgestellte Fehler werden der getesteten Komponente zugeordnet. Die Testumgebung ist die Entwicklungsumgebung, die Arbeitsweise sehr entwicklungsnahe. Die Aufgabe des Komponententests ist die korrekte und vollständige Realisierung des Ein- und Ausgabeverhaltens des Testobjektes. Um Korrektheit und Vollständigkeit der Implementierung zu prüfen, wird die Komponente einer Reihe von Testfällen unterzogen, wobei jeder Testfall eine bestimmte Ein-/Ausgabekombination (Teilfunktionalität) abdeckt. Neben Funktionalität und Robustheit sollen im Komponententest des Weiteren alle diejenigen Komponenteneigenschaften überprüft werden, die in höheren Teststufen nicht mehr oder

nur mit wesentlich höherem Aufwand geprüft werden können. Dies gilt beispielsweise für die nichtfunktionalen Eigenschaften Effizienz und Wartbarkeit.

Beim Komponententest hat der Tester in der Regel Zugang zum Sourcecode, sodass der Komponententest ein typischer **White-Box-Test** ist. Der Begriff White-Box-Test bezeichnet eine Methode des Softwaretests, bei der die Tests mit Kenntnissen über die innere Funktionsweise des zu testenden Systems entwickelt werden. Im Gegensatz zum **Black-Box-Test** ist für diesen Test also ein Blick in den Quellcode gestattet, d. h., es wird am Code geprüft. Der Komponententest wird durch den Entwickler durchgeführt.

- **Integrationstest:**

Beim Integrationstest wird das Zusammenspiel der einzelnen Komponenten miteinander geprüft. Das Vorgehen beim Integrationstest wird in zwei unterschiedliche Strategien unterschieden: Die **testzielorientierte Strategie** benötigt nur die zum Testen notwendigen Systemkomponenten. Dort werden die Testfälle entsprechend definierter Testziele erstellt und ausgeführt. Die **vorgehensorientierte Strategie** ist abhängig von der Integrationsreihenfolge aus der Systemarchitektur.

Eine weitere Unterscheidung ist die Art, wie die Komponenten integriert werden: Die „**inkrementelle Integration**“ fügt die Module schrittweise zusammen. Hierzu müssen fehlende Module simuliert werden. Die „**nichtinkrementelle Integration**“ fügt alle Komponenten gleichzeitig zusammen und integriert sie in das zu testende System. Dazu müssen alle Module bereits vor dem Integrationstest vorhanden sein („Big-Bang“). Diese Variante hat sich in der Praxis als nicht tauglich erwiesen. Wenn man zu viele Komponenten auf einmal zusammenfügt, sind Fehler weitaus schwerer zu lokalisieren. Außerdem kann es zu Maskierung von Fehlern kommen (ein Fehlverhalten bedingt, dass ein anderer Fehler gar nicht erst auftritt und daher beim Test nicht erkannt werden kann, es kommt zu einem „versteckten Fehler“), die einen erhöhten Aufwand bei der Fehlerbehebung nach sich ziehen.

Für jede Abhängigkeit zwischen zwei Komponenten eines Systems wird dazu ein **Testszenario** definiert, welches in der Lage ist nachzuweisen, dass nach der Zusammenführung sowohl beide Komponenten für sich wie auch der Datenaustausch über die gemeinsamen Schnittstellen spezifikationsgemäß funktionieren. Die Funktionstests weisen an dieser Stelle nach, dass die richtige Komponente verwendet wurde. Schnittstellentests dienen zur Überprüfung der Daten, die zwischen den Komponenten ausgetauscht werden.

Zum Integrationstest werden Testtreiber benötigt, die Testobjekte mit Testdaten versorgen und Ergebnisse entgegennehmen sowie protokollieren. In der Regel werden dazu bereits aus dem Komponententest vorhandene Testtreiber verwendet. Das **Testziel** beim Integrationstest ist das Aufdecken von Schnittstellenfehlern. Typische Fehler beim Datenaustausch sind z. B. die Übermittlung syntaktisch falscher Daten, die unterschiedliche Interpretation übergebener Daten, Timing-Probleme (die Daten werden richtig übergeben), aber zum falschen oder verspäteten Zeitpunkt und Durchsatz-, Last- oder Kapazitätsprobleme (die Daten werden in zu kurzen Zeitintervallen übergeben).

- **Systemtest:**

Im Systemtest wird das System als Ganzes in einer Testumgebung betrachtet, die der späteren Produktivumgebung so gut wie möglich entspricht. Statt Testtreibern und Platzhaltern sollen also auf allen Ebenen möglichst die später tatsächlich zum Einsatz kommenden Hardware- oder Softwareprodukte in der Testumgebung installiert und konfiguriert sein (Hardwareausstattung, Systemsoftware, Treibersoftware, Netzwerk, Fremdsysteme usw.). Das Testziel besteht in der Verifikation und Validierung der funktionalen und nichtfunktionalen Anforderungen. Fehler und Mängel aufgrund falsch, unvollständig oder im System widersprüchlich umgesetzten Requirements sollen aufgedeckt und bisher nicht dokumentierte oder vergessene Anforderungen identifiziert werden.

- **Abnahmetest**

Der Abnahmetest wird vor Inbetriebnahme der Software als abschließender Test zusammen mit dem Kunden bzw. Anwender durchgeführt. Dabei werden ausgewählte Testfälle präsentiert und nachgewiesen. Damit wird die Erfüllung des vertraglich vereinbarten Funktionsumfangs dokumentiert. Akzeptanztests können auch im Rahmen niedrigerer Teststufen oder über mehrere Teststufen verteilt vorkommen.

Das Testkonzept soll dabei festlegen, welche Systemaspekte wie intensiv getestet werden müssen und auf welcher Teststufe das jeweils geschehen soll [SPIL2007].

Die einzelnen Teststufen sollten terminlich definiert und voneinander deutlich abgegrenzt werden. Auch die Übergabe zwischen einzelnen Teststufen ist dabei zu definieren. Dabei kann es durchaus vorkommen, dass Teststufen parallel durchlaufen werden. Wenn nämlich eine Funktion erst zu einem späteren Zeitpunkt entwickelt wird, während andere Funktionen schon im Integrationstest sind, kann es vorkommen, dass sie erst zu einem späteren Zeitpunkt in ein dann teilweise fertig integriertes Produkt integriert wird.

Die Durchführung der einzelnen Teststufen erfolgt durch unterschiedliche Stakeholder: der Komponententest wird vom Entwickler durchgeführt, während Integrations- und Systemtest von einer eigenen Testinstanz bearbeitet werden und beim Abnahmetest außerdem der Auftraggeber/Kunde an der Testdurchführung beteiligt ist. Die einzelnen Verantwortlichen sind zu benennen und die Aktivitäten im Testkonzept zu beschreiben. Darüber hinaus sollte man regelmäßige Meetings von Anfang an organisieren, um den Testfortschritt zu verfolgen und notwendige Aktivitäten zeitnah einleiten zu können.

Besonders bei Zulieferung einzelner Komponenten muss definiert werden, welche Testabdeckung zur Qualitätssicherung erforderlich ist. Der Zulieferer trägt in diesem Fall die Verantwortung für das Teilprodukt, das integriert und zusammen mit anderen Komponenten in den Teststufen ab Integrationstest getestet wird. Dazu sind genaue Festlegungen erforderlich, damit nicht manche Funktionen mehrfach getestet werden und andere gar nicht.

Die Termine der einzelnen Teststufen ergeben sich meist aus der Releaseplanung.

9.2 Exploratives Testen

In diesem Zusammenhang ist auch der Umfang für das **explorative Testen** festzulegen: neben den methodischen Ansätzen sind Testfälle intuitiv zu ermitteln. Das ist erforderlich, weil in der Praxis die Anforderungen nie vollständig sind, sondern immer wieder lückenhafte Definitionen vorzufinden sind, die meist erst während der Testdurchführung aufgedeckt werden können. Die intuitive Fähigkeit und die Erfahrung, Testfälle nach erwarteten Fehlerzuständen und -wirkungen auszuwählen, ist Grundlage des explorativen Tests, bei dem methodische Verfahren hier sehr schnell an ihre Grenzen kommen. Die Erfahrung, welche Funktionen in der Vergangenheit besonders fehleranfällig waren und die Vermutung, wo sich Fehler in Zukunft wahrscheinlich wiederholen, sind dazu notwendig und bedeuten, dass man erfahrene Tester im Projekt benötigt. Diese Art der Testfallerstellung wird auch als **Error Guessing** bezeichnet [SPIL2007].

Das explorative Testen ist als eine Form des Softwaretests definiert, die die persönliche Freiheit und Verantwortung des einzelnen Testers hervorhebt, damit die Qualität seiner Arbeit optimiert werden kann, indem Testdesign, Testdurchführung, Testinterpretation und testbezogenes Lernen als sich gegenseitig unterstützende Aktivitäten behandelt werden und gleichzeitig während der Projektlaufzeit fortgeführt werden. Die Betonung liegt hier darauf, dass der einzelne Tester zwar die persönliche Freiheit bei seiner Arbeitsplanung genießt, aber diese Freiheit auch mit der Verantwortung für seine Arbeit einhergeht. Verantwortung in diesem Zusammenhang bedeutet, dass der Tester auch dafür Sorge trägt, dass seine Arbeitsergebnisse nachgestellt werden können, indem beispielsweise Notizen zu seinen Testaktivitäten vorliegen.

Beim explorativen Testen sind also die drei Einzelteile des Testens per Definition miteinander verbunden. Exploratives Testen ist simultanes Lernen, Testdesign und Testdurchführung in einem. Das bedeutet, dass exploratives Testen die drei Einzelteile des traditionellen Testens nicht vollständig voneinander trennt.

Eine Struktur im explorativen Testen besteht darin, die Applikation unter verschiedenen Aspekten zu betrachten. Dafür wurde die **FCC-CUT-VIDS-Touring-Heuristik** eingeführt. Die Buchstaben stehen für Features, Complexity, Claims, Configuration, User, Testability, Scenario, Variability, Interoperability, Data und Structure. Diese stehen für einzelne Touren, die als Tester durch die Applikation genommen werden können. Sie geben der Testtätigkeit damit Struktur – Einzelteile, die sich zu einem Ganzen zusammenfügen.

Eine weitere Struktur für Exploratives Testen bietet die **HICCUPPS** Eselsbrücke von James Bach und Michael Bolton. HICCUPPS steht für History, Image, Comparable Products, User Expectations, Product, Purpose und Standards & Statutes. Dahinter verbergen sich verschiedene **Orakel** – ein Prinzip oder Mechanismus, anhand dessen wir feststellen können, ob die Software im Sinne eines bestimmten Anwenders funktioniert. Orakel helfen dabei, Inkonsistenzen in einem Produkt festzustellen. Wenn sich beispielsweise von der Produktversion 2003 zur Produktversion 2007 von Microsoft Office die Benutzerober-

fläche komplett ändert, dann ist sie inkonsistent zur gewohnten Arbeitsweise der Benutzer, der History in der obigen Liste.

Dass die Änderung in Microsoft Office – hoffentlich – mit voller Absicht vorgenommen wurde, verdeutlicht einen weiteren Aspekt von Orakeln: sie sind Heuristiken und können auch mal danebenliegen. Das bedeutet, dass der Tester mithilfe von Heuristiken weiterhin sein Gehirn benutzen muss. Denn ein automatisierter Test kann nicht selbstständig entscheiden, ob eine Änderung gewollt oder ungewollt ist, man braucht die menschliche Interpretation der Ergebnisse, wie leider viel zu viele Teams immer wieder mit Erschrecken feststellen. Eine fehlerhafte Konfiguration kann Tausende automatisierte Testfälle auf einmal fehlerhaft werden lassen (und bei einer Automatisierung der Fehlererfassung dadurch auch Tausende Tickets erfassen). Ein erfahrener Tester hingegen wird schnell bemerken, dass der Fehler in der Konfiguration sich höchstwahrscheinlich auf weitere Testfälle auswirken wird und diese Tests vor Behebung des Konfigurationsfehlers gar nicht weiter ausführen.

Heuristiken, Orakel und Touring-Modelle sind verschiedene Strukturen, die wir beim explorativen Testen einsetzen. Dabei kann Testautomatisierung exploratives Testen genauso unterstützen, wie auch das explorative Testen die Testautomatisierung über neue Testfälle informieren kann. Beide Ansätze bilden gerade in der agilen Softwareentwicklung eine Symbiose, da sie sich sehr stark gegenseitig unterstützen. Beispielsweise kann man mithilfe von Testautomatisierung damit beginnen, um die getesteten Werte herumzutesten. Wenn man beispielsweise einen automatisierten Test für die Werte 5 und 10 hat, kann man schnell einen Test mit den Werten 1, 11, 42 und MAXINT durchführen.

Exploratives Testen gilt als Testansatz, der sich mit anderen Techniken verbinden lässt. Dabei ist jede Form von Testen explorativ zu einem gewissen Grad und gleichzeitig vorbestimmt oder als Script erstellt. Sich wiederholende Testtätigkeiten überlässt man dabei besser der Automatisierung und setzt den Faktor Mensch gezielt dort ein, wo er einen wesentlichen Beitrag liefern kann: beim Lernen und beim kreativen Designen von Testfällen auf Basis von bereits ausgeführten Testfällen. Denn genau dafür ist das menschliche Gehirn geschaffen [EXPL2019].

9.3 Teststufen in einem agilen Projekt

Folgende Skizze (Abb. 9.1) verdeutlicht eine Best Practice für die Teststufen in einem agilen Projekt:

Hier wird der Fokus der einzelnen Teststufen im Testprozess definiert und wie diese Teststufen zusammenwirken, um doppelte Prüfungen zu vermeiden. In einer Teststrategie, die für das Projekt erstellt wird, sowie eine Detailplanung für jeden Sprint, sollte das Vorgehen frühzeitig definiert werden, um eine effiziente Testausführung zu gewährleisten.

Eine Auswahl und Verfeinerung der Teststufen sollte dabei durchgeführt werden (z. B. kann ein Integrationstest sich aufspalten in einen Modul- und einen Systemintegrationstest), ebenso sollten die Anforderungen an das zu testende System klar sein. Darüber

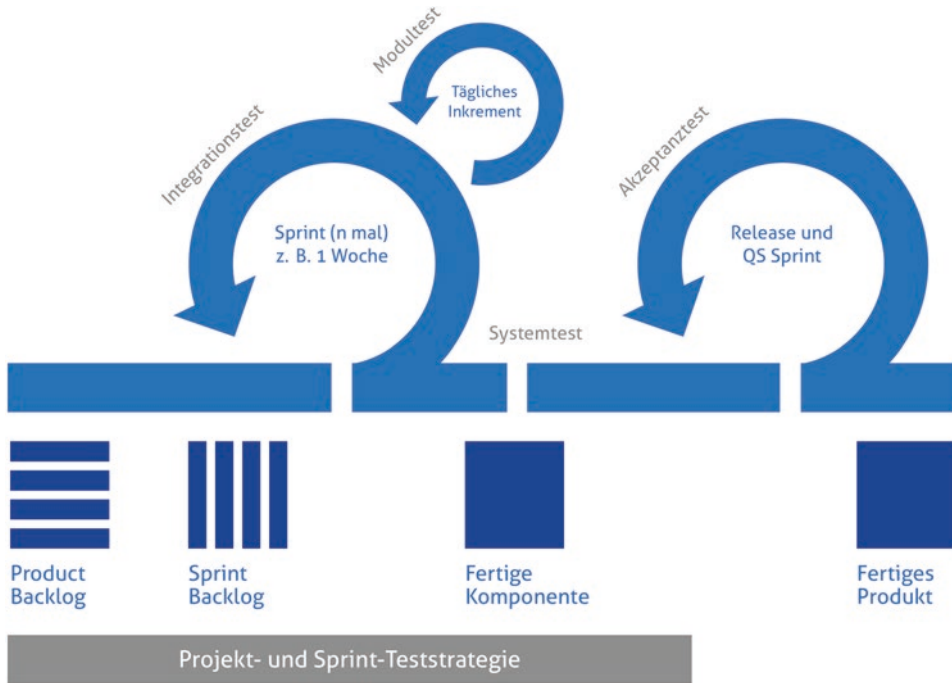


Abb. 9.1 Beispiel einer Projekt- und Sprint Teststrategie

hinaus muss der konkrete Entwicklungsprozess bekannt sein, um die Strategie sowie die konkret durchzuführenden Testaktivitäten auf den jeweiligen Teststufen zuordnen zu können.

Vor allem bei größeren Projekten empfiehlt es sich, Stufentestkonzepte je Testebene inklusive Beschreibung der zu verwendenden Testmethoden, Hilfsmittel und testende Kriterien zu erstellen. Das Testkonzept hat in diesem Fall Abschnitte, die für alle Teststufen übergreifend gelten neben Abschnitten, die nur für eine spezielle Teststufe relevant sind.

Bei der stufenweisen Vorgehensweise geht es zunächst darum, die Teststrategie der einzelnen Teststufen zu definieren. Ein wesentlicher Aspekt dabei ist eine Festlegung, wie intensiv auf welcher Stufe getestet werden soll.

Das hängt von mehreren Einflussgrößen ab:

- verfügbares Testbudget,
- Risiko,
- Komplexität des zu entwickelnden Systems,
- Anzahl der Teams und
- Know-how über Testaktivitäten.

Dem Testmanager obliegt hierbei die Aufgabe, möglichst viele dieser Faktoren zu ermitteln und zu berücksichtigen, oftmals in enger Abstimmung mit weiteren Rollen wie einem Projektleiter oder dem Product Owner. Als **Product Owner** (deutsch: Produkteigentümer) bezeichnet man eine der sechs Rollen im **Scrum Prozess**. Der Product Owner gehört dabei neben dem Scrum Master und dem Entwicklungsteam zu den zentralen Akteuren. Diese Person ist für die Wertsteigerung des Produktes im Entwicklungsprozess verantwortlich und leitet das Team an. Eine Strategie kann sein, dass auf der Modulebene jedes Modul mit **Test Driven Development** entwickelt werden muss, bevor es integriert wird und den Integrationstest durchläuft und auf Systemtestebene dann alle User Stories manuell durchlaufen werden müssen.

Test Driven Development (testgetriebene Entwicklung) bedeutet, dass Tests dazu benutzt werden, um die Softwareentwicklung zu steuern. Der Ablauf dieser Programmierung ist zyklisch:

- Ein Test wird geschrieben, der zunächst fehlschlägt.
- Es wird genau so viel Produktivcode implementiert, dass der Test erfolgreich durchläuft.
- Test und Produktivcode werden refaktorisiert.

Die Tests werden typischer-, aber nicht notwendigerweise, mit dem XUnit-Framework in der gleichen Sprache wie der Produktivcode implementiert. Tests, die erfolgreich durchlaufen, werden durch einen grünen, nicht erfolgreiche durch einen roten Balken dargestellt. Man spricht daher vom „Red-Green-Refactor-Zyklus“.

Testgetriebene Entwicklung geht inkrementell vor. Jeder durchlaufene TDD-Zyklus reichert die Software um neue Fähigkeiten an – und das minutiös, denn jeder Zyklusabschnitt sollte nicht länger als ein paar Minuten dauern.

Die Tests noch vor den Komponenten zu schreiben, die man eigentlich testen möchte, ist sehr markant für TDD. Dies wird als Test-First bezeichnet und darum ist TDD keine Test-, sondern eine Designstrategie. Denn wird der Test zuerst geschrieben, so wird die Schnittstelle der zu testenden Komponente bereits benutzt, bevor sie tatsächlich existiert. Der Entwickler bekommt frühestmöglich Feedback, ob das Design auch verwendbar sein wird.

Die Implementierung des produktiven Codes erfolgt erst, wenn ein Test vorliegt, der dies verlangt. Es soll dann genau so viel Code geschrieben werden, dass der Test erfolgreich durchläuft. Wird zu viel Produktivcode für einen Test geschrieben, entstehen nicht getestete Stellen, die beim Refactoring Probleme bereiten können.

Beim **Refactoring** werden die Tests und der Produktivcode gleichermaßen aufgeräumt. Ziel hierbei ist es, die Software einfach, redundanzfrei und verständlich zu gestalten. Diese Phase des TDD-Zyklus geht dem Zyklusanfang unmittelbar voraus: Um einen bestimmten Test zu schreiben, kann es notwendig sein, zuerst ein Refactoring an den anderen Tests oder dem Produktivcode vorwegzunehmen.

Der Zyklus ist überschneidungsfrei, jede Aktivität beim testgetriebenen Entwickeln lässt sich einem Abschnitt zuordnen. Es sollten keine Tests in Phase 2 und 3 und kein

Produktivcode in Phase 1 und 3 geschrieben werden. Beim Refactoring wird das Verhalten des Codes nicht verändert, also dabei weder bei den Tests (Phase 1) noch im Produktivcode (Phase 2) etwas funktional verändert.

Prinzipien, die TDD vereint, sind die kontinuierliche Designverbesserung, einfaches Design und Test-First. TDD selbst ist eine Kerntechnik von **Extreme Programming** und damit Teil der agilen Softwareentwicklung. Es verspricht Qualitätssoftware und eine deutliche Aufwertung der Softwarearchitektur dank evolutionärem Design.

Vorteile von Test Driven Design sind:

- Wartbare Qualitätssoftware (kein ungetesteter Code, saubere und testbare Architektur durch Test Driven Design als Designstrategie, keine bzw. wenig Redundanzen durch rechtzeitiges Refactoring)
- Effektive und effiziente Erstellung der Software (kein unnötiger Code auf Vorrat, Konzentration auf das Wesentliche) [ITAG2020]

Auf jeder Teststufe kann diese Strategie weiter verfeinert werden. Beispielsweise kann auf Systemtestebene jede als kritisch eingestufte Userstory von mindestens zwei Testern sowohl erfahrungsbasiert als auch systematisch getestet werden, während für weniger kritische Users Stories nur ein Tester Tests durchführt. Der Entwicklertest auf Unitebene kann sowohl auf White-Box- als auch auf Black-Box-Basis stattfinden und sollte von den Entwicklern durchgeführt werden. Dabei geht es nicht nur um die korrekte Funktionalität bei gültigen Eingaben, sondern auch um das korrekte Abfangen ungültiger Eingabewerte. Jede „kleinste Einheit“ („Unit“) wird möglichst isoliert getestet und geprüft.

Bei agilen Entwicklungslebenszyklen hat sich der Einsatz von Test-First oder Test Driven Development etabliert. Im Integrationstest wiederum werden die Schnittstellen zwischen Einheiten geprüft und somit das Zusammenspiel mehrerer Einheiten getestet. Auch wenn jede Einheit für sich korrekt arbeitet, kann das Zusammenspiel trotz allem zu fehlerhaften Ergebnissen führen. Der Integrationstest kann in mehrere Phasen aufgeteilt werden, um von „kleinen“ Integrationsschritten zu „großen“ Integrationsschritten alles abzudecken. Auf dieser Teststufe spielt insbesondere bei iterativer Entwicklung der Regressionstest eine große Rolle. Mit der Prüfung nichtfunktionaler Aspekte wie Performanz oder Security kann bereits im Integrationstest begonnen werden.

Der Systemtest wiederum wird auf Black-Box-Basis durchgeführt und umfasst die Prüfung des kompletten Systems, in der Regel von der Benutzeroberfläche aus. Geprüft werden soll, ob auf Basis der Eingaben die Ziele der Benutzer mit der Anwendung erreicht werden können. Auf dieser Teststufe finden auch nichtfunktionale Tests statt.

Alle diese Teststufen finden in der Umgebung der Entwicklung statt. Erst der Akzeptanztest findet in der Umgebung der künftigen Anwender statt und wiederholt Testfälle, die im Systemtest bereits durchlaufen wurden. Abb. 9.1 zeigt ein Beispiel eines agilen Vorgehens und nennt die Teststufen – der Modultest wird hier täglich durchgeführt, der Integrationstest am Ende des einwöchigen Sprints. Systemtests werden nach einigen Sprints begonnen und erste Testfälle durchlaufen. Der letzte Sprint vor dem Release wird

explizit für die Qualitätssicherung genutzt. In diesem Sprint werden auch Akzeptanztests durchgeführt.

Bei diesem Vorgehen wird fortlaufend geprüft, ob die initiale Teststrategie sinnvoll ist und ggf. angepasst. Stellt sich zum Beispiel während der Entwicklung heraus, dass eine bisher nicht als risikoreich eingeschätzte Komponente risikoreich ist, sollten Testaufwände hier zusätzlich konzentriert werden, beispielsweise durch weitere Modul- oder Integrationstests. Fehler, die während der Tests auf den unterschiedlichen Stufen gefunden werden, sollten dokumentiert werden (z. B. in einem Bugtracking-Werkzeug) und analysiert werden, um zukünftige Qualitätssicherung weiter zu optimieren und ggf. zu entscheiden, ob Fehler schon früher gefunden bzw. gänzlich vermieden werden können.

9.4 Integration der Testaktivitäten mit anderen Projektaktivitäten

Die Testaktivitäten sind Teil des Gesamtprojektes. Testaktivitäten sind davon abhängig, dass Testobjekte zur Testdurchführung zur Verfügung stehen, und Rolloutpläne sind erst dann möglich, wenn die Software abgenommen und für die produktive Einführung freigegeben wurde. Daher ist der **Testplan** als Teil des Gesamtprojektplans zu sehen.

Risiken und Konsequenzen, die sich aus diesen Abhängigkeiten ergeben, sind von Anfang an zu prüfen und zu bewerten. In der Prozesskette bestimmt das schwächste Glied die Geschwindigkeit und letztlich auch die Qualität des Endproduktes. Daher kommt der Bewertung des Testumfeldes eine besondere Bedeutung zu.

Man sollte dazu mithilfe von geeigneten Parametern die Auswirkungen und das Risiko quantifizieren. Die **zyklomatische Komplexität** nach McCabe oder das **Komplexitätsmaß nach Rechenberg** sind dazu geeignete Werte.

Die strukturelle Komplexität eines Programmcodes mithilfe der Pfade zeichnet sich auf dem Kontrollflussgraphen ab. McCabe ging davon aus, dass alle Programmverzweigungen, ausgelöst durch bedingte Anweisungen (z. B. If-Statements) oder Schleifenblöcke, die Komplexität eines Programms erhöhen. Die Menge der einzelnen Pfade im Kontrollflussgraphen spiegeln die Anzahl der notwendigen Testfälle für eine vollständige Pfadabdeckung wider. Viele mögliche Pfade bedeuten ein hohes Maß an Komplexität. Hohe Komplexität führt normalerweise dazu, dass Quellcode schwerer zu verstehen und zu testen ist. In der Folge droht die Gefahr, dass sich schon zur Entwicklungszeit schwerwiegende Fehler in den Code einschleichen.

Beim Komplexitätsmaß nach Rechenberg hängen Lesbarkeit, Verständlichkeit und Änderbarkeit von Anweisungsstruktur, Ausdrucksstruktur und Datenfluss eines Programms ab. Dabei wird die Komplexität von Verschachtelungen aus den einzelnen Komplexitäten abgeleitet, die mit geeigneten Faktoren gewichtet werden.

Literatur

- [SPIL2007] Basiswissen Softwaretest, dpunkt Verlag Heidelberg, 3.Auflage 2005 korrigierter Nachdruck 2007
- [EXPL2019] <https://www.it-agile.de/wissen/agiles-engineering/exploratives-testen-mit-struktur/>,
zugegriffen am 10.05.2020
- [ITAG2020] : <https://www.it-agile.de/wissen/agiles-engineering/testgetriebene-entwicklung-tdd/>,
zugegriffen am 10.05.2020

Zusammenfassung

Im Testkonzept werden die Testobjekte beschrieben. Dabei müssen funktionale Tests, Lasttests und Stresstests für die zu testenden Leistungsmerkmale geplant und näher definiert werden. Für den Test der Benutzeroberfläche werden für Internetapplikationen A/B-Tests und multivariate Tests eingesetzt.

10.1 Arten von Systemtests

Das Testkonzept beschreibt die **Leistungsmerkmale** des Prüflings. Es beschreibt also, was bzw. welche Funktionen und Features getestet werden müssen, es benennt die Testobjekte, aber nicht, wie der Test im Detail auszuführen ist.

Das Testkonzept enthält daher nicht die einzelnen Testspezifikationen und Testfälle, sondern referenziert auf deren Ablageort. Das kann ein spezielles Laufwerk oder ein definiertes Tool sein.

Die wesentlichen Funktionen (also die Überschriften zu Kapiteln der Testspezifikationen oder **Test Cluster**) sollten aber vor allem bei umfangreichen Projekten im Testkonzept beschrieben sein. In diesem Zusammenhang ist zu klären, wie lange die Testfälle aufgehoben werden müssen. Im Fall von Fehlern, die erst im Lebenszyklus des getesteten Produktes auftreten, muss man evtl. noch darauf zurückgreifen können.

Generell sollte man folgende Arten von Systemtests definieren:

1. Funktionaler Test

Der funktionale Test überprüft die korrekte Umsetzung der funktionalen Anforderungen an das System, also ob das Verhalten der Software bzw. des Systems den Anforderungen (Requirements) im Pflichtenheft entspricht. Funktionale Tests werden für

Systemtests und Akzeptanztests (auch Abnahmetests, Verfahrenstests oder User-Acceptance-Tests genannt) eingesetzt. Bei diesen Testarten handelt es sich um Black-Box-Tests, da „lediglich“ das externe Verhalten ohne Beachtung der internen Struktur bzw. des Codes überprüft wird.

Um überhaupt die funktionalen Anforderungen zu kennen, ist ein effizientes **Requirementsmanagement** im Projekt erforderlich. Dabei geht es um folgende Tätigkeiten:

- **Festlegung einer Attributierung der Anforderungen:** Bewertung, Priorisierung, Quelle der Anforderungen sind mögliche Attribute, die dokumentiert und verfolgt werden.
- **Änderungsmanagement:** Umgang mit Änderungen von Anforderungen, damit ihre Auswirkungen getrackt werden können und man die Konsequenzen der Änderung aufzeigen kann und sie dadurch entsprechend bewerten kann.
- **Bewertung und Priorisierung von Anforderungen:** Analyse der Kritikalität der Anforderung und Zuordnung der Anforderungen zu Versionen, um den Zeit- und Kostenrahmen zu erfüllen.
- **Verfolgbarkeit von Anforderungen:** Herstellung von Abhängigkeiten zwischen Anforderungen, Nachweis der Umsetzung einer Anforderung (Welche Anforderung wurde mit welchen Testfällen getestet? Auf welche Anforderungen bezieht sich ein bestimmter Testfall?). Dazu gehört die Definition der Prozesse zur Verfolgbarkeit.
- **Versionsmanagement:** Versionierung aktueller Arbeitsstände, Voraussetzung für ein Konfigurationsmanagement im Projekt.
- **Variantenmanagement:** Wenn eine Produktlinie entwickelt wird, werden unterschiedliche Varianten festgelegt. Der Requirementsmanager muss diese Variabilität für den zu entwickelnden bzw. verwendeten Prozess berücksichtigen [BLHO2015].

In der Praxis kommt es immer wieder vor, dass das Anforderungsmanagement nicht oder nur rudimentär vorhanden ist. Oft wird es auch dann, wenn das Projekt zu spät gestartet oder zu optimistisch geplant wurde, reduziert oder weggelassen (manchmal auch aus vermeintlicher Einsparung, was sich später als fataler Fehler herausstellt). In diesem Fall ist ein deutliches Testrisiko im Testkonzept zu deklarieren. Die Güte des Testmanagements und der Entwicklung selbst wird stark vom Anforderungsmanagement beeinflusst, da es im Entwicklungsprojekt am Anfang steht und andere Prozesse erst darauf aufbauen können. Der Test kann also am Ende immer nur so gut sein wie das Anforderungsmanagement, weil es die Basis darstellt.

In jedem Fall müssen im Testkonzept Testaufgaben für alle zentralen Funktionen von Anfang an definiert sein. Für die Überprüfung der Qualitätsmerkmale müssen ebenfalls Testaufgaben definiert sein. Man muss dabei auch beachten, dass jegliche Weiterentwicklung von Produkten problematisch wird, wenn die erste Version ohne ausreichende Basis entwickelt wurde.

Zu beachten ist weiterhin, dass manche Anforderungen zu allgemein formuliert sind, um in einem Test nachgewiesen werden können, oder dass sie auf andere Anforderungen (in einem weiteren Dokument) verweisen. In diesem Zusammenhang ist es erforderlich, diese Anforderungen so detailliert herunterzubrechen und so fein granular

zu definieren, dass ein Test oder eine andere geeignete Form eines Nachweises möglich ist.

Im Zusammenhang mit Requirements ist besonders darauf zu achten, dass Anforderungen nicht widersprüchlich und vollständig sind. Bei der Erstellung von Testfällen fällt regelmäßig auf, dass es offene Fragen wegen der zugrunde liegenden Dokumentation der Anforderungen gibt. Sollten diese offenen Fragen nicht ausreichend geklärt sein, stellt das ebenfalls ein Testrisiko dar, da in diesem Fall der funktionale Test nicht umfassend genug dokumentiert ist. Dieser kann zwar durchgeführt werden, das erwartete Verhalten bleibt aber fraglich.

2. Test des Lastverhaltens (Lasttest):

Beim Lasttest wird überprüft, wie sich das System bei einer dauerhaft hohen Anzahl an Zugriffen verhält. Der Lasttest erzeugt dabei eine sehr hohe Last auf dem zu testenden System, um es an den Rand seiner Leistungsfähigkeit zu bringen und untersucht dessen Verhalten. Ziele des Lasttests sind

- die Aufdeckung der im funktional orientierten Systemtest/Integrationstest nicht gefundenen Fehler;
- die Erfüllung nichtfunktionaler Anforderungen, wie z. B. geforderte Antwortzeiten sowie Mengenverarbeitungen oder zahlreiche gleichzeitige Zugriffe auf eine Datenbank, für den Produktivbetrieb nachzuweisen;
- die Dimensionierung der Hardwareausstattung zu überprüfen.

Bei einem Lasttest wird ein Werkzeug benötigt, da es hier darauf ankommt, innerhalb kurzer Zeit mehrere Tausend Transaktionen hintereinander und nebeneinander auszuführen, d. h. zur selben Zeit werden an verschiedenen Arbeitsplätzen mehrere Testskripte in einer Schleife ausgeführt. Dabei können die Arbeitsplätze auch von einem Programm simuliert werden. Es ist wichtig, darauf zu achten, dass auf die gleichen Daten von parallel laufenden Testsequenzen heraus zugegriffen wird, damit Warteschlangen vor den Datenbanken entstehen. Auf diese Weise wird das Auftreten von **Dead-Lock-Situationen** provoziert. Dabei wird die Zeit für jede Transaktion zwischen Betätigung der Enter-Taste bzw. des Startsignals und der Rückgabe der Antwort gemessen. Am Ende erhält man eine mittlere Antwortzeit für alle Transaktionen sowie eine Gesamtdauer für die Abarbeitung von n Transaktionen. Weiterhin wird dabei ermittelt, ob das System einer solchen Belastung überhaupt gewachsen ist, ohne abzubrechen oder fehlerhaft zu reagieren. Deshalb müssen die Ergebnisse beim Belastungstest auch inhaltlich auf Richtigkeit korrigiert werden. Angesichts der Menge an Daten muss diese Kontrolle automatisiert erfolgen [SNEE2012].

3. Stresstest:

Wenn das System bewusst über die definierte Lastgrenze hinaus an seine Grenzen gebracht oder plötzliche Belastungsspitzen simuliert werden, wird dies als Stresstest bezeichnet. Während es beim Belastungstest um eine hohe, aber konstante Belastung geht, geht es beim Stresstest um Ausreißer.

Dabei wird die Last (Anzahl der virtuellen User) schrittweise bis über die definierte Lastgrenze hinaus erhöht werden.

Damit werden folgende Fragestellungen untersucht:

- Wie ändert sich das Antwortzeitverhalten in Abhängigkeit von der Last?
- Kann mit dem System auch unter hoher Last noch akzeptabel gearbeitet werden?
- Zeigt das System undefiniertes Verhalten (z. B. Absturz)?
- Kommt es zu Dateninkonsistenz?
- Geht das System nach Rückgang der Überlast wieder in den normalen Bereich zurück?

Für Stresstests sollten speziell geeignete Tools eingesetzt werden.

4. **Test der Benutzeroberfläche:**

Wie ist das Human-Machine-Interface, das Look and Feel der Anwendung? Vor allem für Webanwendungen gibt es hier die Möglichkeiten von Userbefragungen im Labor, auch mittels Eye Tracking. In diesem Fall geht es auch um die Einhaltung von Designkriterien.

Auch Tests der Benutzeroberflächen werden seit einigen Jahren verstärkt automatisiert durchgeführt. Bei JUnit-Tests besteht die Möglichkeit der Anbindung der Testausführung an Continuous-Integration-Systeme (CI-Systeme) wie beispielsweise Jenkins.

Für Webapplikationen kann sich auch ein Test in einem speziellen Labor empfehlen. Spezialisierte Dienstleister bieten dafür Lösungen an, bei denen z. B. mittels Eye Tracking das Look and Feel einer Oberfläche getestet werden kann.

Vor allem bei Internetapplikationen werden dazu auch A/B-Tests oder multivariate Tests verwendet.

10.2 Spezielle Testverfahren

Der **A/B-Test** (auch „split test“) ist eine Testmethode zur Bewertung zweier Varianten eines Systems, bei der die Originalversion gegen eine leicht veränderte Version getestet wird. Diese Methode findet hauptsächlich bei Software und im Webdesign Anwendung mit dem Ziel, eine bestimmte Nutzeraktion oder Reaktionen zu steigern. Im Laufe der Jahre hat sie sich zu einer der wichtigsten Testmethoden im Online-Marketing entwickelt. Mit dem A/B-Test werden aber auch Preise, Designs und Werbemaßnahmen verglichen. Beim A/B-Test wird die Zielgruppe (z. B. Besucher einer Webseite oder Empfänger eines Newsletters) in zwei Untergruppen aufgeteilt: Gruppe A und Gruppe B. Diese Aufteilung muss zufällig erfolgen. Entsprechend der Aufteilung der Zielgruppe wird auch das Testobjekt, wie beispielsweise eine Landingpage oder eine Anzeige, zweigeteilt produziert: die Originalvariante und eine veränderte Variante. Beide Varianten sollten sich nur in einer Komponente unterscheiden, denn nur so können Unterschiede in den Reaktionen eindeutig auf die Änderung zurückgeführt werden. Anschließend wird bei der Gruppe A das Original und bei der Gruppe B die veränderte Version eingesetzt und die Reaktionen verglichen. A/B-Tests bilden dadurch eine einfache Methode zur Auswertung eines Seitendesigns und sind in einer Vielzahl von Situationen nützlich.

Multivariate Tests bedienen sich derselben Grundmechanismen wie A/B-Tests, vergleichen dabei aber eine größere Anzahl an Variablen und liefern mehr Informationen darüber, wie sich diese Variablen gegenseitig beeinflussen. Wie auch bei A/B-Tests wird der Traffic auf der Website zwischen verschiedenen Designvariationen einer Seite aufgeteilt. Der Zweck von multivariaten Tests besteht darin, die Effektivität der verschiedenen Elementkombinationen bei der Erreichung des eigentlichen Ziels zu messen. Wenn eine Seite über ausreichend Traffic für die Durchführung eines solchen Tests verfügt, so lässt sich mit den Daten der einzelnen Variationen nicht nur feststellen, welches das erfolgreichste Design ist, sondern eventuell auch bestimmen, welche Elemente einen besonders positiven oder negativen Effekt auf das Verhalten der Besucher haben.

Beim Test der einzelnen Leistungsmerkmale sind von Anfang an geeignete Metriken zu erheben.

So kann man z. B. die Anzahl der Anforderungen ermitteln, einen Indexwert bilden, wie viele Anforderungen durch Testfälle abgedeckt sind, oder die Anzahl der bereits mittels geeigneter Testprozeduren definierter Testfälle erheben (Kap. 3).

Dadurch schafft man im Testprojekt Transparenz und wird aussagefähig. Im Testkonzept sollte man bereits definieren, welche Parameter man auswerten und welche Metriken man dazu aufsetzen will.

Literatur

- [BLHO2015] : <http://blog.hood-group.com/blog/2015/03/31/der-requirements-manager-als-vorarbeiter-fuer-den-requirements-engineer/>, zugegriffen am 09.05.2020
- [SNEE2012] Sneed, Baumgartner, Seidl: Der Systemtest, Carl Hanser Verlag München 2012

Leistungsmerkmale, die nicht getestet werden

11

Zusammenfassung

Das Testkonzept soll auch darlegen, welche Kriterien nicht nachgewiesen werden und die Testaktivitäten klar abgrenzen, damit keine unrealistische Erwartungshaltung an den Test erzeugt wird. In diesem Zusammenhang ist die Bedeutung von Reviews hervorzuheben.

11.1 Testabdeckung durch Testverfahren

Neben den zu testenden Leistungsmerkmalen muss im Testkonzept auch berücksichtigt werden, welche Kriterien nicht nachgewiesen werden. Hier geht es also um eine Abgrenzung der Arbeiten und eine klare Kommunikation an die Erwartungshaltung des Tests. Mit der Ausklammerung dieser Funktionen entzieht sich das Testteam der Verantwortung für die Funktionsfähigkeit dieser Funktionen und erzwingt eine Festlegung an anderer Stelle, wer für die Qualitätssicherung der Funktionen verantwortlich sein soll [SNEE2012].

Man muss im Testkonzept beschreiben, ob evtl. Lieferanten für bestimmte Tests verantwortlich sind und eine bereits getestete Komponente als definiertes Produkt innerhalb des Testobjektes mit eingebunden wird. In diesem Fall entfällt der Test dieser Komponenten.

Häufig ist es leider so, dass die vertraglichen Regelungen mit den Lieferanten nur unzureichend auf die Notwendigkeit einer Vorkontrolle eingehen. Es ist aber gerade wichtig, ein gemeinsames Verständnis im gesamten Lieferprozess hinzuarbeiten. Schon bei der Ausschreibung eines Projektes sollte optimaler Weise ein Testmanager hinzugezogen werden und definierte Tests vom Lieferanten eingefordert werden, um den geforderten Qualitätsstandard gegenüber den eigenen Kunden erfüllen zu können. Dabei geht es auch um die Abgrenzung der Testaktivitäten zwischen Auftraggeber und Lieferant, damit nicht bestimmte Funktionen doppelt, andere dafür gar nicht getestet werden.

Gründe, auf bestimmte Tests zu verzichten und dadurch eine geringere **Testabdeckung** zu erzielen, sind zum Beispiel:

- Ressourcenmangel,
- Terminrestriktionen,
- Budgetbeschränkung oder
- technische Voraussetzungen.

Dabei muss aber auch bereits im Testkonzept deutlich gemacht werden, welche Konsequenzen und Risiken sich aus der bewusst gewählten geringeren Testabdeckung ergeben. Es empfiehlt sich, diese Risiken in EUR umzurechnen, denn mit prognostizierten Zahlen kann dieses Risiko erheblich transparenter dargestellt werden.

11.2 Nachweise durch andere Methoden

Es geht aber auch darum, Nachweise zu dokumentieren, die anderweitig zu treffen sind, also nicht durch einen Softwaretest, sondern zum Beispiel durch ein Dokumentenreview.

Ein **Review** ist eine formell organisierte Zusammenkunft von Personen zur inhaltlichen oder formellen Überprüfung eines Produktteils oder Produktes. Die Überprüfung erfolgt nach vorgegebenen Prüfkriterien und Prüflisten.

Ein Review kann nur dann zu guten Ergebnissen führen, wenn die Prüfunterlagen rechtzeitig zur Verfügung stehen und wenn die beteiligten Personen die erforderliche Kompetenz besitzen. Eine Systematik wird erreicht, wenn für die Prüfung Checklisten zur Verfügung gestellt werden. Ebenfalls ist die anschließende Überwachung von Bedeutung, damit alle gefundenen Mängel auch behoben werden.

Reviews mit ihren besonderen Ausprägungen der Inspektionen und Walkthroughs dienen aber nicht nur zum manuellen Prüfen von Dokumenten. Testergebnisse, die auch nicht formal sind, oder Quellcode kann dabei zusätzlich zu anderen Testarten überprüft werden. Die Inspektion von Programmen als spezielle Form des Reviews ist unter dem Namen **Code-Inspektion** bekannt.

Haupteinsatzgebiet von Reviews ist die Überprüfung von Dokumenten.

Beispiele dafür sind

- Systemdokumentationen:
 - Anforderungsspezifikationen (Requirements),
 - Machbarkeitsstudien,
 - Dokumente für Systemanalyse,
 - Entwurfsdokumente,
 - Programmbeschreibungen,
 - Testentwürfe und
 - Bedienungsanleitungen;

- Anleitungen für den Systemadministrator.

Durch Reviews angestrebte Ziele sind

- die Entdeckung von Fehlern und damit die Vermeidung von Folgekosten,
- die Dokumentation des Projektfortschritts,
- die Verbreitung des Know-hows im Projektteam sowie
- die Konsenserziehung im Projektteam.

Dabei werden während des Reviews keine Lösungen diskutiert [GOLL2011].

Ein weiterer Sonderfall, bei dem bewusst auf Nachweise im Testumfeld verzichtet wird, besteht dann, wenn der Einsatz der Software teilweise erst im Rollout unter Echtbedingungen möglich ist.

Vor allem im Bereich der Logistik oder industrieller Fertigung, z. B. wenn eine Maschinensteuerung produktiver Systeme vom Test betroffen ist, ist die Bereitstellung einer eigenen Testumgebung zu aufwändig. Die Kosten für die zusätzlich benötigte Hardware wären in diesem Fall zu hoch. In diesem Fall wird etwa an einem Standort ein System evaluiert, das zu einem späteren Zeitpunkt auf weitere Standorte ausgerollt wird. Die Testaktivitäten werden in diesem Fall am Pilotstandort in der Produktion durchgeführt, etwa nur an einer von mehreren Sortieranlagen oder in der Fertigung an einer von mehreren Produktionsstraßen. In diesem Fall ist es wichtig, die Risiken hervorzuheben und die geringere Produktivität des Pilotstandorts während der Testphase zu berücksichtigen. Dieses Vorgehen ist auch nur möglich, wenn der Pilotstandort und der geplante Einsatz intensiv vor Ort betreut werden. Dafür sind ebenfalls genaue Einsatzpläne und Meilensteine zu erarbeiten. Es ist auch zu prüfen, inwiefern der Test die Produktion ggf. behindert und welche Risiken das bedeutet.

Die gewählte Referenzanlage muss in diesem Fall den Echtbetrieb möglichst gut abbilden. Wenn es hier an anderen Standorten lokale Besonderheiten gibt, ist der Testbetrieb auch an diesen anderen Standorten durchzuführen. Offizielle Abnahmeprozesse und Softwarefreigaben sind dabei aber genauso notwendig, als wenn der Test in einer Testumgebung durchgeführt worden wäre.

Literatur

- [SNEE2012] : Sneed, Baumgartner, Seidl: Der Systemtest, Carl Hanser Verlag München 2012
[GOLL2011] : Joachim Goll: Methoden und Architekturen der Softwaretechnik, Vieweg+Teubner Verlag Wiesbaden 2011

Zusammenfassung

Der sinnvolle Umfang der Testaktivitäten muss im Rahmen der Testplanung erhoben werden. Im Rahmen der Priorisierung von Testfällen sind Testfälle, die bei fehlerhafter Funktionalität hohe Risiken aufweisen und Testfälle zum Nachweis häufig vorkommender Geschäftsprozesse, vorzuziehen.

12.1 Testumfang

Im Rahmen des Testkonzeptes muss man sich Gedanken machen, wie hoch der sinnvolle **Testumfang** des betrachteten Projektes ist.

Das folgende Beispiel zeigt, dass ein komplettes „Austesten“ eines Programmes so gut wie unmöglich ist:

- Es gibt genau x Möglichkeiten, ein Programm auszuführen, und wenn alle x Möglichkeiten im Test getestet werden, ist das Programm komplett ausgetestet.
- Angenommen, eine Prozedur erwartet 2 Integerwerte als Eingabedaten.
- Bei 32-Bit-Länge ergeben sich bereits $2^{32} \times 2^{32}$ Kombinationen von Eingabewerten (ungefähr $1,8 \times 10^{19}$)
- Selbst wenn ein automatischer Testtreiber in 1 ms einen Testfall ausführen kann, dauert der vollständige Test aller Eingabewerte ca. 570 Millionen Jahre!

Wenn man sämtliche Kombinationen der Software testen würde, würde man also zumindest Hunderte und Tausende von Testfällen behandeln müssen. Mit einer Testautomatisierung kann man zwar manuelle wiederholte Tätigkeiten vermeiden, aber es bleibt der Aufwand der Automatisierung, der Pflege, der Aktualisierung, der einmaligen Erstellung der Skripte und

der zielgerichteten Auswertung der Ergebnisse. Auch eine Testautomatisierung ist kein Selbstläufer, sondern bedingt einen hohen organisatorischen und administrativen Aufwand.

Ein Softwaretest ist kein mathematischer Beweis. Ein Test soll die Funktionalität dahingehend nachweisen, dass die Geschäftsvorfälle in der Praxis den Erwartungen entsprechen, und dabei sollte man durchaus auch Kombinationen prüfen, die im täglichen Betrieb selten vorkommen. Die optimale Testabdeckung bedeutet aber nicht, jeden nur erdenklichen Fehler aufzuspüren, der in vielen Jahren vielleicht einmal auftritt, vielleicht aber auch im Produktivbetrieb ohnehin nie entdeckt worden wäre. Ein Softwareprojekt hat immer Restriktionen durch Termine und Kosten, daher kommt der Fall, dass man am Ende zu viele Testfälle getestet und eine zu große Testabdeckung hat, nach meinen Erfahrungen in der betrieblichen Praxis ohnehin so gut wie nie vor. In den meisten Fällen hat man am Projektende eher zu wenig Tests durchgeführt, und regelmäßig kommt es vor, dass man zumindest vereinzelte Fehler doch erst nach der Produktionseinführung bemerkt.

Teilweise sind inkrementelle Lieferungen der Software vorgesehen, vor allem im agilen Umfeld. Daher ist es dann oft gar nicht möglich, bei jeder Lieferung den gesamten vorgesehenen Testumfang zu testen, sondern z. B. nur einmalig kurz vor Produktauslieferung.

Aufgrund dieser Beschränkungen ist es notwendig, Testfälle zu priorisieren.

12.2 Priorisierung beim Systemtest und beim Abnahmetest

Die Priorisierung der Testfälle für den Systemtest ist im Detail beim Design der Testfälle vorzunehmen. Im Testkonzept sollte man aber darauf hinweisen, was die Kernfunktionen sind, auf welche Bereiche im Test ein besonderer Focus gerichtet sein soll.

Hin und wieder werden theoretisch existierende Testfälle, die in der Praxis aber kaum vorkommen, sehr detailliert getestet und man verliert sich dabei in akademischen Diskussionen, aber andere Geschäftsvorfälle, die weitaus häufiger vorhanden sind, vernachlässigt. Diesem Umstand sollte man bereits im Vorfeld begegnen. Daher ist es auch ratsam, vor allem bei Anwendungen, die im eigenen Unternehmen auch angewendet werden, einen Anwender aus der Praxis von Anfang an hinzuzuziehen, der die betrieblichen Abläufe genau kennt und den Fokus auf die entscheidenden Geschäftsprozesse legt. Das verhilft auch bei Pilotierung und Einführung der neuen Software von Anfang an zu mehr Akzeptanz.

Beim Abnahmetest werden ohnehin nur ausgewählte Testfälle dem Kunden vorgeführt oder bereits einige in der Testumgebung durchgeführte Testfälle auf einer produktionsnahen Umgebung wiederholt. Der Abnahmetest ist in der Regel auf höchstens einige Werkzeuge zeitlich eng begrenzt, wodurch sich eine Konzentration auf ausgewählte häufig durchgeführte Geschäftsvorfälle von selbst ergibt.

12.3 Priorisierung beim Modultest und Integrationstest

Vor allem beim Modultest und beim Integrationstest benötigt man zum Test neu entwickelter Software in der Regel zusätzliche Software. Daher ist die Reihenfolge beim Test teilweise von Anfang an gegeben. Sollte man aber frei entscheiden können, in welcher

Reihenfolge zu testen ist, sollte man mit dem Baustein beginnen, der am wenigsten Abhängigkeiten hat. Unter einem Baustein versteht man beim Methodentest weitere Methoden, beim Klassentest weitere Klassen und beim Integrationstest weitere Komponenten.

Abb. 12.1 stellt eine entsprechende Ausgangssituation dar, in der verschiedene Bausteine als Kästen mit ihren Abhängigkeiten dargestellt werden. Bei den Abhängigkeiten wird nicht der möglichst zu erreichende Fall skizziert, dass sie nicht zyklisch sind, also z. B. A nicht auf weiteren Wegen von B oder D von B abhängig ist, sodass man dadurch wieder den Pfeilen folgend zu A gelangen könnte.

Hat man die freie Auswahl, sollte man im Beispiel mit dem Test von Baustein G anfangen, da dieser von keinen anderen Bausteinen abhängig ist. Danach testet man nur Bausteine, die ausschließlich von bereits getesteten Bausteinen abhängen. Dabei können durchaus verschiedene Möglichkeiten für die Reihenfolge, wie z. B. F-D-A-H-E-C-B oder F-H-D-E-A-C-B vorhanden sein. Die Auswahl der geeigneten Reihenfolge hängt in diesem Fall von der Reihenfolge der Fertigstellung der einzelnen Blöcke ab.

Generell gilt beim Test, dass man sich ausschließlich auf die zu testende Funktionalität konzentriert und die restliche Funktionalität als korrekt annimmt. Nachdem z. B. Baustein G getestet wurde, geht man bei der Testfallerstellung für die Komponente F davon aus, dass G korrekt und schreibt keine weiteren Testfälle, um diese Korrektheit erneut zu überprüfen. Dies ist vergleichbar damit, dass man die Korrektheit von Klassen aus einer Java-Bibliothek als gegeben annimmt. Erst wenn man es für wahrscheinlich hält, dass ein vorher getesteter Baustein doch einen Fehler enthält, kann man weitere Tests für diesen Baustein entwickeln. Das gilt analog für den Test von Schnittstellen, bei denen man sich ausschließlich auf diese Schnittstellen an sich, aber nicht auf die Bausteine konzentriert, deren Funktionalität ohne die Schnittstellen ohnehin bereits überprüft wurde.

Oftmals ist es in der Praxis aber so, dass der vorgestellte sinnvolle Ansatz aus verschiedenen Gründen nicht nutzbar ist. Das ist z. B. dann der Fall, wenn sich die Erstellung von G als komplexer herausstellt oder die Ressource, die G erstellen sollte, einfach nicht fertig geworden ist (wegen anderer Prioritäten, Ausfälle ...), der Entwickler von D aber unbedingt eine Möglichkeit zum Test benötigt, um die Lieferung seiner Komponente abzu-

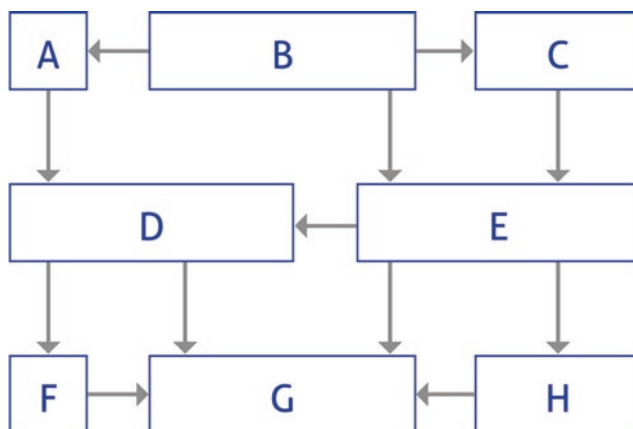


Abb. 12.1 Visualisierung von Abhängigkeiten von Bausteinen/Objekten

schließen. Noch eine Stufe komplexer wird es, wenn die Komponente D auch im Rahmen eines anderen Projektes dringend benötigt wird, sodass die Priorität auf deren Fertigstellung liegt und G dafür terminlich verschoben wurde. In diesen Fällen sind Mocks oder Stubs zu entwickeln, was einen zusätzlichen Aufwand darstellt.

Häufig ist es bei komplexen Systemen aber auch der Fall, dass gewisse Bausteine nur sehr aufwändig getestet werden können, da es sich z. B. um Datenbanken oder Webserver handelt. In unserem Beispiel könnten die Bausteine A, B und C z. B. für eine Weboberfläche stehen, die mit einem Server verbunden ist, auf dem die fachliche Logik in den Bausteinen D und E realisiert wird. Die Fachlogik nutzt dann die Bausteine F, G und H zur effizienten Datenhaltung. Auch hier muss zunächst der Ansatz versucht werden, die Schichten getrennt zu testen und später bei der Verknüpfung der einzelnen Schichten davon ausgehen, dass die vorher getestete Kernfunktionalität korrekt ist. Gerade der Ansatz, möglichst die gesamte Funktionalität über die Oberfläche zu testen, scheitert nämlich oft aus verschiedenen Gründen. Häufig kann nicht die gesamte Funktionalität einfach über die Oberfläche angesprochen werden und oft ist es technisch sehr aufwändig, für alle Tests die benötigte Hard- und Software für jeden Tester zur Verfügung zu stellen. Auch in diesem Fall kann es sinnvoll sein, auf **Mock-Objekte** zurückzugreifen, die dann die benötigten Teile der verwendeten Bausteine simulieren. Mock-Objekte implementieren die Schnittstellen, über die das zu testende Objekt auf seine Umgebung zugreift. Sie stellen sicher, dass die erwarteten Methodenaufrufe vollständig, mit den korrekten Parametern und in der erwarteten Reihenfolge durchgeführt werden. Das Mock-Objekt liefert keine Echtdaten zurück, sondern vorher zum Testfall passend festgelegte Werte. Das Mock-Objekt kann somit dazu verwendet werden, ein bestimmtes Verhalten nachzustellen. Deswegen sind vor allem Datenbanken, Netzwerkverbindungen oder angeschlossene, eventuell teure externe Software damit interessante Kandidaten für eine Umsetzung als Mocks, um den Test von Software effizienter zu gestalten.

Dabei stellt sich die Frage, ob alle Tester im Modultest Experten in Frameworks sein müssen. Dies ist bei größeren Projekten typischerweise nicht der Fall, da die Mock- oder Framework-Nutzung so von den anderen Testern abgeschlossen wird, dass diese nur noch das Wissen darüber besitzen müssen, was getestet werden soll. Diese Abstraktion der konkreten Test-Frameworks wird auch **Testarchitektur** genannt. Neben der durch die Abstraktion ermöglichten wesentlich effizienteren Testentwicklung spielt auch die Chance eine wichtige Rolle, dass man bei kleineren Änderungen im Umfeld der zu testenden Software oder des benutzten Frameworks nur eine bzw. wenige Stellen ändern muss und so die meisten Tests und Tester von den Änderungen nicht betroffen sind [KLEU2018].

12.4 Priorisierung nach unterschiedlichen Testarten

Bei der Priorisierung der verschiedenen Testarten kann man weiterhin festlegen, welche Testtypen mit welcher Priorität eingesetzt werden sollen. Je nachdem, welche Testziele dabei verfolgt und welche Qualitätsanforderungen an die Webanwendung gestellt werden und in welchem technischen Umfeld sie betrieben werden soll, sind die Tests zu planen und durchzuführen.

So ist zum Beispiel für eine Website, die hauptsächlich Informationen bereitstellt, ein Auffindbarkeitstest von größerer Bedeutung als ein Test der Ausfallsicherheit. Wird eine Applikation in einem Intranet betrieben, in dem alle Benutzer ohnehin den gleichen Browser verwenden, ist ein Browsertest über alle existierenden Browser nicht notwendig. Wenn aber der Nutzerkreis aus der weltweiten Web Community besteht, haben Browser- und Zugänglichkeitstest eine höhere Priorität.

In Tab. 12.1 werden die einzelnen Testtypen im Webtest für ein beispielhaftes Testobjekt exemplarisch bewertet.

Das Beispiel basiert auf einem Finanzierungsrechner, der die Monatsraten für eine Autofinanzierung berechnet und in einem Webportal betrieben wird.

Tab. 12.1 Beispielhafte Bewertung von Testtypen für eine spezielle Anwendung [FRAN2007]

Testtyp	Bewertung	Begründung
Dokumententest	Hoch	Anforderungsspezifikationen enthalten geschäftsrelevante Funktionsbeschreibungen
Tests zur Funktionalität		
Klassentest	Niedrig	Intensiver Komponententest durch das Testteam
Komponententest	Hoch	Geschäftsrelevante Funktionen
Integrationstest	Hoch	Integration geschäftsrelevanter Subsysteme
Funktionaler Systemtest	Hoch	Geschäftsrelevante Funktionen
Linktest	Niedrig	Keine externen Links vorgesehen
Cookietest	Hoch	Geschäftsrelevante Funktionen setzen Cookies ein
Plugintest	Niedrig	Nur zum Lesen von PDF-Dokumenten werden Plugins benötigt
Sicherheitstest	Hoch	Verarbeitung sensibler Daten, Durchführung geschäftsrelevanter Transaktionen
Tests zur Benutzbarkeit		
Contenttest	Hoch	Geschäftsrelevanter Web-Auftritt
Oberflächentest	Hoch	Unternehmensstandards sind einzuhalten
Browsertest	Mittel	Benutzer des Firmenportals setzen in der Regel nur einen bestimmten Browser ein
Usabilitytest	Hoch	Kundenzufriedenheit ist ein kritischer Erfolgsfaktor
Zugänglichkeitstest	Mittel	Zugänglichkeit nicht explizit gefordert
Auffindbarkeitstest	Niedrig	Benutzer kommen über das gut besuchte Firmenportal
Tests zur Änderbarkeit und Übertragbarkeit		
Codeanalysen	Mittel	Nur stichprobenweise Codeinspektionen notwendig
Installationstest	Nicht relevant	Anwendung wird im Hause betrieben, keine Clientinstallationen notwendig
Tests zur Effizienz und Zuverlässigkeit		
Performanz-/Lasttest	Hoch	Verarbeitung von geschäftsrelevanten Transaktionen, Performanz ist daher kritischer Erfolgsfaktor
Ausfallsicherheitstest	Niedrig	Sicherheitsvorkehrungen zum bestehenden Firmenportal sind etabliert und geprüft
Verfügbarkeitstest	Hoch	24-Stunden-Verfügbarkeit gefordert

12.5 Risikobasiertes Testen

Der erschöpfende Test von Softwaresystemen ist aufgrund komplexer Schnittstellen und Umgebungsbedingungen in der Realität kaum möglich. Zusätzlich schränkt das vorhandene Testbudget die Anzahl der durchführbaren und auswertbaren Testfälle ein. Deshalb sind eine Auswahl zweckmäßiger Testfälle und deren Priorisierung für die Reihenfolge der Testausführung notwendig. Ein vielversprechender Ansatz zur Priorisierung von Testfällen ist das risikobasierte Testen. Ein risikobasierter Testplan stellt sicher, dass Teile einer Software umso intensiver getestet werden, je höher das Risiko ist, dass diese Teile beim Einsatz der Software zu einem nicht vernachlässigbaren Schaden führen. Beim risikobasierten Testen werden also die Testfälle insbesondere hinsichtlich des Schadensmaßes bei Fehlfunktionen gewichtet. Je höher die Gewichtung des zu testenden Prozesses und damit der Umfang eines möglichen Schadens, desto höher ist die Priorität des Testfalls [INNO2014]. Üblicherweise müssen Testfälle der Priorität „hoch“ in allen Testzyklen und jene der Priorität „mittel“ mindestens in einem Testzyklus erfolgreich durchgeführt werden. Testfälle der Priorität „gering“ sind „Nice to Have“.

Eine direkte Priorisierung von Testfällen auf Basis dieses Risikos ist bei komplexen Softwaresystemen jedoch aufwendig und jede Aktualisierung der Risikobewertung erfordert eine erneute Priorisierung der gesamten Testfallmenge [FRAU2008].

Nach wie vor kommt es in vielen Projekten vor, dass zwar die Testfälle spezifiziert sind, die Zeit zur Durchführung aller spezifizierten Testfällen aber nicht vorhanden ist. Das ist vor allem dann häufig zu beobachten, wenn das Testteam zwar zu Beginn des Projektes wie geplant zur Verfügung stand, das Entwicklungsteam aber (aufgrund Ressourcenengpässen oder komplexen Anforderungen) Verzögerungen bei der Entwicklung hat, sodass eine testfähige Version erst spät im Projektzeitraum überhaupt zur Verfügung steht. Oftmals ist dabei auch keine Zeit vorhanden, diese Testfälle zu automatisieren, da sich der Aufwand der Testautomatisierung eher mittel- bis langfristig niederschlägt und nicht für das laufende Projekt, dessen Zeitbudget ohnehin schon knapp bemessen ist. Dann ist eine sinnvolle Auswahl an Testfällen zu treffen, um sicherzustellen, dass trotz der eingeschränkten Anzahl an durchzuführenden Testfällen noch möglichst viele kritische Fehlerwirkungen gefunden werden. Die Priorisierung muss also in der Weise durchgeführt werden, dass bei einer (vorzeitigen) Beendigung der Tests zu einem beliebigen Zeitpunkt das bis dahin bestmögliche Ergebnis erreicht wird.

Eine Priorisierung hat darüber hinaus den Vorteil, dass die wichtigsten Testfälle zuerst zur Ausführung kommen und schwer wirkende Fehlerwirkungen frühzeitig erkannt und behoben werden können. Es empfiehlt sich generell, die Dauer der durchschnittlichen Fehlerbehebung – also Dauer von Erfassung bis erfolgreichem Nachtest eines Fehlers – zu erheben. Aus meiner eigenen Erfahrung heraus ist das Management oft überrascht, dass dabei in aller Regel durchschnittliche Werte von 3–4 Wochen pro Fehler erhoben werden. Man unterschätzt immer wieder die Liegezeiten bei der Prozessdurchführung.

Eine Gleichverteilung nach dem Gießkannenprinzip auf alle Testobjekte des Projektes ist wenig sinnvoll, da bei einem solchen Vorgehen kritische und unkritische Programmteile mit der gleichen Intensität getestet werden. Bei den kritischen Teilen ist dann nicht ausreichend genug getestet worden, während bei den unkritischen Programmteilen Ressourcen unnütz verschwendet wurden [SPIL2007].

Literatur

- [KLEU2018] : Stephan Kleuker, Qualitätssicherung durch Softwaretests, Springer-Vieweg, Wiesbaden 2018
- [FRAN2007] : Klaus Franz, Handbuch zum Testen von Web-Applikationen, Springer-Verlag, Berlin Heidelberg 2007
- [SPIL2007] : Basiswissen Softwaretest, dpunkt Verlag Heidelberg, 3.Auflage 2005 korrigierter Nachdruck 2007
- [FRAU2008] : <http://publica.fraunhofer.de/dokumente/N-76418.html>, zugegriffen am 10.05.2020
- [INNO2014] : https://www.innobis.de/uploads/tx_news/2014_12_Banken_Sparkassen_26f.pdf, zugegriffen am 10.05.2020

Zusammenfassung

Eine permanente Testorganisation übernimmt definierte Testdienstleistungen für mehrere Organisationseinheiten. Ziel der permanenten Testorganisation ist die Beschleunigung von Abläufen bei geringeren Kosten und Steigerung der Qualität. Die Aufteilung der Aufgaben zwischen permanenter Testorganisation und projektbezogenen Testaktivitäten wird in Testrichtlinien festgelegt,

13.1 Wesen der permanenten Testorganisation

Organisationen können entscheiden, eine **permanente Testorganisation** einzurichten, die vielerlei Testdienstleistungen anbietet. Im Gegensatz zu projektorientiertem Testen richtet eine permanente Testorganisation ein bestimmtes Element eines Testprozesses projektübergreifend ein, was jedoch nicht bedeutet, dass nicht auch innerhalb der Projekte Tests durchgeführt werden.

Welche Aktivitäten von einer permanenten Testorganisation durchgeführt werden, unterscheidet sich je nach Organisation. So kann etwa die Vorbereitung und Durchführung von Tests der permanenten Testorganisation übertragen werden, aber auch die Einrichtung und Wartung der Testumgebung oder der Testwerkzeuge. Im Ergebnis bedeutet dies, dass jeder Prozess im Zusammenhang mit einem bestimmten Testteil (Testdurchführung, Testumgebung, Testwerkzeuge usw.) mit einer festen Methode und mit wiederverwendbaren Hilfsmitteln durchgeführt wird. Die permanente Testorganisation stellt diese Elemente für Projekte als sogenannte **Testdienstleistungen** bereit. Der Prozess wird, unabhängig vom jeweiligen Projekt, die gleiche Qualität haben.

Es hängt vom Auftraggeber ab, wie die einzelnen Elemente des Testprozesses von der permanenten Testorganisation ausgearbeitet und welche Testdienstleistungen zur Verfü-

gung gestellt werden. Der Auftraggeber kann zum Beispiel ein Tester sein, der eine Umgebung benötigt, um manuelle Testskripte zu automatisieren, oder der Eigentümer einer Anwendung, der bestimmte Regressionstests benötigt.

Abgesehen vom Auftraggeber kann auch die Zielsetzung einer permanenten Testorganisation die Testdienstleistungen bestimmen. So kann die Organisation etwa einen Auftrag von außen erhalten haben („Stellen Sie sicher, dass alle Testumgebungen effizient eingerichtet sind und gewartet werden“ oder „Bieten Sie eine Lösung für alle Fragen im Zusammenhang mit der Testautomatisierung“). Eine Organisation kann sich aber auch selbst ein Ziel setzen („Wir führen die Regressionstests durch“). Das Ziel einer Organisation kann sich im Laufe der Zeit ändern, ebenso wie die angebotenen Testdienstleistungen. Die Organisation wird durch die Zuordnung von Aufgaben, Berechtigungen und Zuständigkeiten zu Jobs für die Mitarbeiter festgelegt. Zudem werden Prozesse für eine Organisation allgemeingültig definiert, eingerichtet und ausgewählt, sodass die Abläufe möglichst optimal funktionieren. Diese Festlegungen sind pro Organisation unterschiedlich, dafür existieren keine festen Regeln.

Ziel der permanenten Testorganisation ist, die Durchlaufzeit und Kosten im Testprozess zu reduzieren und gleichzeitig die Qualität des gesamten Workflows zu erhöhen. Spezifische Elemente des Testprozesses sind dabei für alle Projekte übergreifend zu organisieren, um folgende Vorteile daraus zu gewinnen:

- **Optimale Nutzung von (knappen) Fachkenntnissen:** Oft fehlt es in einer Organisation an Testexperten. Es fehlen nicht nur Fachkenntnisse zum Erstellen und Durchführen von Testskripten, sondern auch beispielsweise für die Anwendung von Testwerkzeugen oder die Einrichtung einer Testumgebung. Kombiniert man hingegen das gesamte verfügbare Fachwissen und fasst alle Nachfragen nach diesem Fachwissen an einer Stelle zusammen, kann man eine Reihe problematischer Punkte optimieren. Es ist einfacher, an benötigtes Expertenwissen heranzukommen, wenn man Einblick in das gesamte verfügbare Expertenwissen hat. Zudem kann das vorhandene Fachwissen wesentlich besser den einzelnen Problembereichen zugeordnet werden. Mitarbeiter, die z. B. über Spezialwissen zu Testumgebungen verfügen, können dadurch für mehrere Projekte als Dienstleister zur Verfügung stehen.
- **Vorhersagbare Produktqualität:** Die von der permanenten Testorganisation angebotenen Dienstleistungen sind standardisiert. Es gibt Produkte und Prozesse mit zugehörigen Standards, die Dienstleistungen weichen nur minimal voneinander ab, sodass die Servicequalität vorhersehbar ist. So wird zum Beispiel die Dienstleistung „Einrichtung einer Testumgebung“ zu einer Routineaufgabe, die mit allen zum Prozess gehörigen Vorlagen, Werkzeugen, Techniken und Checklisten einfach aus der Schublade gezogen werden kann und von erfahrenen und qualifizierten Mitarbeitern ausgeführt wird.
- **Kurze Anlaufzeit:** Durch die Standardisierung der Prozesse beschränkt sich die Anlaufzeit auf ein Minimum. Es ist weder erforderlich, vorab Schulungen zu besuchen, noch irgendwelche Teile auszuprobieren oder das günstigste Vorgehen auszuwählen. Die Vorbereitungszeit wie etwa die Einrichtung eines Testmanagementwerkzeugs kann

auf ein Minimum reduziert werden: alles ist sofort verfügbar, nur die speziellen Wünsche und Anforderungen des Kunden müssen ermittelt und entsprechend parametrisiert werden.

- **Kontinuierliche Verbesserung der Prozesse, eingebettet in die Organisation:** Die Testorganisation ist für die angebotenen Prozesse verantwortlich und führt sie aus. Dabei kann festgelegt werden, dass nach jedem durchgeführten Prozess ein Assessment angeschlossen wird. Die Erfahrung aus diesem Assessment wird anschließend in die Organisation eingebracht und in eine verbesserte Version der Dienstleistung integriert. Ein solches Assessment und die Verarbeitung der Resultate müssen in die Prozesse der Testorganisation eingebettet werden.
- **Konsolidierung und Entwicklung von Erfahrungen:** Aus der Kombination des vorhandenen Fachwissens mit der oben genannten Auswertung und Verarbeitung von Erfahrungen erhält man einen anhaltenden Lerneffekt. Dies wird ferner durch Synergieeffekte zwischen einzelnen Mitarbeitern verstärkt.
- **Leichter planbare Kosten und Durchlaufzeiten:** Durch die Standardisierung der Aufgaben und die permanente Auswertung, Verarbeitung und Verbesserung der Arbeiten erhält man einen Lerneffekt und verstärkt positive Synergieeffekte zwischen einzelnen Mitarbeitern. In der Regel besteht auch eine höhere Flexibilität innerhalb der Organisation, wenn man Mitarbeiter kurzfristig mehreren unterschiedlichen Projekten zuordnen kann. Dadurch können im optimalen Fall die Testzeit verkürzt und Leerlaufzeiten vermieden werden.
- **Kostenreduzierung durch Zentralisierung und Skalierung:** Wird an unterschiedlichen Stellen in einer Organisationseinheit getestet, bedeutet das oft auch, dass mit externen Anbietern an unterschiedlichen Stellen zu verhandeln ist. Zentralisiert man alle testrelevanten Aktivitäten, kann die Testorganisation Rahmenverträge mit allen Anbietern, z. B. von Werkzeugen oder Testpersonal, abschließen. Damit sind Skaleneffekte verbunden, durch den die Verhandlungsposition gestärkt und bessere Konditionen möglich werden.

13.2 Bedingungen für eine permanente Testorganisation

Nicht für jede (IT-)Organisation ist eine permanente Testorganisation möglich, denn die Organisation muss einige Bedingungen erfüllen:

- Der Testaufwand einer permanenten Testorganisation muss eine gewisse Größenordnung erreichen, sodass sie nur für größere Unternehmen generell infrage kommt.
- Im Rahmen der Unternehmenskultur muss es für die permanente Testorganisation formal möglich sein, konkrete Vereinbarung mit ihren Kunden treffen zu können.
- Die Organisation muss mit wiederkehrenden Prozessen und Projekten arbeiten, damit die Standardisierung von Dienstleistungen für den Testbereich möglich ist.

- Die Organisation muss in der Lage sein, mit einer zentralen Organisation, so wie es eine permanente Testorganisation darstellt, umzugehen und diese zuzulassen. Zentrale Organisationen werden (z. B. aufgrund ihrer Größe) ggf. als Bedrohung wahrgenommen. Das ist zu vermeiden.

13.3 Kritische Punkte für permanente Testorganisationen

Nachdem eine permanente Testorganisation aufgebaut und begonnen wurde, müssen eine Reihe von Punkten kontinuierlich überwacht werden. Die folgenden kritischen Punkte sind Voraussetzung für den anhaltenden Erfolg der Organisation:

- **Testdienstleistungen:** Es muss immer wieder verifiziert werden, ob die von der permanenten Testorganisation angebotenen Dienstleistungen noch dem entsprechen, was die Auftraggeber benötigen. Das geforderte Qualitätsmaß legt der Auftraggeber fest.
- **Testprofessionalität:** Die Professionalität der Testorganisation basiert einerseits aus dem Wissen und der Fachkompetenz der Tester, andererseits auf der Stabilität des Testpersonals in der Organisation.
- **Wiederverwendung:** Häufig werden Kosteneinsparungen als einer der wichtigsten Gründe für eine permanente Testorganisation genannt. Das kann durch die Wiederverwendung etwa von Testware, Testdaten und Testinfrastruktur erreicht werden. Dabei ist ständig auf die tatsächliche Wiederverwendung und ihre effiziente und optimale Organisation zu achten. Vor allem die Bereitstellung der Testinfrastruktur ist dabei erfahrungsgemäß eine kritische Größe.
- **Autonomie:** Für eine permanente Testorganisation ist es wichtig, die bereitgestellte Software oder Hardware unabhängig vom Rest der IT-Organisation objektiv zu beurteilen. Gleichzeitig bleibt die Testorganisation Teil einer übergeordneten Einheit mit übergeordneten Interessen. Diese Situation bringt eine beträchtliche Herausforderung und kann zu Interessenkonflikten führen.
- **Verhindern einer Abkapselung:** Zentralisiert man die Testaktivitäten, schafft man eine eigene Einheit innerhalb der IT-Organisation. Daraus könnte eine Abkapselung vom gemeinsamen Ziel der gesamten Organisationseinheit entstehen, die für die Zusammenarbeit zwischen Entwicklern und Testern kontraproduktiv ist.

13.4 Testrichtlinien

Die in einem Unternehmen definierten Testrichtlinien bestimmen, welche Elemente des Testprozesses der permanenten Testorganisation zugewiesen werden und ihre Struktur.

Die Testrichtlinien müssen für alle Arten von Systemen, Infrastrukturen und Entwicklungsmethoden gelten. Da Testen eines der Werkzeuge zur Qualitätssicherung ist, müssen die Testrichtlinien auch an den anderen Richtlinien und Initiativen mit Bezug zum Quali-

tätsmanagement ausgerichtet sein. Die Testrichtlinien müssen daher mit den strategischen, taktischen und betrieblichen Unternehmensrichtlinien in Übereinstimmung stehen. Auf der strategischen Ebene muss die Bedeutung der Unternehmensrichtlinien im Hinblick auf das Testen für das gesamte Unternehmen festgelegt werden. Dies führt zu strategischen Testrichtlinien, die von dieser Ebene aus eingeführt und unterstützt werden müssen. Auf der taktischen Ebene müssen die Testrichtlinien (je nach Aufbau des betreffenden Unternehmens) in alle Unternehmensteile, Abteilungen, Produktgruppen, Programme oder Projekte übertragen werden. Das schließt auch Ressourcen und Budgets zur klaren und unmissverständlichen Umsetzung der Testrichtlinien mit ein.

Eine konsistente Implementierung der Testrichtlinien führt zu einem einheitlichen Testansatz auf operativer Ebene:

- **Strategisch:** Die strategischen Richtlinien einer IT-Organisation haben Einfluss auf alle untergeordneten Organisationsbereiche und deren Aktivitäten, einschließlich des Testens. Strategische Richtlinien können viele Formen annehmen. So können sie beispielsweise Bedingungen für die interne IT-Organisation festlegen, aber auch für die Qualitätsziele und die Möglichkeiten, diese zu erreichen. Die strategischen Richtlinien lassen sich durch Anregungen und Anforderungen von innerhalb einer Organisation formen, aber auch externe Faktoren können eine Rolle spielen. Beispiele dafür sind Anforderungen von externen Überwachungs- und Regulierungsorganen, gesetzliche Anforderungen, die zu erfüllen sind, oder Industrievereinbarungen und Normen.
- **Taktisch:** Auf der taktischen Ebene werden die strategischen Richtlinien in ihre operativen Entsprechungen übersetzt. Dies erreicht man durch die Definition von Regeln, die die Vorbedingungen und Standards festlegen, nach denen sich Personen, Mittel und Methoden zu richten haben, um die strategisch definierten Ziele zu erreichen. Dieses Regelwerk beschreibt, wie ein strukturierter Testansatz innerhalb einer Organisation, Abteilung, Produktgruppe, Programm oder eines Projektes (anhängig vom Aufbau der betrachteten Organisation) eingerichtet werden soll.
- **Operativ:** Auf der operativen Ebene kann man zwischen allgemeiner Unterstützung des Testprozesses und seiner aktuellen Durchführung unterscheiden. Beispiele für die allgemeine Unterstützung des Testprozesses sind methodologische, technische und funktionale Empfehlungen für die Tester. Beispiele für die Testdurchführung sind der eigentliche Test und das Testmanagement, die beide eigenständig organisiert werden können. So können diese etwa in einer Matrixorganisation, auf Projektbasis oder teils in der Linie durchgeführt werden [TMAP2008].

Literatur

[TMAP2008] : Koomen van der Aalst Broekmann Vroon, TMap Next, dpunkt Verlag Heidelberg 2008

Zusammenfassung

Abnahmekriterien sind messbare Leistungen oder Eigenschaften des Systems, die im Rahmen eines Abnahmetests nachgewiesen werden und für die Freigabe des Testgegenstands relevant sind. Für die Festlegung von Abnahmekriterien sind ebenso wie für Vorbereitung und Durchführung von Abnahmetests einige Punkte zu beachten.

14.1 Merkmale von Abnahmekriterien

Mit der Abnahme bestätigt der Auftraggeber, dass seine Anforderungen erfüllt sind, die in Projektauftrag bzw. Lastenheft oder Kundenspezifikation beschrieben sind. Die Abnahme ist Voraussetzung für den Projektabschluss. Sie hat darüber hinaus rechtliche Auswirkungen, z. B. für Rechnungsstellung oder Gewährleistung. Grundlage der Abnahme ist der Abnahmetest, der überprüft, ob alle Abnahmekriterien erfüllt sind. Wichtig ist, dass die Abnahmekriterien frühzeitig definiert werden und allen Beteiligten bekannt sind.

Bei IT-Projekten verbindet das Testen somit die wertschöpfenden Projektprozesse (d. h. die Erstellung) und die Projektmanagementprozesse, z. B. des Qualitätsmanagements und der Abnahme. Zum einen müssen die Testprozesse deshalb in den Plan des jeweiligen Projektes integriert werden, zum anderen muss eine Testorganisation eingerichtet werden, die Test- und Managementprozesse aufeinander abstimmt [PRMG2019].

Abnahmekriterien sind nach IEEE 610 jene Kriterien, die ein System oder eine Komponente erfüllen muss, um eine Abnahme durch den Benutzer, Kunden oder eine bevollmächtigte Instanz erfolgreich abschließen zu können. Für den Test ist ein Abnahmekriterium eine Anweisung einer Anforderung betreffend, welche die Prüfung und Bewertung des erstellten Produktes oder durchgeführten Prozesses gegenüber dieser Anforderung beschreibt. Das bedeutet aber auch, dass in Abnahmekriterien keine zusätzlichen

Leistungen oder Eigenschaften versteckt sein dürfen, die in den eigentlichen Anforderungen nicht auftauchen.

Abnahmekriterien stellen demnach vom Auftraggeber meist vertraglich festgelegte Punkte dar, die festlegen, wann ein Produkt vom Auftragsnehmer an den Auftraggeber übergeben werden kann.

Werden diese Punkte nicht erfüllt, kann der Auftraggeber das Produkt ablehnen, Zahlungen teilweise verweigern und eventuell kostenfreie Nachbesserungen fordern. Der Auftragsnehmer sichert sich somit also vertraglich gegenüber dem Auftraggeber ab. Werden nur weniger wichtige Kriterien nicht erfüllt und das Produkt allgemein als akzeptabel anerkannt, kann auch eine teilweise Freigabe erfolgen und dem Auftragsnehmer eine Frist zur Nachbesserung, eventuell auch während des laufenden Betriebes, eingeräumt werden.

Dabei ist entscheidend, dass Abnahmekriterien messbar sind. Bei der Definition der Abnahmekriterien muss daher das Verfahren und die Messmethode festgelegt werden, da sie sonst interpretierbar sind.

Die Abnahmekriterien definieren dadurch, wann ein Test erfolgreich durchgeführt wurde.

Der Abnahmetest ist die letzte Testphase vor der Freigabe des Systems. Nach der Abnahme geht das System in Produktion, ob es nun direkt zur Verfügung gestellt wird (etwa über eine Webanwendung oder eine Großrechnerapplikation), oder mittels eines Rollout auf dezentrale Systeme und Geräte aufgebracht wird.

14.2 Abnahmekriterienkatalog

Anbei eine mögliche Vorlage für einen **Abnahmekriterienkatalog**:

Änderungshistorie,
Genehmigungen und
Verteiler.

Inhaltsverzeichnis,
Begriffe und Abkürzungen sowie
weitere Verzeichnisse.

1. Einführung,
 - 1.1 über das Dokument,
 - 1.2 Produktinformationen,
 - 1.3 Projektinformationen.

2. Produkt-Abnahmekriterien:

2.1 Funktionalität

2.2 Produktqualität und Dokumentationen

2.3 Benutzbarkeit

2.4 Ressourcen

2.4.1 Hardware

2.4.2 Software

2.4.3 Datenbanken

2.4.4 Systemkonfigurationen

2.5 Performance Level

2.6 Zuverlässigkeit und Verfügbarkeit und

2.7 Sicherheit.

3.1 Zeitplan und Meilensteine

3.2 Kosten

3.2.1 Entwicklung und

3.2.2 Betrieb.

Folgende Bereiche können Anhaltspunkte für Kriterien liefern:

- Finanzen;
- Management: Produkt, Änderungen, Prozesse, Strukturen;
- Verantwortliche;
- Software;
- Plattform für den Betrieb: Hardware, Datenbanken, Betriebssystem, Systemkonfigurationen;
- Sicherheit;
- Monitoring, Logging;
- Dokumentation: Benutzer, Test, Produkt, Technik, Service;
- Schulungen;
- Gewährleistung und
- Release Informationen [[BULL2008](#)].

Als Abnahmekriterien sollen definierte Testfälle ausgewählt und dem Auftraggeber zur Verfügung gestellt werden. Zur Erstellung des Testkonzeptes müssen diese Testfälle noch nicht im Detail definiert sein, es ist ausreichend, den Ablageort und das Dokument zu erwähnen, mit dem die Definition und Bestätigung dieser Testfälle erfolgt. Es muss aber geplant werden, zu welchem Zeitpunkt die Abnahmetestfälle beschrieben und dem Auftraggeber vorgelegt werden können. Die Abnahme findet in der Regel an einem Arbeitstag oder in wenigen Stunden statt. Der Umfang muss also begrenzt sein und soll sich auf ausgewählte, häufig vorkommende Geschäftsvorfälle beschränken. Meist ist es eine sinnvolle

Untermenge der Testfälle aus dem Systemtest. Das darf aber im Umkehrschluss keinesfalls bedeuten, dass nur die abnahmerelevanten Testfälle getestet werden und beim Test auf andere Testfälle verzichtet wird.

Die Abnahmekriterien werden pro Testobjekt individuell gewählt. Die Schärfe der Abnahmekriterien orientiert sich dabei an der Risikoanalyse; für unkritische Testfälle werden weichere Abnahmebedingungen aufgestellt als für sicherheitskritische Prozesse. Der Testaufwand konzentriert sich daher auf die wesentlichen Teile des Testobjektes.

14.3 Organisation der Abnahme

Außerdem ist zu prüfen, in welcher Umgebung die Abnahme durchgeführt werden soll. In der Regel ist das die Abnahmetestumgebung des Kunden. Die **Abnahmetestumgebung** soll so weit wie möglich der späteren Produktivumgebung entsprechen. Eine Testdurchführung in der Produktivumgebung selbst ist aber zu vermeiden, um den produktiven Betrieb laufender Softwaresysteme nicht negativ zu beeinflussen oder zu gefährden. Wegen der unterschiedlichen Testumgebungen kann ein Testfall in der Abnahmeumgebung fehlschlagen, der im Systemtest nie Probleme bereitet hat. Auch die Prozeduren zur Installation und Konfiguration des Systems sollen als Teil der Abnahme dabei überprüft werden [SPIL2007].

Dabei muss man im Vorfeld prüfen, wie hart die Abnahmekriterien sein sollen: Wenn ein Abnahmekriterium lautet, dass die Antwortzeit nicht über 1 Sekunde dauern darf, im Abnahmetest aber 1,1 Sekunden gemessen werden, ansonsten aber alle Funktionen erfolgreich getestet werden können und das System wie spezifiziert zur Verfügung steht, ist dann die Abnahme wirklich erfolglos? Ist das nicht eher eine gewisse Einschränkung, die in einem Folgerelease verbessert werden sollte, aber kein Hindernis, das System einzuführen?

Die Abnahmetestfälle werden vor der Abnahme ohne den Auftraggeber bzw. Kunden getestet, um die Qualität des Systems zu bewerten. Wenn man dabei zum Ergebnis kommt, dass noch erhebliche Fehler im System verborgen sind, empfiehlt es sich, den Abnahmetermin proaktiv zu verschieben und nicht den Kunden einzuladen und ihm mit einer desaströsen Abnahme seine Zeit zu stehlen. Das ist allerdings bei sehr engen Terminvorgaben manchmal kaum noch möglich. Umso wichtiger ist es, im Vorfeld bereits Pufferzeiten einzuplanen. Wenn man bei der Definition des Testkonzeptes bereits feststellt, dass der Abnahmezeitraum zu eng ist und wahrscheinlich kaum zu halten ist, sollte man bereits hier ein Projektrisiko anmelden.

Zur Definition der Abnahme gehört auch die organisatorische Klärung, wer die Abnahme erteilen darf und wer dazu einzuladen ist bzw. wem die Abnahmetestfälle vorgelegt werden müssen und welche Nachweise dazu benötigt werden. Manchmal reist zur Systemabnahme ein Mitarbeiter des Kunden an, der lediglich eine Empfehlung ausspricht, aber die Abnahme nicht selbst erteilt. Es sollte also auch von vornherein feststehen, wie der **Abnahmeprozess** im Detail aussieht, weil die Abnahme in der Regel einen Meilenstein definiert, zu dem ein bestimmter Prozentsatz zur Zahlung fällig ist. Teilweise werden ge-

plante Abnahmen daher auch vom Kunden aus Budgetgründen verzögert und hinausgeschoben.

14.4 Definition geeigneter Akzeptanzkriterien

Leider werden häufig immer noch Tester nicht frühzeitig genug in das Projekt eingebunden und perfekte, fehlerfreie Anforderungen sind schwer zu finden. Vor allem unter Zeitdruck und mit neuen, unerfahrenen Projektteams oder vage formulierten Lastenheften des Auftraggebers. Aber gerade in diesen Situationen sollte nicht auf Abnahmekriterien verzichtet werden – auch wenn es häufig schwerfällt. Oft sind perfekte, fehlerfreie Anforderungen im Anforderungsdokument schwer zu finden, vor allem, wenn unerfahrene Projektteams oder vage formulierte Lastenhefte des Auftraggebers vorhanden sind. In der agilen Welt sind die Abnahme- oder **Akzeptanzkriterien** häufig die einzige Detailierung der User Stories, aber auch in klassischen Projekten bieten Abnahmekriterien eine weitere Möglichkeit, Testfälle zu entwerfen und Schwerpunkte zu setzen. In beiden Fällen bieten sie Orientierung und Sicherheit.

Denn auch für das Requirements Engineering bieten die Abnahmekriterien eine indirekte Hilfestellung. So muss man bei der Definition der Requirements nochmals und in anderen Worten über die Anforderung nachdenken. So manche Doppeldeutigkeit ist durch die Formulierung der Abnahmekriterien schon ans Tageslicht gekommen, viele Anforderungen wurden so geschärft. Letztlich können vereinbarte Abnahmekriterien auch vor

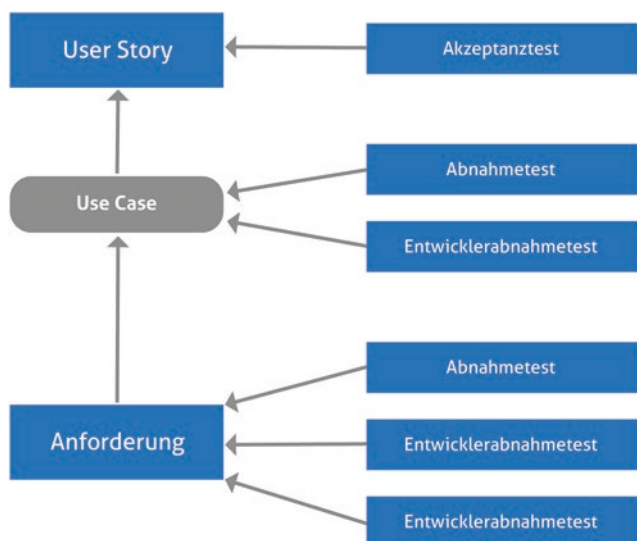


Abb. 14.1 Abnahmetests

juristischen Klagen schützen [ANEC2016]. Den Zusammenhang zwischen Anforderung, Use Case und User Story verdeutlicht folgende Abbildung (Abb. 14.1):

Im agilen Umfeld können Abnahmekriterien auf Anforderungsartefakte bezogen werden. Abnahmekriterien für Anforderungsartefakte legen fest, welche Bedingungen ein Anforderungsartefakt bzw. das Anforderungsdokument erfüllen muss, damit es abgenommen wird. Ein Anforderungsartefakt ist ein Ziel, ein Szenario oder eine Anforderung. Diesen Umstand kann man sich zunutze machen, wenn es darum geht, Abnahmekriterien gemeinsam mit dem Kunden zu entwickeln. Dabei werden zuerst die Abnahmekriterien für Use Cases entwickeln. Die Beschreibung der Abnahmekriterien kann sich an den entsprechenden Szenarien entlanghangeln, aus denen sich der Use Case zusammensetzt. Für Use Cases muss eine möglichst vollständige Abdeckung mit Abnahmekriterien erreicht werden. Danach kann man bei der Beschreibung in die Tiefe gehen und Abnahmekriterien für einzelne Anforderungen entwickeln.

Dabei sind Akzeptanzkriterien nicht von der Umsetzung des Produktes oder eines Testumfeldes abhängig. Jede weitere umgesetzte User Story erweitert oder verändert den Umfang der Akzeptanzkriterien des Gesamtproduktes. Neben den Akzeptanzkriterien der User Stories gelten auch „unausgesprochene“ Kriterien, die gar nicht explizit erwähnt werden und meist sehr offen formuliert werden, wie z. B. „Es dürfen keine unbehandelten Fehler/Speicherlecks/Abstürze vorkommen“ oder „lesbare und übersichtliche Benutzeroberflächen“.

Um die Qualität der aktuellen Umsetzung zu prüfen, werden für die Akzeptanzkriterien explizite Testfälle erzeugt. Diese Testfälle berücksichtigen das aktuelle Testumfeld (Technologien, Testdaten) in dem die aktuelle Umsetzung getestet wird. Je offener ein Akzeptanzkriterium formuliert ist, desto mehr Testfälle werden benötigt und desto schwieriger wird es, sinnvolle Testfälle zu formulieren. Um die Qualität zu überprüfen, muss die Menge dieser Akzeptanztests durchgeführt werden. Meist ist eine manuelle Ausführung dieser Tests bereits nach kurzer Projektlaufzeit nicht mehr regelmäßig möglich und eine Automatisierung dieser Testfälle daher unabdingbar [CODE2012].

Eine Möglichkeit, Akzeptanzkriterien für Projekte im agilen Umfeld zu definieren, erfolgt über das Hinterfragen von Schlüsselwörtern. Diese müssen zunächst im Anforderungstext bzw. der **User Story** identifiziert werden und nachfolgend ist zu prüfen, ob Fragestellungen – basierend auf den sogenannten W-Fragen (wer, wann, wie oft usw.) – mit einer sinnvollen Antwort versehen werden können. Fragen, die in dem jeweiligen Kontext keine sinnvolle Antwort zulassen, werden ignoriert. Unter Schlüsselwörtern sind Vollverben – auch als Prozessverben bekannt, Adjektive und Substantive zu verstehen. Deren Hinterfragen muss zu einer sinnvollen Antwort führen und einen Anforderungstext oder eine User Story inhaltlich ergänzen bzw. konkretisieren. Unter den Stakeholdern muss dabei ein gemeinsames Verständnis für die anzustrebende Umsetzung erlangt werden. Die Antworten auf die Fragestellungen ermöglichen, konkrete Akzeptanzkriterien zu definieren, die Qualitätskriterien entsprechen müssen. Zwischen den Antworten der Stakeholder und den Akzeptanzkriterien muss keine 1:1-Beziehung bestehen, sondern mehrere Antworten können zu einem Akzeptanzkriterium zusammengefasst werden oder eine Antwort

zu mehreren Akzeptanzkriterien führen. Die formulierten Akzeptanzkriterien müssen abschließend von den Stakeholdern abgenommen werden. Im letzten Schritt werden Testfälle erstellt, die dem bewährten Muster

- Vorbedingung,
- auszuführende Testschritte,
- erwartetes Ergebnis,
- Nachbedingung und
- verwendete Testdaten

entsprechen. Die formulierten W-Fragen, bei denen in den Platzhalter die Schlüsselwörter eingesetzt werden müssen, stellen einen Auszug dar, der für unterschiedliche Branchen und Domänen auf das jeweilige Fachgebiet angepasst oder ergänzt werden muss [[SIGS2011](#)].

Von User Stories abgeleitete Akzeptanzkriterien eignen sich also gut dazu, um eine ausreichende Testbasis für den Akzeptanztest sicherzustellen. Bei einer Vorgehensweise mithilfe von fünf Schritten steht die Definition von Akzeptanzkriterien im Mittelpunkt. Ziel soll es dabei sein, die Teamkommunikation zu unterstützen und konkrete Details zu User Stories zu definieren, um die Basis für einen erfolgreichen Akzeptanztest zu legen. Zudem kann am Beispiel der Definition von Akzeptanzkriterien durch das Hinterfragen von Schlüsselwörtern demonstriert werden, wie eine klassische Methode ins Scrum-Framework eingebettet werden kann [[HOOD2011](#)].

14.5 Eigenschaften von Akzeptanztests

Laut SCRUM soll ein **Akzeptanztest** über folgende Eigenschaften verfügen:

- **Abstrakt:** Der Akzeptanztest soll nicht jede Einzelheit umfassen, sondern sich auf das Wesentliche reduzieren.
- **Vertrauenswürdig:** Das Verhalten in der Test- und Abnahmeumgebung muss bezogen auf den zu testenden Sachverhalt immer mit dem Verhalten in der tatsächlichen Betriebsumgebung des Kunden identisch sein. Ebenso muss der Test für den Kunden nachvollziehbar reproduzierbar sein.
- **Kohärent** (zusammenhängend): Ein Akzeptanztest sollte nur einen (und nicht mehrere) fachlichen Zusammenhang abdecken.
- **Entkoppelt** (unabhängig): Ein Akzeptanztest muss unabhängig von allen anderen sein und darf nicht durch die Ergebnisse oder Verfügbarkeit anderer Tests im Ergebnis beeinflusst werden.
- **Aussagekräftig und verständlich:** die Motivation, der Inhalt des Testes und sein erwartetes Ergebnis sollten für Fachleute des Testgebietes (Domänenexperten) gut verständlich sein.

- **Redundanzfrei:** Ein zu testender Sachverhalt soll nur durch einen Test und nicht mehrfach abgedeckt werden.

Ein Akzeptanztest ist dabei als eine Beschreibung zu verstehen, wie die Akzeptanzkriterien überprüft werden sollen [OOS2010].

Es empfiehlt sich, dem Kunden bzw. für die Abnahme verantwortlichen die geplanten Testfälle vor der Abnahme zur Verfügung zu stellen und den Umfang zu planen, also für die Abnahme eine Agenda zu erstellen. Damit besteht von Anfang an ein gemeinsames Verständnis über Umfang und Ziel der Abnahme.

Sollte es bei einer Abnahme zu Problemen kommen, ist es unbedingt erforderlich, damit offen umzugehen. Wenn der Kunde den Eindruck bekommt, dass bestimmte Fakten verschwiegen werden, fühlt er sich hinters Licht geführt und wird sowohl das System besonders kritisch beachten als auch ein allgemeines Misstrauen gegenüber der Entwicklungs- und Testabteilung entwickeln. Eine Präsentation, was möglich war und was nicht erreicht wurde, ist dabei in jedem Fall besser als Herumdrucksen oder Falschinformationen. Oft kann man dann trotzdem eine Abnahme mit bestimmten Auflagen erreichen, die in einem vereinbarten Zeitraum nachgebessert werden. Sollte eine Abnahme nicht erteilt werden, so ist von beiden Seiten zu definieren und zu protokollieren, welche Punkte bis wann nachzuliefern sind, damit zumindest der nächste Versuch einer Abnahme nicht erneut scheitert. Dabei sollte auch der Umfang mit der folgenden Abnahme festgelegt werden, ob nur die reklamierten Punkte nach Behebung getestet werden oder das gesamte System neu überprüft wird.

Literatur

- [PRMG2019] : https://www.projektmagazin.de/artikel/testmanagement-it-projekten-teil-1_7180, zugegriffen am 09.05.2020
- [BULL2008] : <http://robert.bullinger.over-blog.com/article-abnahmekriterien-fuer-software-25457484.html>, zugegriffen am 09.05.2020
- [SPIL2007] Basiswissen Softwaretest, dpunkt Verlag Heidelberg, 3. Auflage 2005 korrigierter Nachdruck 2007
- [ANEC2016] <http://www.anecon.com/blog/abnahmekriterien-bruecke-oder-kruecke-zwischen-requirements-engineering-und-test/>, zugegriffen am 09.05.2020
- [CODE2012] <https://blog.codecentric.de/2012/04/akzeptanzkriterien-oder-tests-als-anforderungsdokumentation/>, zugegriffen am 09.05.2020
- [SIGS2011] https://www.sigs-datacom.de/uploads/tx_dmjournals/becker_OS_testing_11.pdf, zugegriffen am 09.05.2020
- [HOOD2011] <http://blog.hood-group.com/blog/2011/12/06/akzeptanzkriterien-fur-user-stories-definieren-aber-nur-wie/>, zugegriffen am 09.05.2020
- [OOS2010] <https://www.oose.de/blogpost/akzeptanzkriterium-vs-akzeptanztest-vs-testfall/>, zugegriffen am 09.05.2020

Kriterien für Testabbruch und Testfortsetzung

15

Zusammenfassung

In bestimmten Situationen müssen Tests abgebrochen oder unterbrochen und später fortgeführt werden. Dafür gibt es bestimmte Kriterien. Die Wiederaufnahme nach Testunterbrechung führt zu einer Neuplanung des Testprozesses.

15.1 Fälle für Testabbrüche

Es gibt Situationen, in denen es nicht sinnvoll ist, mit dem Test fortzufahren. Das kann zum Beispiel dann der Fall sein, wenn eine wesentliche Komponente Fehler enthält oder gar nicht vorhanden ist oder, wenn Konfigurationsprobleme die Verfügbarkeit der Testumgebung so einschränken, dass viele der geplanten Testfälle auf einen Fehler laufen würden. In diesem Fall würde ein Testaufwand entstehen, bei dem man schon von vornherein weiß, dass eine große Anzahl Fehler zu erfassen wären, die dann, wenn das auslösende Problem behoben sind, ohnehin wieder behoben sind. Professionelle Testtools erlauben daher, Testfälle für einen Testlauf auf „not applicable“ zu setzen. In diesem Fall spricht man von einem **Testabbruch**.

Ein weiterer Fall ist, dass ein Testobjekt einen derart geringen Qualitätsstand aufweist, dass auch nach mehreren Tests die Abnahmefähigkeit nicht erreicht wird. Dafür kann es sinnvoll sein, dieses Objekt komplett zu entfernen, um nicht die Einsatzfähigkeit des gesamten Produktes zu gefährden, sondern das Produkt mit einem kleineren Funktionsumfang auszuliefern. Auch in diesem Fall sind rechtzeitig weitere Testaktivitäten zu unterbinden.

Ein weiteres Abbruchkriterium ist, dass die Hardwarekapazitäten ungenügend sind. Das passiert zwar in den letzten Jahren wegen des enormen Fortschritts in der Hardwareentwicklung wesentlich seltener als früher, aber es kommt immer noch vor. Die Por-

tierung auf eine andere Hardware oder die Beschaffung neuer Hardware ist dann mit Zusatzaufwänden verbunden, und zunächst müssen Testaktivitäten unterbrochen werden.

Es kann auch der Fall sein, dass die zentrale Komponente, die Auswirkungen auf zahlreiche Testfälle hat, nicht fehlerhaft ist, aber in einer neuen Version entwickelt wird, und man während der Testphase erst entscheidet, die neuere Version für das fertige Produkt zu verwenden. Manchmal kommt es auch vor, dass die Systemarchitektur geändert werden muss, dass man bemerkt, dass die Anwendung zwar funktional korrekt ist, aber die Anwendung viel zu lange Antwortzeiten hätte, man also erst grundlegende Änderungen vornehmen muss. Daher macht es eventuell Sinn, einen Test zu unterbrechen und dann fortzuführen, wenn die Entwicklung der neuen Version abgeschlossen ist. Dieses Vorgehen ist pro Teststufe zu definieren, da der Komponententest im zeitlichen Ablauf vor Integrations- und Systemtest liegt. In diesem Fall muss die gesamte Projektplanung angepasst und das Testkonzept aktualisiert werden.

Eine andere Ursache kann sein, dass ein anderes Projekt eine höhere Priorität vor dem beschriebenen Projekt bekommt. Wenn mehrere oder alle Tester aus dem Projekt abgezogen werden, um das andere Projekt zu retten, muss der Test unterbrochen und später fortgeführt werden. Das hat selbstverständlich Auswirkungen auf die Terminsituation. Manchmal muss das Management dabei abwägen, ob eine Terminverschiebung für ein Projekt A machbar ist, um ein anderes Projekt B aus strategischen Gründen vorzuziehen oder eine drohende Penaltzzahlung wegen Terminverzug in Projekt B abzuwehren. In so einem Fall ist es wichtig, dass der Testmanager von Projekt A die Risiken einer Unterbrechung transparent darstellt.

Manchmal ist schlicht und ergreifend die vorgesehene Zeit für den Test abgelaufen. Entweder das Produkt kam zu spät in den Test oder das Testintervall war zu kurz geplant oder es gab Ressourcenengpässe im Testteam oder eine Kombination dieser Ursachen. Dann muss man das Produkt – schon aus vertraglichen Gründen – ausliefern und auf eine zunächst geplante Testabdeckung verzichten. Das bedeutet in der Konsequenz aber ggf. ein erhebliches Einführungsrisiko. Daher ist es besonders wichtig, Testfälle zu priorisieren, damit vor Auslieferung des Produktes immerhin noch die wichtigsten Testfälle überhaupt durchgeführt werden können.

15.2 Wiederanlauf und Testfortsetzung nach Testabbruch

Zu beachten ist, dass jeder Abbruch und jeder Wiederanlauf mit zusätzlichen Kosten und Risiken verbunden ist. Vielleicht sind die Testressourcen zu einem späteren Zeitpunkt nicht mehr verfügbar, man muss eine neue Einarbeitungsphase berücksichtigen und hat schon aufgrund der Unterbrechung gewisse Dokumentations- und Abstimmungsaufwände zu berücksichtigen. Außerdem hat eine Unterbrechung oder ein Abbruch auf die Projektmitarbeiter fatale Auswirkungen, was ihre Motivation anbelangt. Im Allgemeinen wird ein Projekt immer erheblich teurer, wenn es länger dauert. Manchmal bekommt man als Test-

manager Zweifel, ob diese Tatsache dem Management und dem Controlling wirklich in seiner gesamten Tragweite bewusst ist.

Dieses Vorgehen muss im Testkonzept beschrieben werden. Dabei sollte darauf eingegangen werden, ob (projektespezifisch) Risiken bekannt sind, wegen der eine Unterbrechung der Testphase von vornherein wahrscheinlich ist. Auch die kritischen Kernprozesse und wesentlichen Komponenten, die für die Funktionsfähigkeit der gesamten Anwendung entscheidend sind, sollten dabei dokumentiert sein.

Es gibt Projekte, bei denen aus gesetzlichen Gründen (z. B. EU-weite Inkrafttreten einer Verordnung) oder äußeren Gegebenheiten (z. B. Umzug eines Rechenzentrums, das zu einem bestimmten Zeitraum gekündigt wird), der Termin fix ist und ein Abbruch oder eine Verschiebung besonders fatal ist. Wenn das der Fall ist, sollte dieser Aspekt im Testkonzept besondere Erwähnung finden.

Zusammenfassung

Viele Projekte weisen keinen erfolgreichen Verlauf auf. Projektrisiken müssen von Anfang an erhoben und bewertet werden. Es gibt zahlreiche mögliche Risiken für den Projekterfolg, die zu erheblichen Verzögerungen und Zusatzkosten führen können.

16.1 Projekterfolg

Zwar hat sich der Anteil erfolgreich durchgeführter Projekte wegen verbesserter Methoden im Projektmanagement deutlich erhöht, sodass die überwiegende Anzahl als erfolgreich bewertet wird, es besteht aber nach wie vor ein großes Verbesserungspotenzial. Nach wie vor sind viele Projekte nicht oder nur in Teilaspekten erfolgreich. Aus einer Studie zur Vermessung der Projektstätigkeit in Deutschland, in deren Rahmen 500 privatwirtschaftliche und öffentliche Unternehmen befragt wurden, liegen folgende Zahlen vor (Abb. 16.1):

Interessanterweise werden Ergebnis und Qualität besser eingeschätzt als andere Kriterien. Das kann bedeuten, dass die Qualitätssicherungsmaßnahmen auf einem allgemein hohen Niveau sind (das wäre die optimistische Variante) oder aber die Qualität nicht genau genug betrachtet wird (das ist nach meinen Augen wahrscheinlicher, wenn der Test halbwegs erfolgreich war und nicht allzu viele offene Fehler auftraten, gibt man sich schon zufrieden). Probleme mit der Kosten- und Zeitsituation sind bekannt und die Werte erscheinen plausibel, dass die Stakeholder Zufriedenheit geringer ist, muss nicht nur in den Projekten an sich begründet sein, sondern an allgemeiner Unzufriedenheit mit der Arbeit, dem Unternehmen, der Bezahlung, personellen Differenzen und Kommunikationsproblemen.

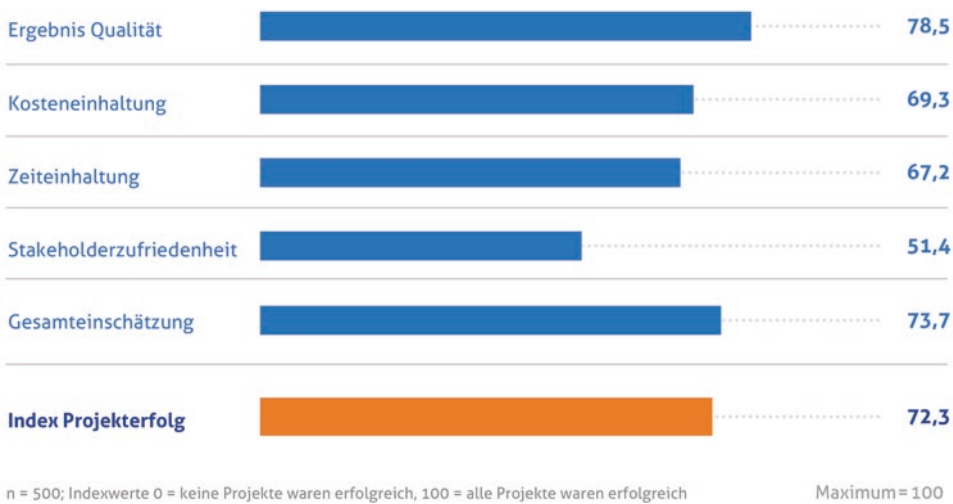


Abb. 16.1 Ausprägungen des Projekterfolgs [GPIM2015]

Tab. 16.1 Kalkulation von Projektrisiken

Risiko	Eintrittswahrscheinlichkeit	Kosten des Risikos	Kalkuliertes Risiko
Risiko 1	10 %	50.000 EUR	5000 EUR
Risiko 2	15 %	100.000 EUR	15.000 EUR
Risiko 3	5 %	300.000 EUR	15.000 EUR
Risiko 4	5 %	40.000 EUR	2000 EUR

16.2 Bewertung von Risiken

Viele Risiken sind bereits zu Projektbeginn bzw. in der Testplanungsphase bekannt. Daher müssen die **Testrisiken** und ihre möglichen Auswirkungen im Testkonzept genau beschrieben werden.

Dabei sollten die einzelnen Risiken gewichtet und bewertet werden. Was bedeutet es für den Fall, dass das Risiko eintritt, und wie hoch sind dabei die Kosten (Tab. 16.1)? Dazu müssen die Kosten erhoben und bekannt sein, was teilweise nicht trivial ist. Die Eintrittswahrscheinlichkeiten kann man aus der Erfahrung heraus schätzen, wobei man damit natürlich auch stark daneben liegen kann.

Die Summe der kalkulierten Risiken wird addiert. Im obigen Beispiel beträgt das Risiko also $5000 + 15.000 + 15.000 + 2000 = 37.000$ EUR

Man erkennt in diesem Beispiel, dass es vor allem sinnvoll ist, Risiko 2 und Risiko 3 zu vermeiden, weil diese beiden Risiken das Projekt viel stärker treffen würden als Risiko 1 und Risiko 4. Man kann also von Anfang an seine Ressourcen besser und zielgerichteter steuern, wenn man sich dieser Risiken bewusst macht und sie bewertet.

In den regelmäßigen Projektreviews soll darauf eingegangen werden, ob die Risikobewertung noch aktuell ist, ob ggf. neue Risiken erkannt werden oder ob eines der Risiken eingetreten ist.

16.3 Mögliche Testrisiken

Wichtig ist es also, sich einen Überblick über mögliche Testrisiken zu verschaffen. Beiliegende Liste soll mögliche Risiken und deren Auswirkung beleuchten. Die Liste ist keineswegs vollständig, aber sie zeigt in der Praxis häufig auftretende Probleme und kann als Gedankenstütze für die Erstellung eines Testkonzeptes herangezogen werden.

Management

1. Der Projektleiter hat nicht die Macht, Projektziele durchzusetzen und wird vom Management nicht ausreichend unterstützt.
2. Der Test wird im Unternehmen als nachrangig betrachtet. Die Führungsebene interessiert sich für das Projekt nicht und vernachlässigt die Kommunikation.
3. Es gibt Unstimmigkeiten über den Projektumfang auf der Managementebene. Einige Führungskräfte wollen das Projekt politisch nicht.
4. Ein Wechsel im Management führt zu einer erneuten Priorisierung von Aufgaben.

Projektumfang

1. Die Anforderungen sind unzureichend, fehlerhaft oder widersprüchlich. Der Projektumfang ist dadurch nicht ausreichend beschrieben.
2. Unkontrollierte Änderungen lassen den Projektumfang steigen.
3. Das Projektteam fügt Produkt-Features hinzu, die nicht in den Anforderungen und auch nicht in Change Requests vorhanden sind.
4. Die Schätzungen sind von Anfang an ungenau oder zu optimistisch. Testaktivitäten wurden nicht ausreichend bewertet.
5. Abhängigkeiten von äußeren Umständen oder anderen Projekten wirken sich auf die Termin- und Kostensituation des eigenen Projektes ungünstig aus.
6. Einzelne Aktivitäten im Projekt wurden in der Planungsphase nicht ausreichend berücksichtigt.

Kostenmanagement

1. Die Kosten für Software, Hardware oder Personal sind nicht vollständig erhoben.
2. Es kommt zu nachträglichen Änderungen der Projektkosten.
3. Verzögerungen oder Leerlaufzeiten aufgrund zu später Realisierung erhöhen die Projektkosten.
4. Umfangreiche Fehleranalysen und wiederholte Regressionstests führen zu steigenden Kosten.

Änderungsmanagement

1. Eine zu große Anzahl von Änderungen oder eine zu hohe Dynamik in den Anforderungen führt zu Komplikationen, Verzögerungen und Verwirrung im Projekt.
2. Es besteht keine Einigkeit über Änderungen oder deren Auswirkungen. Das führt zu Konflikten im Projekt.
3. Es gibt kein geeignetes System, um Änderungen nachzuvollziehen.
4. Der Prozess im Change-Management fehlt oder ist undurchsichtig. Entscheidungswege sind intransparent oder werden von den falschen Stakeholdern getroffen.
5. Wechselnde oder ungenaue Prioritäten bei Änderungen beeinflussen den Terminplan.
6. Die Formulierung der Änderungen ist nicht exakt genug, berücksichtigt nicht alle Aspekte oder steht im Widerspruch zum bestehenden System, der Systemarchitektur oder anderen Anforderungen.

Stakeholder

1. Probleme in der Kommunikation durch fehlende Beteiligung der Stakeholder.
2. Die Stakeholder haben Erwartungen an das Projekt, die nicht definiert sind.
3. Fluktuation und wechselnde Stakeholder verlangsamen Projektaktivitäten.
4. Gruppendynamische Prozesse oder Konflikte zwischen Stakeholdern, schwelende Konflikte, Unstimmigkeiten, Vorurteile oder persönliche Abneigungen. Aus meiner Sicht ist das einer der schwierigsten Probleme. Man schätzt, dass Probleme in Projekten zu 20 % technische oder organisatorische und 80 % zwischenmenschliche Ursachen haben. Folgerichtig kann nach Schätzungen in etwa 80 % der Fälle, in denen ein Projekt in Schieflage gerät, die Ursache auf interne Machtkämpfe oder aber persönliche Konflikte zurückgeführt werden. Deswegen ist ein gut funktionierendes Konfliktmanagement für Erfolge im Projektgeschäft essenziell [THEQ2020].
5. Negative Einstellung von Stakeholdern, die Stakeholder wünschen sich ein Scheitern des Projektes.
6. Mangelhafte Zuarbeit von Stakeholdern: dazu gehören fehlende Erfahrung und mangelnde Informationen genauso wie wenig Sorgfalt, wenig Motivation oder häufige Abwesenheiten.

Kommunikation

1. Fehlinterpretation von Anforderungen oder ungenau formulierte Anforderungen führen zu fehlerhaften Testfällen. Workshops von Testern mit den Auftraggebern und dem Anforderungsmanagement werden leider in vielen Fällen unterlassen oder viel zu spät durchgeführt.
2. Fehlende Kommunikation führt zu Missverständnissen und Unzufriedenheit. Aus eigener Erfahrung: es ist oft schwierig herauszufinden, wer welche Information benötigt. Entweder man gibt Informationen an zu viele, die damit nichts anfangen können oder der einzelne Mitarbeiter erkennt nicht, dass es Handlungsbedarf für ihn gibt. Wenn betroffene Personen nicht informiert werden und sich übergangen fühlen, kann das ebenfalls zu Störungen und Widerständen im Projektverlauf führen. Zu viel Kommuni-

kation oder Informationen an die falschen, nicht zuständigen Stakeholder kann das Projekt ebenfalls verzögern. Den „goldenen Mittelweg“ am Umfang und Ziel der Kommunikation zu finden, ist eine enorme Herausforderung.

3. Gegenseitige Schuldzuweisungen, politisches Agieren, Allianzen und Aversionen innerhalb des Projektteams. Es zeigt sich immer wieder, dass persönliche Konflikte innerhalb von Projekten zu den größten Risiken zählen, die einen Projekterfolg verhindern.

Ressourcen

1. Zur Erledigung der einzelnen Aufgaben fehlen Testressourcen.
2. Die Einarbeitung neuer Mitarbeiter gestaltet sich schwieriger als angenommen: Die Lernkurven von Mitarbeitern, die neu im Beruf, im Unternehmen oder im Projekt sind führen zu einer geringeren Produktivität. Mit der Einarbeitung sind häufig diejenigen Mitarbeiter betraut, die ohnehin schon überlastet sind. Fehlendes oder unzureichendes Training und mangelnde Erfahrung führen zu Verzögerungen im Projektverlauf.
3. Mangelnde Leistung, negative Einstellung, geringe Motivation oder fehlende Unterstützung durch Liniovorgesetzte können zu Termin- oder Kostenüberschreitungen führen.
4. Fluktuation während des Projektes vor allem erfahrener Mitarbeiter bei mangelhafter Dokumentation der durchgeführten Arbeiten. Viel Know-how ist nur in den Köpfen, aber nicht an der richtigen Stelle nachvollziehbar dokumentiert. Die Übergabe- und Einarbeitungsphasen sind meist zu kurz, um in einem Projekt noch ausreichend Wirkung zu entfalten.
5. Personelle Schlüsselressourcen sind zu einem anderen Projekt abgezogen und stehen daher nicht zur Verfügung.

Systemarchitektur

1. Die Systemarchitektur passt nicht zu den vorgeschriebenen Grundsätzen.
2. Die Systemarchitektur ist nicht vollständig und aktuell beschrieben. Dadurch kann es vorkommen, dass Auswirkungen auf angebundene Systeme und Umverfahren vergessen und nicht getestet werden.
3. Die Architektur ist unflexibel oder mangelhaft.
4. Die Architektur ist schwer implementierbar, kostenintensiv oder erfüllt die Anforderungen nicht.

Design

1. Das Design ist unflexibel oder mangelhaft.
2. Das Produktdesign ist nicht ohne weiteres umsetzbar oder passt nicht zu den Anforderungen.
3. Die Qualität des Produktdesigns ist nicht ausreichend.
4. Stakeholder oder Experten lehnen das Design ab.

Technik und Qualität

1. Einzelne technische Komponenten erfüllen die Anforderungen nicht ausreichend.
2. Technische Komponenten skalieren nicht und erfüllen Performance-Anforderungen nicht.
3. Technologische Komponenten arbeiten nicht zusammen und haben Schnittstellenprobleme.
4. Teile des Systems weisen Sicherheitslücken auf. Durch diese Sicherheitslücken können Informationen nach außen dringen.
5. Mangelnde Stabilität oder Systemabstürze einzelner Komponenten führen zu Problemen beim Test und verzögern deren produktiven Einsatz.
6. Einzelne Komponenten arbeiten nicht zuverlässig oder fallen nach kurzer Zeit aus.
7. Standards und Best Practices werden nicht eingehalten.
8. Komponenten sind „over-engineered“, also mit unnützen Funktionen oder Design-Features aufgebläht. Das führt zu erhöhter Komplexität und höherem Fehlerrisiko beim Test.
9. Technische Komponenten sind nicht ohne weiteres mit neuen Funktionen erweiterbar.
10. Ausfall kritischer Systeme wie z. B. Testumgebungen oder mangelnde Verfügbarkeit von Testumgebungen.
11. Die Erhebung und Administration von Testdaten gestalten sich weitaus schwieriger als zunächst angenommen.
12. Fehlende Dokumentation bestehender Komponenten führen zu Problemen bei der Systemintegration.
13. Fehlende Updates, fehlender Hardware-Support und zu geringe Wartungsmöglichkeiten nicht mehr unterstützter Komponenten gefährden die Systemstabilität.
14. Komponenten oder Produkte können nicht in Betrieb genommen werden, weil wichtige Kriterien nicht erfüllt werden.
15. Die Qualität der Software ist schlechter als geplant, es gibt dadurch Verzögerungen in der Fehlerbehebungsphase, und es fallen mehr Regressionstests als vorgesehen an. Zusätzliche Lieferungen führen zu Zusatzaufwand und Überstunden vor allem in der Endphase des Projektes.
16. Die Tools (z. B. zur Fehlererfassung oder Testfalldefinition) sind nicht geeignet oder es werden unterschiedliche Tools eingesetzt. Ein Wechsel von Tools während der Projektlaufzeit oder unterschiedliche Tools können das Reporting für den Testmanager erheblich erschweren und binden zusätzlichen ungeplanten Aufwand der Stakeholder.

Integration

1. Der Aufbau von Hardware und Software oder die Bereitstellung der Testumgebung verzögert sich.
2. Das Produkt lässt sich nicht ohne weiteres in betriebliche Prozesse oder existierende Systeme integrieren.

3. Das Projektergebnis kann nicht in die Unternehmensorganisation integriert werden, weil zwar das Projekt umgesetzt, jedoch nicht das Gesamtbild betrachtet wurde. Beispiel: Es wurde ein Vertriebssystem aufgebaut, aber es gibt kein Vertriebsteam.
4. Schlüsselkomponenten sind fehlerhaft oder stehen zum gewünschten Termin nicht zur Verfügung.
5. Das Zusammenspiel einzelner Komponenten funktioniert nicht.
6. Das Projekt stört operative Unternehmensprozesse und beeinflusst das finanzielle Ergebnis negativ.
7. Das Projekt stört Verkaufsprozesse oder Complianceprozesse wie Audits und das Berichtswesen.

Anforderungen

1. Anforderungen passen nicht zur Unternehmensstrategie oder zu Unternehmensprozessen.
2. Anforderungen passen nicht zu bereits existierenden Systemen.
3. Anforderungen sind mehrdeutig, widersprüchlich oder unterschiedlich interpretierbar.
4. Anforderungen sind unvollständig oder von nicht ausreichender Qualität.
5. Anforderungen stehen im Widerspruch zu bestehenden Regularien oder gesetzlichen Vorgaben.
6. Unklarheiten und mangelhafte Qualität der Anforderungen führen zu Rückfragen und unvollständigen oder falschen Testfällen.

Entscheidungen

1. Offene Entscheidungen verzögern den Projektverlauf, wenn keine klaren Termine und Regeln zur Entscheidungsfindung definiert werden.
2. Entscheidungen sind mehrdeutig aus Angst vor der Übernahme von Verantwortung.
3. Entscheidungen werden unsachlich oder aufgrund falscher oder unvollständiger Informationen getroffen.
4. Unvollständig getroffene Entscheidungen lösen Probleme nur teilweise und ziehen weitere offene Fragen nach sich,

Beschaffung

1. Inakzeptable Vertragsbestandteile sind bei Zulieferungen von Lieferanten enthalten, die nicht verhandelt werden können.
2. Zulieferungen weisen eine zu geringe Qualität auf und können im Projekt nicht verwendet werden.
3. Konflikte zwischen Lieferanten und Auftraggeber oder Konflikte und Abhängigkeiten unter Lieferanten führen zu Terminverzögerungen.
4. Lieferanten starten eine Aufgabe zu spät oder liefern sie verspätet ab.

5. Service und Beratung vom Lieferanten sind nicht in ausreichender Qualität.
6. Lieferanten fallen z. B. aufgrund von Insolvenz aus und sind dadurch nicht mehr lieferfähig. Alternative Beschaffung ist schwierig und verzögert das Gesamtprojekt.
7. Komponenten oder Zulieferungen verstoßen gegen Patente.

Macht und Befugnis

1. Dem Projektteam fehlt die Autorität, bestimmte Aufgaben um- oder durchzusetzen.
2. Befugnisse zur Durchsetzung von Projektzielen sind unklar.
3. Widersprüchliche Aussagen und Anweisungen führen zu Verwirrung.

Genehmigungsverfahren und Bürokratie

1. Verzögerte Zustimmungen von Stakeholdern führen zu Problemen im Terminplan.
2. Nicht rechtzeitige Bereitstellung von Geldmitteln wirken sich negativ auf das Projekt aus.
3. Verzögerungen bei Einkaufsprozessen oder der Rekrutierung von Mitarbeitern verzögern das Projekt.
4. Weiterbildungen oder Trainings finden nicht rechtzeitig statt. Einzelne Stakeholder werden in andere Projekte abberufen.

Organisation

1. Das Projekt passt nicht zur Unternehmenskultur.
2. Eine Umstrukturierung im Unternehmen verzögert das Projekt oder beeinflusst die Ziele.
3. Eine Fusion oder ein Unternehmenszukauf oder -verkauf stören den Projektverlauf.
4. Andere Projekte bekommen eine höhere Priorität, sodass Ressourcen aus dem Projekt abgezogen werden müssen.
5. Zuständigkeiten für bestimmte Funktionen oder Fehleranalysen sind nicht klar geregelt.

Externe Risiken

1. Rechtliche Änderungen beeinflussen das Projekt.
2. Änderungen auf dem Markt (z. B. Börsencrashes) oder höhere Gewalt wirken sich auf das Projekt aus. Dabei werden Entscheidungen verschoben oder Finanzmittel verspätet bereitgestellt.
3. Technische Veränderungen in der Branche oder Businessinnovationen verzögern den Projektverlauf.

Projektmanagement

1. Prozesse im Projektmanagement werden nicht in ausreichendem Maße eingehalten.
2. Fehlendes oder unzureichendes Projektmanagement führt zur fehlenden Einhaltung von Best Practices.
3. Fehler in Methoden des Projektmanagements (z. B. fehlerhafte Terminplanung oder Algorithmen) verzögern das Projekt.

Nachrangige Risiken

Nachrangige Risiken entstehen dann, wenn ein Risiko nach außen, z. B. zu einem Lieferanten, verlagert wird.

Kunden- oder Nutzerakzeptanz

1. Die Benutzeroberfläche ist von geringer Qualität, langsam oder schwer zu bedienen.
2. Die Benutzeroberfläche ist nicht barrierefrei und verstößt gegen Regeln.
3. Das Projekt wird von Nutzern als Störfaktor in der eigenen Produktivität empfunden.
4. Nutzer lehnen den Prototypen ab, was Nacharbeiten oder komplett neue Designs erfordert.

Im Allgemeinen unterscheidet man hierbei Projekt- und Produktrisiken. **Projektrisiken** bedrohen die Fähigkeit des Projektes, ein Produkt abzuliefern. **Produktrisiken** hingegen resultieren auf Problemen mit dem gelieferten Produkt.

Risikoorientiertes Testen berücksichtigt Informationen über Produkt- und Projektrisiken und legt den Schwerpunkt der Testaktivitäten auf Bereiche hohen Risikos [PRLE2019].

Literatur

- [PRLE2019] : <https://projekte-leicht-gemacht.de/blog/pm-in-der-praxis/130-projektrisiken-beispiele/>,
zugegriffen am 09.05.2020
- [GPIM2015] : https://www.gpm-ipma.de/fileadmin/user_upload/GPM/Know-How/GPM_Studie_Vermessung_der_Projektaetigkeit.pdf, zugegriffen am 09.05.2020
- [THEQ2020] : <https://thequaliteers.de/nachrichten-leser/items/10-schritte-start-testprojekt-02.html>,
zugegriffen am 09.05.2020

Zusammenfassung

Das Testdatenmanagement regelt Einsatz und Verwaltung von Testdaten. Die Verfügbarkeit von Testdaten ist dabei sicherzustellen und besondere Fragestellungen sind zu beantworten. Dazu sind unterschiedliche Ansätze und Werkzeuge anzuwenden.

17.1 Testdatenmanagement

Beim **Testdatenmanagement** sind mehrere Aspekte zu bedenken:

- Personen, Rollen und Risiken,
- Tools, Methoden und Vorgehen,
- definierte Prozesse,
- Aufgaben und nötige Skills,
- technische Herausforderungen,
- Sicherheit, Gesetze, Datenschutz und Regularien sowie
- Effizienz und Wirtschaftlichkeit.

Zu bedenken ist, wie Testdaten bei Bedarf automatisiert zur Verfügung gestellt werden, wie die Testdaten nach der Verarbeitung wieder auf den Originalzustand zurückgesetzt werden und wie die Validität und die Aktualität des Testdatenbestands nachgehalten werden kann.

Testdaten müssen schnell und effizient bereitgestellt werden, gerade wenn es sich um Massendaten handelt. Verknüpfungen und Abhängigkeiten über unterschiedliche Systeme hinweg sind zu bedenken.

Beim Testdatenmanagement ist zu klären, welche Art von Testdaten verwendet werden sollten: rein **synthetische Daten** oder ein **Produktionsdaten-Abzug**, der mit einer **Anonymisierung** (Datenverfremdung) bearbeitet wurde, oder eine Kombination aus beiden Verfahren. Das kann in einem Projekt auch je nach Teststufe unterschiedlich sein [TEST2019].

Typische Herausforderungen beim Testdatenmanagement sind der Einsatz und die Verwaltung der Daten:

- Testfälle sind nicht ausführbar, weil Datenkonstellationen fehlen oder nicht verfügbar sind.
Beispiel: Eine Bank möchte neue personalisierte Angebote testen, aber im Depot fehlen Kundendaten.
- Die Testumgebung ist zu klein für die benötigten Testdaten.
Beispiel: Mehrere Testteams benötigen Daten für repräsentative Regressionstests, die sich während der Tests verbrauchen. Sie müssen daher mehrfach bereitgestellt werden.
- Die verfügbaren Daten dürfen aus regulatorischen Gründen nicht benutzt werden.
Beispiel: Die Kundendaten dürfen aus datenschutzrechtlichen Gründen nicht für den Test der personalisierten Angebote verwendet werden.
- Der Aufbau eines Testdatenbestandes ist aufwändig und langwierig.
Beispiel: In einem Testdatenbestand mit mehreren tausend Attributen sind diejenigen zu identifizieren, die aus Gründen der Compliance relevant sind. Dabei muss entschieden werden, ob die Daten synthetisch oder anonymisiert erzeugt werden.
- Für die Testauswertung werden Vergleichsdaten benötigt.
Beispiel: Die im Test erzeugten Buchungssätze müssen gegen eine Referenzdatei geprüft werden.

In vielen Fällen dauert es zu lange, bis die Daten beim Tester ankommen. Nur die wenigsten Mitarbeiter der Entwicklungsabteilungen haben selbstverantwortlichen Zugriff auf die Daten. Sie müssen mit dem arbeiten, was ihnen bereitgestellt wird.

Um agiles Testdatenmanagement in der Softwareentwicklung zu betreiben, müssen die dafür notwendigen Daten möglichst flexibel bereitgestellt werden.

Folgende Schritte sind dabei erforderlich:

- **Datenvirtualisierung:** Zur Virtualisierung von Testdaten werden hochkomprimierte, virtualisierte Datenkopien aus beliebigen Datenquellen erstellt. Die Kopien ermöglichen den Benutzern einen schnellen Zugriff auf die benötigten Informationen. Das optimiert die Ressourcen, die eine Datenbank verbraucht, steigert die Speichereffizienz und senkt damit Kosten. Die Datenvirtualisierung gelingt durch eine Verbindung zu einer externen Lösung, die unterbrechungsfrei, minimalinvasiv, kontinuierlich und effizient Daten virtualisiert.
- **Datenmaskierung:** Im zweiten Schritt werden die Daten maskiert. Die Informationen in den Ursprungsdaten werden dabei allesamt anonymisiert. Das hat den Vorteil, dass

Kunden- und Firmendaten konsequent geschützt werden und dem Datenschutz Rechnung getragen wird, ohne dass der Tester gänzlich auf sensible Daten verzichten muss.

- **Testdatenbereitstellung:** Die Tester und Entwickler können eigenständig auf die virtualisierten und maskierten Daten zugreifen. Dabei ist darauf zu achten, zeitaufwändige Autorisierungsverfahren zu vermeiden, um Agilität und Effizienz sicherzustellen. Tester und Entwickler müssen sich auf die für sie relevanten Daten konzentrieren können, ohne unnötig Zeit in Freigabe- und Warteschleifen zu verlieren. Dabei muss auch ein einfaches Zurücksetzen der Testdaten in einen beliebigen früheren Zustand gewährleistet sein, damit der Tester sorgenfrei mit den Testdaten experimentieren kann.
- **Automatisierte Datenbereitstellung:** Eine automatisierte Datenbereitstellung ermöglicht Testern und Entwicklern, effizienter und genauer zu arbeiten sowie Zeit zu sparen. Diese können sie für produktive und nichtrepetitive Tätigkeiten nutzen, die ihr ganzes Know-how und ihre ungeteilte Aufmerksamkeit erfordern [ITDA2019].

17.2 Ansätze zur Testdatenerstellung

Es gibt unterschiedliche Ansätze, um geeignete Testdaten zu erstellen. Die verschiedenen Arten zeigt folgende Abbildung (Abb. 17.1):

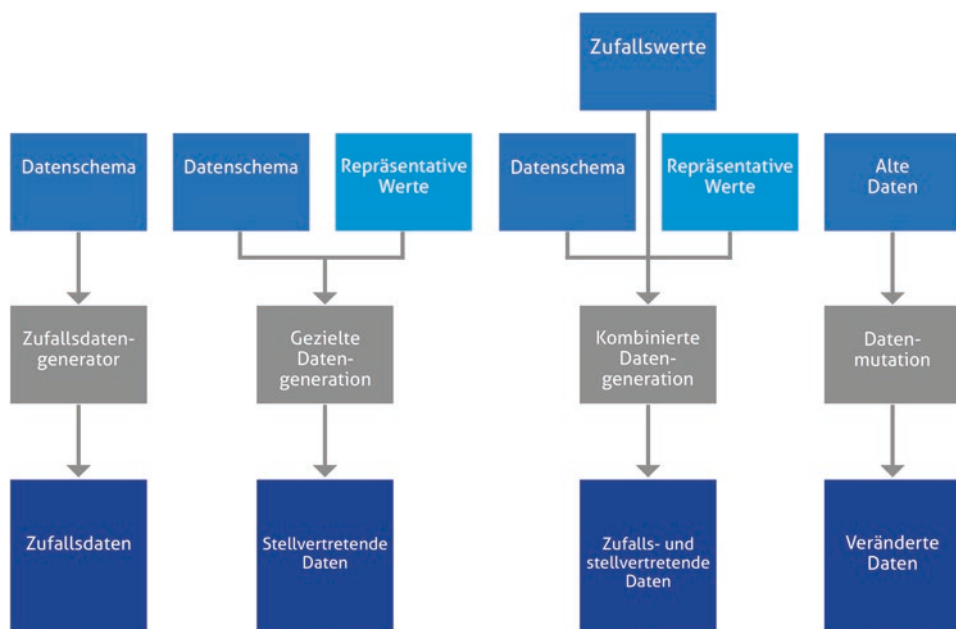


Abb. 17.1 Ansätze zur Testdatenerstellung

- **Blinder Ansatz:** Beim blinden Ansatz werden Zufallsdaten erzeugt. Der Erzeuger der Daten betrachtet dabei weder die Spezifikation noch den Code. Nur die Datenbeschreibung (Typ, Bezeichnung und Größe der Datenfelder) ist dem Datengenerator bekannt. Da der blinde Testdatengenerator die Algorithmen in der Software nicht kennt, versucht er, möglichst viele Wertekombinationen aller stellvertretenden Werte zu erzeugen. Dieser Ansatz ist sehr aufwändig und die Chancen, eine bestimmte Funktion zu testen und darin Fehler zu finden, sehr gering. Die Testdatenerzeugung durch einen Zufallsgenerator ist zwar sehr einfach möglich, die Testdurchführung aber umso aufwändiger und die Testabdeckung tendenziell gering. Daher ist dieser Ansatz für die Praxis wenig geeignet.
- **Gezielter Ansatz:** Beim gezielten Ansatz werden genau jene Daten erzeugt, die zu einem bestimmten Ausgang führende Funktionen auslösen, man leitet also die Testdaten aus der Funktionsspezifikation ab. Jeder Testfall zielt auf eine oder mehrere Funktionen mit dem Zweck, eine ihrer Ausgänge zu durchlaufen. Der Ansatz entspricht der White-Box-Testmethode, nur ist nicht der Code, sondern die Spezifikation der Analysegegenstand. Vorteilhaft beim gezielten Ansatz ist die höhere Testabdeckung mit weniger Daten, von Nachteil der höhere Aufwand zur Analyse der Funktionen und zur Spezifikation der Testfälle.
- **Kombinierter Ansatz:** Der kombinierte Ansatz vereint den blinden und den gezielten Ansatz. Zuerst werden einige Daten für jedes Datenbankattribut sowie für jede Eingabevariable wahllos generiert. Anschließend werden anhand der Funktionsspezifikation die wahllos generierten Daten manipuliert, um bestimmte Funktionen gezielt auszulösen. Der Vorteil des kombinierten Ansatzes besteht darin, dass einige kritische Funktionen gezielt, andere aber nur zufällig getestet werden. Man spart also einiges an Analyseaufwand gegenüber dem gezielten Ansatz und einiges an Testausführung gegenüber dem blinden Ansatz und erhöht die Testabdeckung gegenüber einem reinen Zufallstest.
- **Mutationsansatz:** Beim Mutationsansatz wird ein bestimmter Datenbestand manipuliert. Der Test beginnt mit nur wenigen Testdaten, die eine Art Smoketest verursachen. Danach wird der Testdatenbestand schrittweise ausgebaut, zum Teil durch Zufallsdaten, zum Teil durch gezielte Werte für den Test bestimmter Funktionen. Dadurch kann mit diesem Ansatz früh mit dem Test begonnen und je nach der verbleibenden Zeit können die Testdaten durch mehrere Iterationen erweitert werden. Nach jeder Iteration sollten die Funktionsüberdeckung sowie die Zahl der gefundenen Fehler zunehmen. Kritische Funktionen werden dabei zuerst geprüft, zu einem späteren Zeitpunkt weitere Routinefunktionen in den Test eingebunden. Der Test kann jederzeit abgebrochen werden, um das Produkt bzw. das Release auszuliefern. Nachteilig dabei ist, dass man dazu neigt, sich mit einem Minimum an Testdaten und einer entsprechend niedrigen Testabdeckung zufrieden zu geben [[SNEE2012](#)].

17.3 Werkzeuge zur Testdatenerstellung

Bei der Erzeugung geeigneter Testdaten können den Testdesigner sogenannte **Testdatengeneratoren** unterstützen. Dabei lassen sich verschiedene Ansätze unterscheiden, abhängig davon, aus welcher Testbasis die Testdaten abgeleitet werden:

- **Datenbankbasierte Testdatengeneratoren** verarbeiten Datenbankschemata und sind in der Lage, daraus Testdatenbestände zu generieren oder aus Datenbankinhalten Testdaten in geeigneter Form herauszufiltern. Analoges gilt für die Generierung von Testdaten aus Dateien unterschiedlichster Datenformate.
- **Codebasierte Testdatengeneratoren** erzeugen Testdaten, indem sie den Quellcode des Testobjektes analysieren. Dabei können allerdings keine Sollwerte generiert werden, denn dazu wird ein Testorakel benötigt. Auch wird (bei allen White-Box-Methoden) nur der Quellcode berücksichtigt, der existiert. Fehlerzustände aufgrund vergessener Programmanweisungen bleiben unentdeckt. Daher sind codebasierte Testdatengeneratoren nur sehr eingeschränkt sinnvoll verwendbar.
- **Schnittstellenbasierte Testdatengeneratoren** analysieren die Testobjekteschnittstelle, erkennen die Definitionsbereiche der Schnittstellenparameter und leiten z. B. mittels Äquivalenzklassen- und Grenzwertanalyse daraus Testdaten ab. Werkzeuge existieren für verschiedenste Schnittstellenarten, von Programmschnittstellen (**API – Application Programming Interface**) bis zur Analyse grafischer Benutzeroberflächen. Das Werkzeug erkennt, welche Datenfelder in einer Maske vorhanden sind (z. B. numerisches Feld, Datumsfeld) und generiert Testdaten, welche die jeweiligen Wertebereiche, z. B. per Grenzwertanalyse, abdecken. Auch hier besteht das Problem, dass keine Sollwerte generiert werden können. Die Werkzeuge sind daher gut zur automatischen Erzeugung von Negativtests geeignet, da hier keine konkreten Sollwerte relevant sind, sondern nur geprüft wird, ob das Testobjekt mit einer passenden Fehlermeldung reagiert.
- **Spezifikationsbasierte Testdatengeneratoren** leiten Testdaten und zugehörige Sollwerte aus einer Spezifikation ab. Die Spezifikation muss in diesem Fall in einer formalen Notation vorliegen. Die Aufrufreihenfolge von Methoden kann beispielsweise durch ein UML-Modell, dem **Sequenzdiagramm**, spezifiziert sein. Ein solches Vorgehen wird auch als **modellbasiertes Testen** (MBT) bezeichnet. Das UML-Modell wird mit einem CASE-Tool erstellt und dann vom Testgenerator importiert. Der Testgenerator erzeugt Testskripte, die an ein passendes Werkzeug zur Testdurchführung übergeben werden.

Ein Testdatengenerator kann gewisse Regeln anwenden (z. B. zur Grenzwertanalyse), um Testfälle systematisch zu erstellen. Der Testdatengenerator kann aber keineswegs beurteilen, ob die erzeugte Testfälle gut oder schlecht, wichtig oder irrelevant sind. Hier ist nach wie vor vom Testdesigner ein umfassendes Verständnis des Testobjektes, Erfahrung, Kreativität und Intuition gefragt. Tools können nur in sehr begrenztem

Umfang unterstützen, aber keinen Tester ersetzen. So muss z. B. die Sollreaktion meist manuell ergänzt werden. Auch die korrekte Interpretation der Testergebnisse bleibt eine wesentliche Aufgabe für den Testdesigner [SPIL2007].

17.4 Testdatentypen

Testobjekte können Benutzeroberflächen, Systemschnittstellen oder Datenbanktabellen bzw. Dateien sein. Jedes dieser Testobjekte weist eine andere Art von Testdaten auf, die auf unterschiedliche Weise erzeugt werden:

- **Datenbanken:** Beim Test von Datenbanken müssen die Funktionalität und die Struktur der Datenbank bekannt sein. Beim Test gibt es vier Datenüberdeckungsgrade:
 - **Satzüberdeckung:** Für jede Satzart bzw. jede Tabelle existiert eine Ausprägung.
 - **Attributüberdeckung:** Für jedes einzelne Attribut existieren mindestens zwei verschiedene Ausprägungen.
 - **Werteüberdeckung:** Jeder stellvertretende Wert kommt im Wertebereich eines Attributs mindestens einmal vor (Anzahl Datensätze = Anzahl stellvertretende Werte für das Attribut mit den meisten Werten).
 - **Zustandsüberdeckung:** Alle Kombinationen aller stellvertretenden Werte werden generiert.
- **Systemschnittstellen:** Beim Test von Schnittstellen sind drei Arten von Schnittstellen zu differenzieren:
 - Application Programming Interfaces (API): Schnittstellen zum Aufruf fremder Systeme
 - Messages/Nachrichten
 - Import-/Export-Dateien

Literatur

- [TEST2019] : <https://www.testing-board.com/testdatenmanagement/>, zugegriffen am 09.05.2020
- [ITDA2019] : <https://www.it-daily.net/it-management/software-development/20587-in-vier-schritten-zu-agilem-testdatenmanagement>, zugegriffen am 09.05.2020
- [SPIL2007] : Andreas Spillner, Tilo Linz, Basiswissen Softwaretest, 2007 dpunkt Verlag Heidelberg
- [SNEE2012] : Sneed, Baumgartner, Seidl: Der Systemtest, Carl Hanser Verlag München 2012

Zusammenfassung

Die Testdokumentation mit Standard IEEE 829:2008 unterscheidet verschiedene Integritätslevel und verschiedene Arten von Dokumenten, die im Rahmen des Testprozesses angefertigt werden sollen. Der Test soll in die Organisation eingebunden werden und die Testdokumentation einer Struktur folgen. Zur Testdokumentation sind geeignete Metriken zu definieren und regelmäßig zu erheben. Auch die Verfügbarkeit des Testsystems während der Projektlaufzeit soll dokumentiert werden.

18.1 Ziele der Testdokumentation und Integritätslevel

Der IEEE 829:1998 als ältere Variante wurde auch dafür eingeführt, um einen Standard für **Testdokumentation** zu etablieren. Die Ziele des Standards sind für die einzelnen Gruppen von Stakeholdern unterschiedlich:

- **Projektmanagement:** Verständnis für Risiko, Wertschätzung, Kosten, Anforderungen und Zeitplan;
- **Entwicklung:** Erkennen von kritischen Bereichen und Dokumentation des aktuell erreichten Qualitätsniveaus;
- **Test:** Entwicklung und Ausführung der Tests im vorgegebenen Zeit- und Budgetrahmen
- **Testmanager:** Dokumentation, Steuerung und Überblick über Zeit- und Budgetrahmen und Qualität des Produktes.

Um diesen Standard auf den aktuellen Stand zu bringen, wurde IEEE 829:2008 erarbeitet. Agile Projekte fanden Mitte bis Ende der 2000er-Jahre zunehmend Verbreitung. Dieser Standard soll für ein agiles ebenso wie für ein konventionelles systematisches Vorgehen

geeignet sein und bezieht sich auf Systeme wie Software. Man wollte dabei wegkommen vom bisherigen Prinzip „Dokumente folgen Aktivitäten“ hin zu einem Dokumentationsprozess, konform zu ISO 12207 (internationale Norm zur Beschreibung der Prozesse eines **Softwarelebenszyklus**) und ISO 15288 (**Systemlebenszyklus** als Norm zur Beschreibung der Prozesse innerhalb eines Lebenszyklus von einem vom Menschen geschaffenen System). Diese neue Art der Beschreibung sollte aufgabenorientiert mit Input und Output erstellt werden. Dazu wurden 4 **Integritätslevels**, je nach Fehlertoleranz des Systems, definiert und für jedes Level bestimmte Dokumente vorgeschlagen:

- **4 – Katastrophal:** Master Test Plan (statisch), Level Test Plan (dynamisch und detailliert), Level Test Design, Level Test Case, Level Test Protocol, Level Test Log, Anomaly Report, Level Integration Test Status Report, Level Test Report, Master Test Report
- **3 – Kritisch:** Master Test Plan, Level Test Plan, Level Test Design, Level Test Case, Level Test Protocol, Level Test Log, Anomaly Report, Level Integration Test Status Report, Level Test Report, Master Test Report
- **2 – Marginal:** Level Test Plan, Level Test Design, Level Test Case, Level Test Protocol, Level Test Log, Anomaly Report, Level Integration Test Status Report, Level Test Report
- **1 – Vernachlässigbar:** Level Test Plan, Level Test Design, Level Test Case, Level Test Protocol, Level Test Log, Anomaly Report, Level Test Report

In Anlehnung an das V Modell sind für den Test mehrere Stufen vorgesehen, die Dokumente Test Design, Test Case, Test Protokoll, Test Log, Interim Test Status Report und Anomaly Report werden deswegen bei Bedarf jeweils für die Stufen Component, Component Integration, System und Acceptance Test erstellt. Um den Testmanager zu unterstützen existiert aber auch eine Vorlage für einen Master Test Plan Metrics Report, der Metriken nach **ISO 91266** einschließt. Die ISO 9126 ist eine Norm für möglichst eindeutig und objektiv bewertbare Definition der Qualitätsmerkmale eines Softwareproduktes.

Eine Anpassung des Testdokumentationsprozesses an das eigene Projekt bzw. Unternehmen ist wie üblich ausdrücklich gewünscht, solange die Anforderungen an eine nachvollziehbare Testdokumentation erfüllt werden. Über die Testdokumentation hinaus befasst sich der Standard auch mit der Einbindung des Tests in die Organisation. Er definiert und beschreibt folgende Formen (ISTQB-konform):

- eingegliedert: komplett integriert im Entwicklungsteam,
- intern: Tester im Entwicklungsteam,
- integriert: eigenes Team parallel zu den Entwicklern in selber Organisationseinheit,
- eigenständig: eigene selbstständige Organisationseinheit innerhalb des Unternehmens und
- unabhängig: komplett unabhängig, außerhalb des Unternehmens, extern vergeben [[BULL2008](#)]

18.2 Struktur der Testdokumentation

Eine Testdokumentation besteht aus

- einem Planungsdokument (Testplan, Testkonzept),
- der Beschreibung der einzelnen Testfälle und Testprozeduren,
- Begleitdokumenten zu Testobjektübergaben (Testobjektlieferscheine),
- automatisierten Testskripten,
- der Beschreibung der durchgeführten Testfälle,
- regelmäßigen Testberichten,
- Fehlerberichten,
- definierten Statistiken und geeigneten Metriken sowie
- einem Testabschlussbericht mit Zusammenfassung der Testergebnisse, Bewertung und Empfehlung einer Freigabe der Software.

Das Testkonzept mit seinen einzelnen Kapiteln wird in diesem Buch bereits an anderen Stellen beschrieben.

18.3 Testskript

Ein **Testskript** definiert, wie die einzelnen Testfälle auszuführen sind. Es enthält Informationen über Voraussetzung zum Start der Testfälle und wie der Testfall auszuführen ist. Ein Testskript sollte für jeweils möglichst viele Testfälle mit ähnlichen Voraussetzungen und ähnlichem Ablauf gültig sein. Für abweichende Gruppen von Testfällen sind jeweils separate Testskripte anzugeben. Testskripte, die nur einen einzigen Testfall enthalten, sind möglichst zu vermeiden. Werden mehrere Testskripte benötigt, so erhält jedes eine eindeutige Identifikation. Es sind so viele Abschnitte einzufügen, wie es Testskripte gibt.

Die einzelnen Schritte jedes Testfalls in diesem Testskript sollten im Vorfeld feststehen, um z. B. schon vor Testfallausführung zu klären, was geschehen soll, wenn der Test fehlschlägt.

Die Beschreibung des Testablaufs soll folgende Elemente enthalten:

- Log: Es sind alle speziellen Methoden oder Formate zu beschreiben, um die Ergebnisse der Testläufe, die Zwischenfälle und sonstige wichtige Ereignisse aufzuzeichnen, z. B. Log-Datei im XML-Format.
- Vorbereitungen (optional): Hier wird dokumentiert, was zu tun ist, um die Ausführung des Testskripts vorzubereiten.
- Start: Es ist zu definieren, wie das Testskript zu starten ist.
- Ausführung (optional): Es ist zu beschreiben, was während der Ausführung des Testskripts getan werden muss, z. B. ob weitere Benutzereingaben erforderlich sind.

- Beobachtung: In diesem Abschnitt wird festgelegt, wie Messungen und Testergebnisse gewonnen werden sollen (z. B. die Antwortzeit eines Remote-Terminals ist mit einem Netzwerk-Simulator zu testen).
- Abbruch (optional): Was ist zu beachten, um wegen unvorhergesehenen Ereignissen den Test abzubrechen.
- Neustart (optional): Alle Startpunkte und das jeweilige Vorgehen werden beschrieben, an denen der Test wieder aufgenommen werden kann.
- Stopp (optional): Es wird definiert, wie das Testskript während der Durchführung des automatisierten Dauertests ordnungsgemäß angehalten wird.
- Aufräumen (optional): Es sind die Aufräumarbeiten festzulegen, um nach den Tests den ursprünglichen Zustand wieder herzustellen. Das Testsystem muss nach dem Aufräumen wieder den exakt selben Zustand wie vor der Testdurchführung haben, um einen Testlauf mit denselben Testbedingungen, aber geänderter Software oder geänderter Konfiguration oder aber auch exakt identisch wiederholen zu können.

18.4 Testfall

Ein Testfall ist eine Kombination von Eingabedaten, Bedingungen und erwarteten Ausgaben, die einem bestimmten Zweck erfüllen. Man prüft dabei, ob Vorgaben aus einem Spezifikationsdokument eingehalten werden oder ob der Programmablauf tatsächlich dem erwarteten Pfad entspricht.

Dabei müssen mindestens folgende Attribute definiert sein:

- Testfall-ID: eindeutige Identifikation mit Kurzbezeichnung. Es muss darauf geachtet werden, dass diese Identifikation im Projekt bzw. in der Organisation eineindeutig ist und nicht mehr geändert wird.
- Beschreibung der Testobjekte und Methoden, die von diesem Testfall ausgeführt werden.
- Eingabedaten, die für die Ausführung des Testfalls erforderlich sind.
- Ausgaben (Soll-Reaktion), die erwartet werden.
- Testumgebung (Konfiguration).
- Testfallspezifische Besonderheiten.
- Abhängigkeiten (ist der Testfall von der Durchführung anderer Testfälle abhängig), Testvoraussetzungen.

18.5 Testprotokoll

Das Testprotokoll enthält Angaben über alle ausgeführten Testfälle, deren Ergebnisse und evtl. Abweichungen vom erwarteten Ergebnis.

Das Testprotokoll enthält folgende Teile:

- Testprotokoll-ID: eindeutige Identifikation mit Kurzbezeichnung. Das Testprotokoll muss archiviert werden und wieder auffindbar sein. Man muss zu vergangenen Testläufen jederzeit in der Lage sein, das Ergebnis der Testdurchführung nachzuvollziehen. Es muss darauf geachtet werden, dass diese Identifikation im Projekt bzw. in der Organisation eineindeutig ist und nicht mehr geändert wird.
- Beschreibung der Testdurchführung: getestete Softwarekomponenten, Testumgebung (z. B. Betriebssystem, Datenbanken, Webserver, Applicationserver, Konfigurationsdetails).
- Aufstellung der durchgeführten Testfälle: Dabei ist die Ausführung pro Testfall mit Ergebnis (OK, NOK) je Testlauf zu dokumentieren. Dazu gehören auch Fehlermeldungen, Programmabbrüche und Meldungen, (die vom Tester eine Aktion verlangen). Evtl. Abweichungen in der Testumgebung oder speziell bei einem bestimmten Testlauf vorgenommene Änderungen der Testbeschreibung sind ebenfalls zu beschreiben. Sollten dabei unerwartete Ereignisse aufgetreten sein, so ist hier festzuhalten, was vor und nach den unerwarteten Ereignissen passiert ist. Falls Testfälle nicht ausgeführt werden konnten, werden die Umstände beschrieben, die zu diesem Problem geführt haben. Auch wenn keine unerwarteten Ergebnisse auftreten, ist dies zu vermerken.

18.6 Zusammenfassung der Testdokumentation

In einem abschließenden Teil der Testdokumentation wird eine Zusammenfassung der Testergebnisse aufgelistet. Dabei sind alle Probleme, die behandelt wurden und wie ihre Lösungen erreicht wurden, aufzulisten.

Dabei kann auf folgendes eingegangen werden:

- Umfang des Testverlaufs (Vollständigkeit) im Vergleich zu Umfangskriterien des Testplans, Auflistung der Funktionalitäten, die nicht getestet wurden mit Begründung, warum der Test nicht komplett durchgeführt wurde.
- Ggf. Abweichungen der durchgeführten Tests vom Testplan, Testskript oder Testfällen, Die Abweichungen müssen dabei begründet werden.
- Zusammenfassung der Testergebnisse und Bewertung der Softwarequalität.
- Zusammenfassung aller Hauptaktivitäten und der wichtigen Ereignisse des Tests [TESD2007].

18.7 Vernachlässigung der Testdokumentation

Oft wird man in Test und Entwicklung mit seinen vorgesehenen Arbeiten nur knapp fertig. Der Entwickler schafft es gerade noch, seine Komponente fertigzustellen und verzichtet aus Gründen der Zeitknappheit auf ausreichende Dokumentation. Beim Test läuft es ähnlich.

Leider lernen viele Mitarbeiter auch nicht aus ihren Fehlern, sondern berücksichtigen beim Schätzen ihrer Aufwände nur ihre eigene Arbeit, aber nicht den gesamten Prozess. Daher werden viele Aufwandsschätzungen zu niedrig angesetzt.

Besonders schwierig wird es, wenn der Aufwand für das gesamte Projekt nicht exakt im Vorfeld geschätzt, sondern nur ungefähre Erfahrungswerte angesetzt wurde oder man einen Fertigstellungstermin politisch durchsetzen wollte. Meiner Erfahrung nach wird auch immer wieder die Aufgabenstellung zu wenig im Detail und in zu geringem Ausmaß von denjenigen gleich zu Beginn geprüft, die das Projekt umsetzen sollen. Wenn man bereits in einer frühen Projektphase entsprechende offene Fragen aufwirft, zahlt sich das später immer aus. In der betrieblichen Praxis werden aber die Stakeholder in der Regel zu spät in diesen Prozess eingebunden, weil sie oft noch in anderen (ebenfalls zeitlich verzögerten) Projekten stecken. Man bekommt häufig das Gefühl, mit seiner Arbeit den bestehenden Herausforderungen ständig hinterherzurrennen.

Wenn das Programm aber gerade einmal in letzter Sekunde doch noch zum Laufen gebracht wurde, besteht keine Zeit mehr, die Dokumentation nachzuziehen. Das fällt dann hinten runter, weil man ja sowieso schon zeitlich im Hintertreffen ist. Die Auswirkungen einer unvollständigen oder fehlenden Testdokumentation fallen aber meist nicht sofort auf, und gerade das ist gefährlich.

Ein wesentlicher Grund für Verzögerungen und mangelnde Qualität beim Test und bei Entwicklung, für Rückfragen und offene Fragen ist nun gerade, dass in der Vergangenheit auf ausreichende Dokumentation verzichtet wurde. So pflanzt sich das Problem weiter fort: man findet nichts und wird kaum rechtzeitig fertig, weil die Dokumentation fehlt. Am Ende des Zyklus ist schließlich keine Zeit mehr vorhanden, um die gewonnenen Erkenntnisse zu dokumentieren. In einem späteren Release oder einer späteren Entwicklungsstufe ist wieder keine Zeit, weil man nach Dokumentation suchen muss, die aber leider fehlt oder wieder unvollständig erstellt wurde. Daraus entsteht schnell ein Teufelskreis.

Mit der agilen Vorgehensweise hat sich dieses Problem leider eher noch verstärkt. Im agilen Manifest heißt es schließlich „Funktionierende Software ist wichtiger als umfassende Dokumentation“.

Das wird in vielen Projekten so verstanden, dass man gar keine Dokumentation benötigt und man einen Freibrief für Nachlässigkeit hat. Der Zusatz im agilen Manifest, dass auch die Werte einer Testdokumentation hoch eingeschätzt werden, nur eben die Funktionalität im Vordergrund steht, wird dabei gar nicht mehr zur Kenntnis genommen. Das Problem liegt nicht an den theoretischen Vorgaben selbst, sondern daran, dass man nur das umsetzen will, was sich gut anhört und wenig Aufwand mit sich bringt.

Natürlich muss man in agilen Projekten genauso dokumentieren wie in klassischen Projekten. Wenn dafür nicht ausreichend Zeit bei der Sprintplanung berücksichtigt wird, kriegst man zwar kurzfristig Probleme, bezahlt aber langfristig wesentlich mehr. Leider hat sich diese Erkenntnis in vielen Unternehmen noch nicht hinreichend durchgesetzt. Eine der Probleme in diesem Zusammenhang ist das vorherrschende Quartalsdenken und der Wunsch „low hanging fruits“ umzusetzen. Wenn man zu sehr auf eine Optimierung von Renditen setzt und den Shareholder Value zur obersten Maxime macht, kommt es genau

zu diesem Effekt. Unternehmen, die eine Zeit lang in hohem Maße auf diese vermeintliche Effizienz getrimmt wurden, haben einige Jahre später oft einen umso tieferen Absturz zu verkraften. Dafür ist auch die Qualität der Testdokumentation ein Indikator.

18.8 Testabschlussbericht

Der Testabschlussbericht dokumentiert die Testergebnisse einer bestimmten Teststufe oder der gesamten Testphase. Der Testabschlussbericht ist wie folgt aufgebaut:

- Beschreibung der Funktionalität und des Testobjektes.
- Hinweise zur Wartung und zu Releases.
- Abweichungen gegenüber dem Testkonzept und Begründung der Abweichungen.
- Verlagerung von Testaktivitäten in nachfolgende Teststufen, da die Teststufe termingerecht beendet werden muss, aber nicht alle für die Teststufe vorgesehenen Testaktivitäten abgeschlossen wurden.
- Bekannte Fehler und ihr Status.
- Defects, die zum Zeitpunkt des Testabschlusses noch offen sind und Behebungsplan.
- Testabdeckung und Verweis auf eine Traceability-Matrix, die die Abdeckung pro Anforderung beschreibt.
- Beurteilung des Testfortschritts und Freigabeempfehlung.
- Risiken.
- Testautomatisierung für wiederverwendbare Arbeitsergebnisse.
- Gewonnene Erkenntnisse („Lessons Learned“) für zukünftige Testphasen und Projekte.
- Legende zur Bewertung.

18.9 Metriken

Zur Testdokumentation gehört auch die Erhebung geeigneter Indexwerte und regelmäßige Erstellung von Metriken.

Mehrere Metriken sollen projektübergreifend erfasst werden:

- Fehler nach Auswirkung gesamt;
- Anzahl gefundener Fehler, klassifiziert nach Fehlertyp/Auswirkung (z. B. Blockierend, Hoch, Mittel, Niedrig);
- Fehlerfindungsrate pro Woche gesamt;
- gefundene vs. erledigte Fehler pro Woche gesamt (**Defect Burndown Rate**);
- Aufwände für Reviews und Test absolut und im Verhältnis zu den Entwicklungskosten;
- pro Fehler Aufwand für die Fehlerbehebung;
- Aufwand für die Fehlerbehebung in Summe über alle Fehler;
- durchschnittliche Dauer der Fehlerbehebung pro Fehler;

- Entstehungsort des Fehlers (Messung des Abstands zwischen Entstehung und Entdeckung);
- je Review Aufwände für Reviews und Test.

Diese Werte sollten, getrennt nach Produktion und Test, erhoben werden.

Eine Metrik zur Definition neuer Testfälle ist dann entscheidend, wenn ein größerer Umfang neuer Testfälle erstellt wird.

Die oben aufgeführten Metriken stellen nur den Mindestumfang dar. Man kann zahlreiche weitere Metriken erheben, die eine genauere Bewertung des gesamten Testprozesses erlauben und auf versteckte Probleme hinweisen können. Dazu gehören u. a.:

- anforderungsbasierte Testmetriken,
- Metriken für die Softwarequalität,
- Metriken zur Aufwandsschätzung,
- Metriken zur Testvorbereitung und Testnachbereitung,
- Metriken zur Testplanung,
- Metriken für unterschiedliche Teststufen,
- Metriken zur Messung der Testüberdeckung,
- testfallbasierte Metriken,
- testobjektbasierte Metriken,
- Metriken zur Testdurchführung,
- Metriken für Performance- und Lasttests,
- Metriken zur Testautomatisierung,
- kostenbasierte Testmetriken,
- fehlerbasierte Testmetriken,
- Metriken für Testdokumente,
- Metriken für Usabilitytests,
- Metriken für Testendekriterien

Welche Metriken erhoben werden, wie und zu welchem Zeitpunkt die Information des Managements erfolgt, welche Beteiligte dabei informiert werden, sollte auch im Rahmen der Testvorbereitung definiert werden, es sei denn, es gibt dazu bereits eine projektübergreifende Festlegung.

Vor allem in größeren Projekten wird manchmal ein Projektcontroller zur Unterstützung der Projektleitung eingesetzt. Das **Projektcontrolling** hat folgende Aufgaben:

- Prüfung und Aktualisierung von Projektplanungen, insbesondere der Kostenplanung
- Einholen von Statusmeldungen und Prüfung der monatlichen Statusberichte;
- Durchführung spezieller Soll-Ist-Vergleiche und Organisation von Projekt-Reviews;
- Erstellung von Projektprognosen, insbesondere zu den Finanzmitteln und
- Ableiten von Handlungsempfehlungen für die Projektleitung.

Der Testmanager soll zusammen mit dem Projektcontroller die notwendigen Parameter im Vorfeld ermitteln, die zur erfolgreichen Kontrolle und Steuerung des Projektes in Bezug auf Qualitätssicherungsmaßnahmen benötigt werden, und in die Definition dafür geeigneter Auswertungen und Metriken von Anfang an mit einbezogen werden.

Bei massiver Verschlechterung der Werte aus der projektübergreifenden Festlegung über das gesamte Unternehmen hinweg sind gewisse Maßnahmen zu definieren. Das heißt nämlich im Klartext, dass die Softwareentwicklung und die Qualität in Gänze gefährdet sind und wesentliche Unternehmensziele nicht mehr erreicht werden können. Daher sollte man von Anfang an definieren, was man unter einer „massiven Verschlechterung“ versteht. Dabei sind auch die Priorität und die Auswirkungen der Fehler zu berücksichtigen.

Ziel ist es dabei, den unternehmensweiten Blick nicht zu verlieren, wenn z. B. ein Projekt besonders im Fokus steht.

Bei Erstellung eines übergreifenden Testkonzeptes sollten diese unternehmensweiten Metriken definiert werden. Außerdem muss ein verantwortlicher Mitarbeiter benannt werden, der sie regelmäßig erhebt, analysiert und interpretiert.

18.10 Verfügbarkeit des Testsystems

Während der Testdurchführung sollte auch dokumentiert werden, ob das Testsystem überhaupt zur Verfügung stand (Tab. 18.1).

In obigem Beispiel war es geplant, zum Montag, 15.06., bereits eine Teilkomponente für den Test zur Verfügung zu stellen. Da die Entwicklung aber noch nicht abgeschlossen ist und der Aufbau der gesamten Testumgebung sich verzögert, beträgt die Verfügbarkeit des Testsystems 0 %. Am Donnerstag, 18.06., wird eine Teilfunktionalität, die mit 60 % des Gesamtumfangs bewertet wird, zur Verfügung gestellt und der Test kann starten. Da die Testumgebung aber erst um 10:53 h freigegeben werden kann, werden für den Donnerstag, 18.06., nur 40 % Verfügbarkeit angesetzt und für Freitag, 19.06., und Montag, 22.06., je 60 %. Am Dienstag, 23.06., wird zwar die Gesamtfunktionalität übergeben, aber die Testumgebung muss komplett neu aufgesetzt und um konfiguriert werden. Für den Test ist sie daher an diesem Tag nicht verfügbar. Die Verfügbarkeit des Testsystems beträgt 0 %. Von Mittwoch, 24.06., bis Freitag, 26.06., schließlich stehen die Testumgebung und das gesamte Testobjekt uneingeschränkt zur Verfügung. Die Verfügbarkeit des Testsystems beträgt für diese Tage 100 %. In Summe über den gesamten Testzeitraum lässt sich dadurch eine Verfügbarkeit von 44 % ermitteln.

Damit kann man sowohl im Testabschlussbericht die Verfügbarkeit des Testsystems dokumentieren als auch für das regelmäßige wöchentliche Testreporting. Bereits bei der Erstellung des Testkonzeptes kann man sich für die Verfügbarkeit des Testsystems Ziele setzen und die Abweichung des Ziels ebenfalls laufend bewerten. Man kann diese Übersicht auch für den Fall erweitern, dass Tester nicht wie geplant zur Verfügung stehen (z. B. wegen Krankheit oder weil andere Tätigkeiten höher priorisiert wurden). Ziel ist in

Tab. 18.1 Beispiel Verfügbarkeit Testsystem

	Datum	Status	Bewertete Verfügbarkeit pro Tag	Freigabe nach Übergabe um	Verfügbarkeit in % betrachteter Zeitraum
1	Montag, 15.06.2020	Funktionalität noch nicht von der Entwicklung übergeben, Testverzug	0 %		0 %
2	Dienstag, 16.06.2020	Funktionalität noch nicht von der Entwicklung übergeben, Testverzug	0 %		0 %
3	Mittwoch, 17.06.2020	Übergabe einer Teilfunktionalität, Umgebung steht nicht zum Test zur Verfügung	0 %		0 %
4	Donnerstag, 18.06.2020	Testfreigabe, Teilfunktionalität kann getestet werden	40 %	10:53	10 %
5	Freitag, 19.06.2020	Teilfunktionalität kann getestet werden	60 %		21 %
6	Montag, 22.06.2020	Teilfunktionalität kann getestet werden	60 %		26,7 %
7	Dienstag, 23.06.2020	Übergabe der Gesamtfunktionalität, Umgebung steht nicht zum Test zur Verfügung	0 %		22,9 %
8	Mittwoch, 24.06.2020	Testfreigabe, Teilfunktionalität kann getestet werden	80 %	09:47	30 %
9	Donnerstag, 25.06.2020	Funktionalität kann getestet werden, Testsystem voll verfügbar	100 %		37,8 %
10	Freitag, 26.06.2020	Funktionalität kann getestet werden, Testsystem voll verfügbar	100 %		44 %
	Summe				44 %

allen Fällen, über den möglichen und erreichten Testfortschritt eine bessere Transparenz zu erlangen.

Mit der Erhebung der Verfügbarkeit der Testsysteme kann man die Ursachen näher erforschen, warum es im Einzelnen zu diesen Ausfällen kam und dadurch die Testprozesse zielgerichtet optimieren.

Testen findet in der Regel entwicklungsbegleitend statt, d. h. es ist normal, dass zu Beginn des Testzeitraums noch nicht alle Funktionen vollumfänglich zur Verfügung stehen. Dennoch sollte geprüft werden, welche Mindestanforderungen das Testprojekt aufweisen muss, um überhaupt mit dem Test starten zu können.

Literatur

- [TESD2007] : <https://www.ibr.cs.tu-bs.de/courses/ss07/sep-cm/templates/testdokumentation.pdf>, Softwareentwicklungspraktikum Testdokumentation, zugegriffen am 09.05.2020
- [BULL2008] : <http://robert.bullinger.over-blog.com/article-testdokumentation-nach-ieee-829-20510038.html>, zugegriffen am 09.05.2020

Zusammenfassung

Für die einzelnen Testaufgaben sind Verantwortlichkeiten festzulegen. Für einzelne Tätigkeitsbereiche sollen Rollenbeschreibungen festgelegt werden. Mit Normen (z. B. IEC 62034, IEC 62304) werden regulatorische Aufgaben und Problemlösungsprozesse beschrieben. Im Rahmen der Regressionstests ist eine Impaktanalyse der Änderungsanträge durchzuführen. Für die optimale Erledigung der Testaufgaben ist der Einsatz geeigneter Testtools vorzunehmen.

19.1 Arten von Testaufgaben

Für die **Testaufgaben** müssen alle personellen Fragen überdacht werden: „Wann, Wer, Wo, Was“. Dazu gehört auch „Wer“ kann „Was“. Sprich, wer soll welche „Testerrolle“ und welche „Testaufgaben“ übernehmen?

Zu den Testaufgaben gehören alle Tätigkeiten, die in Zusammenhang mit dem Test durchzuführen sind. Typische Testaufgaben sind:

- Erstellung des Testplans,
- Entwurf eines Testkonzeptes,
- Spezifikation der Testfälle,
- Generierung der Testskripte (Testautomatisierung),
- Bereitstellung der Testdaten,
- Aufbau der benötigten Testumgebungen,
- Durchführung der definierten Testfälle (funktionale Tests, Lasttests, Stresstest, Oberflächentests ...) für die Überprüfung der Qualitätsmerkmale,
- Protokollierung der Testergebnisse,

- Erstellung von Fehlerberichten,
- Fehlerverfolgung, Defect- und Changemanagement,
- Verfassung der Testberichte.

In der Praxis gibt es etliche Ausprägungen dieser Aufgabentypen. Dabei sollen Aufgaben in einem solchen Detaillierungsgrad definiert werden, dass sie eine oder höchstens zwei Personen in wenigen Tagen erledigen können. Je feiner die Aufgaben beschrieben sind, desto leichter ist es, den Testfortschritt zu kontrollieren [[SNEE2012](#)].

19.2 Rollenbeschreibungen

Es empfiehlt sich, dazu **Rollenbeschreibungen** für einzelne Tätigkeitsbereiche zu definieren.

1. Testmanager:

- Aufbau von weiter- und wiederverwendbaren Teststrukturen und Teststandards mithilfe fachlicher und technischer Methoden und Werkzeugen für Testdurchführung und Testdokumentation.
- Übergreifende Planung, Steuerung und Kontrolle sämtlicher Testaktivitäten. Bei Bedarf Zusammenführen von Arbeitsergebnissen der Testkoordinatoren, Ermittlung und Bewertung des Arbeitsfortschritts der Testaktivitäten und Bericht an die Projektleitung.
- Systematische Prüfung (ggf. in Stichproben) der Testdokumentation der Tester hinsichtlich formaler Korrektheit und inhaltlicher Angemessenheit der Tests, Hinweis auf Schwachstellen und Verfolgung deren Behebung (ggf. mit Delegation an einen Qualitätssicherer).
- Überwachung der Einhaltung der Vorgaben und der Konformität mit relevanten Grundsätzen, z. B. den Festlegungen ordnungsgemäßer DV-gestützter Buchführungssysteme (GoBS) und dem Organisationshandbuch (OHB) der Unternehmenseinheit.
- Organisation von Tests für Basisänderungen (Service-Packs, Refresh-Maßnahmen)

2. Testkoordinator (teilweise wird ein Testkoordinator Fachbereich und ein Testkoordinator IT mit verteilten Zuständigkeiten eingesetzt):

- Planung, Koordination und Steuerung der internen fachlichen Testvorbereitung, Testdurchführung, Testauswertung und Testdokumentation für den Funktions- und/oder den Funktionskettentest.
- Planung und Steuerung der Testressourcen.
- Definition der Anforderungen an die Testumgebung und Teststammdaten gemeinsam mit dem Testumgebungsmanager.
- Detailausarbeitung des Testkonzeptes bezüglich der fachlichen Tests. Identifikation, Definition und Priorisierung der fachlichen Testobjekte.

- Ggf. Unterstützung der Tester bei der Bewertung von Testergebnissen und Priorisierung von Fehlern (Defects).
- Durchführung der Risikoeinstufung für Testobjekte, sofern diese nicht bereits vom Fachkoordinator bzw. Auftraggebern vorgenommen wurden.
- Formale Qualitätssicherung des fachlichen Tests (Überprüfung der Testfälle, Ermittlung des Testabdeckungsgrads und Risikoeinschätzung).
- Durchführung des dezentralen Defectmanagements (Zuordnung von Fehlern an Entwickler des Projektteams, Verfolgung der Fehlerbehebung).
- Ermittlung des Testabdeckungsgrads und Einschätzung der Testrisiken.
- Detailausarbeitung des Testkonzeptes bezüglich der Tests (Testobjekte, Risikoeinstufung).
- Entscheidung über Nachtests und Verfolgung der Nachtests.
- Erteilung von Freigabeempfehlungen für das gesamte getestete Softwarepaket aus Sicht des Testers.
- Coaching von Testern bei der Anwendung der Standards für Testmethodiken.
- Erster Ansprechpartner für Tester und Entwickler hinsichtlich aller testbezogenen Fragen.
- Ausarbeitung des Testkonzeptes bezüglich der Tests (Testobjekte, Risikoeinstufung).

3. **Testumgebungsmanager:**

- Klärung der Anforderungen an die Testumgebung und die Teststammdaten mit dem Testkoordinator. Beauftragung und Verfolgung des Aufbaus der Testumgebung und der Verfügbarkeit der Teststammdaten.
- Klärung von Fehlern und Störungen in der Testumgebung.
- Planung und Information über bekannte, in der Zukunft liegende Ausfallzeiten der Testumgebung.
- Prüfung der Testumgebung nach Refreshs und Einspielen von Service-Packs.
- Pflege und Überwachung der Belegungsplanung und ggf. Abstimmung bei Terminkonflikten.
- Erster Ansprechpartner für Tester und Entwickler hinsichtlich aller testbezogenen Fragen.

4. **Defectmanager:**

- Information der Anwendungen über offene Defects und Pflege des Defectstatus.
- Globales Tracking für Fehler, vor allem, wenn sie unbearbeitet oder ungelöst im System stehen und Verfolgung der Bearbeitung und erinnert an die Verantwortlichkeit.
- Einberufung von Meetings für besondere, mehrere Instanzen betreffende Defects.
- Lieferung des Defectstatus an den fachlichen Testkoordinator und Entscheidungsgrundlage für die Freigabe des Transports.

5. **Testanalyst (teilweise unterschiedliche Rollen für Testanalyst Fachbereich und Tester IT):**

- Dokumentenreview des Fachkonzeptes und des Systemdesigns.

- Ermittlung und Erstellung von Testfällen und Abstimmung mit dem Autor des Fachkonzeptes.
- Ermittlung und Erstellung von Testdaten, ggf. mit Unterstützung durch die IT-Abteilung.
- Durchführung der geplanten Tests und daraus entstehender Re-Tests.
- Dokumentation der Testergebnisse.
- Bewertung der Testergebnisse, bei Feststellung von Abweichung Festlegung der Schwere der Abweichung/des Defects bzw. Mangels.
- Automatisierung von Tests, ggf. in Zusammenarbeit mit Testautomatisierer [[HETT2019](#)].

Die Testaufgaben sind den einzelnen Verantwortlichen zuzuordnen. Der Status der Testaufgaben (offen, in Arbeit, zurückgestellt, erledigt wird im laufend zu führenden Teststatusbericht verfolgt).

Da z. B. die Spezifikation der Testfälle in Unit-, Integrationstest und Systemtest in der Regel von unterschiedlichen Mitarbeitern durchgeführt wird, sind die Testaufgaben soweit möglich und sinnvoll dabei den einzelnen Teststufen zuzuordnen. Im Testkonzept müssen sinnvolle Integrationsschritte definiert und exakt spezifiziert werden.

19.3 Norm IEC 62034

Die Norm beschreibt regulatorische Forderungen an den Software- und Systemtest. Diese Norm stammt ursprünglich aus der Elektrotechnik und wird vor allem auch in der Medizintechnik für Testaktivitäten verwendet, rein inhaltlich kann man das Beschriebene aber auch gut auf andere Testaufgaben übertragen.

Diese Norm fordert die Prüfung des Softwaresystems für Software der Klasse B und C. Diese Anforderung weitet das **Amendment I** (IEC 62304:2006 A1:2016) auf alle Software, d. h. Software der Klassen A, B und C aus. Die Klassen sind dabei wie folgt definiert:

- Klasse A: Keine Verletzung oder Schädigung der Gesundheit ist möglich.
- Klasse B: Keine schwere Verletzung ist möglich.
- Klasse C: Tod oder schwere Verletzung ist möglich.

Das Amendment fordert Aufzeichnungen aus dem Software-Systemtest sogar für Legacy Software. Daher kann ein Auditor kann mit Bezugnahme zum Stand der Technik einen Software-Systemtest für alle Softwaresysteme verlangen.

Die Norm stellt folgende Anforderungen an die Software-Systemtests:

- **Vollständigkeit:** Die Erfüllung aller Anforderungen an das System sind zu prüfen, typischerweise durch Tests, und durch ein „Tracing“ nachweisen, dass Ihnen das gelungen ist.
- **Pass-Fail-Kriterien:** Es müssen Kriterien festgelegt werden, wann eine Prüfung als erfolgreich einzustufen ist.
- **Dokumentation und Reproduzierbarkeit:** Die Tests (Spezifikation, Kriterien, Ergebnisse) müssen so dokumentiert sein, dass es möglich ist, sie einschließlich der Ergebnisse zu reproduzieren. Das schließt eine Dokumentation und Konfigurationskontrolle der Prüfumgebung mit ein.
- **Problemlösungsprozess:** Wenn beim Test Fehler gefunden werden, muss ein definierter Problemlösungsprozess gestartet werden (► unten).
- **Wiederholung bei Änderungen:** Falls die Software geändert wird, müssen die Tests in angemessener Weise wiederholt werden. (Es steht also nicht geschrieben, dass immer sämtliche Tests wiederholt werden müssen.)
- **Geeignete Teststrategie:** Die Testmethodik/Teststrategie muss auf Eignung geprüft werden. Wer beispielsweise bei Anwendungen, die zeitkritisch sind oder bei denen „Race-Conditions“ auftreten können, das Laufzeitverhalten nicht testet, würde dieser Forderungen nicht gerecht.

19.4 Problemlösungsprozess

Mit der **Norm IEC 62304** ist gefordert, dass Fehler, die man beim Software-Systemtest findet, in den Problemlösungsprozess eingespeist werden. Dazu sollte man sich im Vorfeld Gedanken machen. Der Problemlösungsprozess sollte unternehmensweit gelten und für alle Projekte einheitlich definiert sein.

Folgende Grundsätze gelten dabei:

1. Fehler vermeiden oder vor dem Systemtest finden

Wenn man in den Software-Systemtests auf sehr viele Fehler stößt (wobei hier exakt zu definieren ist, was „sehr viele“ bedeutet), heißt das, dass die vorangegangenen Quality Gates (z. B. Review der Anforderungen, Code Review, Unittests) nicht ausreichend wirksam waren. Hier sollte unbedingt angesetzt werden, denn die Fehler, die in den Systemtests gefunden werden, sind immer nur die Spitze des Eisbergs. Diese Fehler deuten auf ein wirkliches Qualitätsproblem hin.

2. Vorgehen in den Sprints definieren

In einigen Unternehmen werden die Tests bei den einzelnen Sprints nur rudimentär durchgeführt und aufgetretene Fehler nicht über ein Tool verwaltet bzw. nur intern kommuniziert. Ein „offizieller“ Software-Systemtest wird nur einmalig nach Integration aller Subsysteme direkt vor dem Release durchgeführt. Nur bei diesem letzten Test werden Probleme über den definierten Problemlösungsprozess abgewickelt. Dieses Vorgehen ist zwar erlaubt, aber nicht zu empfehlen, weil man letztlich den Prozess

aushebelt und Schattenprozesse im Unternehmen etabliert. Auch hier sind vermeintliche Einsparungen der Auslöser, die in der Praxis zu erheblichen zusätzlichen Kosten wegen mangelhafter Qualität führen. Es ist wichtig, in Prozessen zu denken. Denken in Prozessen bedeutet für ein Unternehmen manchmal auch eine Änderung der Organisationsstruktur. Nicht mehr die klassische **Aufbaustruktur** mit dem typischen Abteilungsdenken ist maßgeblich, sondern die **Ablaufstruktur** mit den festgelegten Prozessen rückt ins zentrale Blickfeld. Dies bedeutet, dass die klassischen Systemgrenzen aufgebrochen werden und die einzelnen Teilaufgaben in den Abteilungen zu einem Gesamtprozess integriert werden müssen. Damit sind auch die Schnitt- oder Nahtstellen an den einzelnen Übergängen zwischen den Organisationseinheiten zu beseitigen. Problematisch dabei ist, dass Egoismen und Eigeninteressen genau diese Änderungen oft vermeiden und damit die Organisation dem agilen Vorgehen nicht folgen kann. Oft bemerke ich in Projekten, dass die betriebliche Organisation die für Projekte notwendige Dynamik eher bremst, anstatt sie zu fördern. Dafür gibt es in der betrieblichen Praxis immer wieder Beispiele. Ich bezweifle, dass die Tragweite der Konsequenzen dem Topmanagement in vollem Umfang bewusst ist.

3. **Schlanker Problemlösungsprozess:** Entscheidend ist dabei, einen schlanken Problemlösungsprozess zu definieren. Es darf niemand wegen des damit verbundenen Overheads in Versuchung gebracht werden, den Problemlösungsprozess zu umgehen. Zu einem schlanken Prozess zählt, dass
 - die Problemberichte schnell und einfach ausgefüllt werden können.
 - keine unnötigen Freigaben und Unterschriften verlangt werden.
 - der Prozess keine unnötigen Arbeitsschritte verlangt.
 - nur die Daten erfasst werden, die regulatorisch verlangt sind oder später wirklich ausgewertet werden [[JOHN2019](#)].

19.5 Testaufgaben im Regressionstest

Auch für den regelmäßigen Regressionstest sind Testaufgaben zu definieren. Diese Definition findet normalerweise projektübergreifend statt. Dazu gehören:

- laufende Anpassung des Testplans,
- Entwurf eines Regressionstestkonzeptes,
- Auswahl alter Testfälle für Regression und neuer Testfälle für Erweiterung und Änderung von Funktionen,
- Änderung bestehender und Generierung aktueller Testskripte,
- Bereitstellung der Testdaten,
- Prüfen der vorhandenen Testumgebung, ggf. Bereitstellung sowie
- Durchführung der Regressionstests.

Tab. 19.1 Beispiel einer Impaktanalyse

Quellobjekt						
Projekt	ID	Name	Typ	Zugeordnet	Typ	Pfad
Integriertes Sample-Set	AIS-ASTK-2	Abschaltung bei geringer Geschwindigkeit	Requirement des Auftraggebers		Quellobjekt	AIS-ASTK-2
Direkte Beziehungen						
Projekt	ID	Name	Typ	Zugeordnet	Typ	Pfad
Integriertes Sample-Set	AIS-ASYS-50	Geschwindigkeits-Sensoren	System-Requirement		Abgeleitet von	AIS-ASTK-2 → AIS-SYS-50
Integriertes System-Sample-Set	AIS-VAL-12	Abschaltung bei Geschwindigkeit über 6 km/h	Validierung		Validiert von	AIS-ASTK-2 → AIS-VAL-12
Zweiter Grad der Ableitung						
Projekt	ID	Name	Typ	Zugeordnet	Typ	Pfad
Integriertes System-Sample-Set	AIS-BUG-4	Entwicklung über 6 km/h	Defect		Bezogen auf	AIS-ASTK-2 → AIS-VAL-12 → AIS-BUG-4

Im Rahmen der Regressionstests ist eine **Impaktanalyse** der Änderungsanträge durchzuführen. Ziel dieser Aufgabe ist es, die für den Regressionstest erforderlichen Testfälle zu identifizieren [SNEE2012].

In folgender Tabelle ist eine einfache Impaktanalyse zu sehen. In diesem Beispiel soll die Anforderung AIS-ASTK-2 untersucht werden. Voraussetzung für eine Impaktanalyse ist eine gepflegte Traceability. In diesem Beispiel wurden zwei Verknüpfungen „nach unten“ gezogen. Zum einen wurde eine Systemanforderung verknüpft (AIS-ASYS-50). Zum anderen wurde auch ein Akzeptanztest angelegt (AIS-VAL-12). Doch eines dieser Elemente hat ebenfalls eine Verknüpfung: Der Test wurde wohl nicht bestanden und ist daher mit einem Defekt verknüpft (AIS-BUG-4). In der Spalte „Path“ ist das zu sehen. Eine Impaktanalyse (Tab. 19.1) kann über beliebig viele Stufen laufen [SETR2019].

19.6 Testtools

Der Einsatz geeigneter Tools ist entscheidend für den Projekterfolg. Leider wird hier häufig an der falschen Stelle gespart. Ich habe schon mehrfach erlebt, dass Softwarelizenzen, die nur wenige Tausend EUR gekostet, jedoch die Testdurchführung erheblich erleichtert

hätten, aus „Kostengründen“ nicht beschafft wurden. Wenn ein Tester für bestimmte Aufgaben ohne geeignetes Tool längere Zeit benötigt als mit dem Tool, kostet er evtl. ein Vielfaches der Kosten für das Tool. Dennoch werden immer noch unproduktive Abgleiche von Listen und redundante Datenhaltung betrieben, was zu Fehlern und Abstimmungsproblemen führt, was Projekte erheblich verzögern kann.

Diese Kosten werden aber nicht direkt deutlich. Man hinterfragt sie teilweise gar nicht mehr. Aus Erfahrung setzt der Testmanager für einen gewissen Testumfang 3 Monate an, weil er schon weiß, dass es Rückfragen geben wird, dass Tests wiederholt werden müssen, dass die Testumgebung von Anfang nicht richtig zur Verfügung steht und dass Leerlaufzeiten im Projekt entstehen. Der Testmanager weiß auch, dass es andere Phasen gibt, wo man kaum noch pünktlich fertig wird und sogar Ressourcen verstärkt werden müssen. Interessanterweise werden die 3 Monate meist nicht weiter hinterfragt, wenn man auf ein ähnliches früheres Projekt referenziert. Aber schon dieses Vorbild lief alles andere als ideal. Die Kosten für eine Softwarelizenz hingegen sind transparent und schlagen sich sofort auf die Kalkulation durch, die Kosten für Verzögerungen im Arbeitsablauf sind in der betrieblichen Kalkulation unsichtbar und schlagen sich nur am Ende in einer mäßigen Rendite nieder.

Wenn die Testprozesse besser wären, würde der Testmanager im obigen Beispiel vielleicht nur 2 Monaten für denselben Testumfang benötigen und gleichzeitig höhere Qualität liefern können. Das ist bekannt, aber schwierig zu ändern, und weil man schon weiß, dass man das gewünschte Tool sowieso nicht durchs Controlling bekommt, oder dass Abläufe, auch wenn sie suboptimal laufen, nicht geändert werden, da jede Änderung eines Prozesses auch ein gewisses Risiko mit sich bringt, plant man nach wie vor mit den höheren Personalzahlen ohne den Nutzen exakt zu hinterfragen.

19.7 Aufgaben für die einzelnen Stakeholder

Ferner ist wichtig, dass den einzelnen Stakeholdern ihre Aufgaben bekannt sind und sie wissen, was zu tun ist. Auch die Abgrenzung zwischen den einzelnen Bereichen ist entscheidend für den Testerfolg. Sonst kann es zu der Situation kommen, dass einzelne Funktionen mehrfach in unterschiedlichen Teststufen getestet werden und andere Funktionen hingegen gar nicht. Auch das ist in der betrieblichen Praxis häufig anzutreffen. Fachliche Reviews der Testfälle über mehrere Teststufen hinweg sind daher unbedingt zu empfehlen. Leider fehlt gerade dazu im Alltag oft die Zeit.

Literatur

[JOHN2019] : <https://www.johner-institut.de/blog/iec-62304-medizinische-software/software-systemtest/>, zugegriffen am 09.05.2020

[SETR2019] : <https://se-trends.de/was-ist-eine-impaktanalyse/>, zugegriffen am 09.05.2020

-
- [HETT2019] : <https://www.hettwer-beratung.de/konzepte/testkonzept/testrollen/>, zugegriffen am 09.05.2020
- [SNEE2012] : Sneed, Baumgartner, Seidl: Der Systemtest, Carl Hanser Verlag München 2012



Zusammenfassung

Für die Testdurchführung sind geeignete Testumgebungen einzurichten. Das Testumgebungsmanagement umfasst Konfigurationsmanagement, Verfügbarkeitsmanagement, Zugriffsmanagement, Changemanagement und Releasemanagement. Für das Hosting von Testumgebungen können externe Testumgebungen und Cloudservices wie AWS oder private Clouds verwendet werden.

20.1 Notwendigkeit von Testumgebungen

Für die Durchführung von Softwaretests ist es notwendig, eine oder mehrere separate **Testumgebungen** zur Verfügung zu stellen. In der produktiven Umgebung darf niemals getestet werden. Das ist notwendig, weil durch Nutzung der Produktivumgebung für die Testausführung ein unkalkulierbares Risiko für die Produktion entsteht. Ohnehin ist es teilweise regulatorisch so gefordert. Fehler im Testobjekt können sich auf produktive Systeme oder Daten auswirken, sodass sie nicht mehr als vertrauenswürdig betrachtet werden können. Ein weiteres Risiko für die Nutzung der Produktivumgebung besteht darin, dass ein Absturz des Testsystems die Nutzung des Produktivsystems in der gleichen Umgebung verhindern könnte und somit zu Verlusten durch entgangene Geschäfte oder Verzögerungen in der Produktion führt, also zum Beispiel Produktionsausfälle und dadurch Sachfolgeschäden entstehen können [BOCH2010].

Oft wird eine Testumgebung, die sehr entwicklungsnahe ist, auf der ggf. einzelne noch nicht fertige Komponenten mit Mocks oder Stubs ersetzt werden, für den Modul- und Integrationstest, eine Testumgebung ohne Mocks und Stubs für den Systemtest und eine

produktionsnahe Umgebung für den Abnahmetest verwendet. Sollten nichtfunktionale Dauertests (z. B. Lasttests, Performancetests) parallel zum Systemtest terminiert sein, ist eine weitere Testumgebung zu definieren.

Sollten für ein Softwarerelease mehrere Projekte parallel zur Einführung anstehen, ist es ggf. ratsam, die Testdurchführung auf unterschiedlichen Testumgebungen durchzuführen, weil sich Änderungen aus einem Projekt sonst negativ auf die Testdurchführung eines anderen Projektes auswirken können. Das ist besonders dann relevant, wenn während der Testdurchführung die Datenbasis verändert wird und nicht mehr ohne weiteres auf den definierten Ursprungszustand zurückgesetzt werden kann. Für jeden Test ist es wichtig, dass die Umgebungsbedingungen genau definiert und bekannt sind, um Fehler genau eingrenzen und exakt nachweisen zu können und um nicht für irgendwelche Beobachtungen Fehler zu erfassen, die gar keine sind.

20.2 Angemessenheit der Testumgebung

Grundsätzlich sollte die Testumgebung umso produktionsähnlicher werden, je näher die Teststufe an der produktiven Nutzung ist. Für Komponententests auf Entwicklerebene sind häufig die Entwicklungscomputer oder Emulatoren der Zielsysteme als Testumgebung ausreichend. Demgegenüber sollte für den Abnahmetest eine Testumgebung zur Verfügung stehen, welche mit der Produktionsumgebung vergleichbar ist. Ein exaktes Abbild ist dann notwendig, wenn z. B. Datenmigrationen in Produktivsystemen (Fusionen) zu testen oder Last- und Performancetests durchzuführen sind.

In der Regel kann auf Grundlage einer systematischen Testfallermittlung mit einem reduzierten Datenbestand getestet werden, welcher nur einen Bruchteil des Produktivdatenbestands umfasst. Unterscheidet sich die Testumgebung jedoch grundsätzlich (also nicht nur in Bezug auf ihre Leistungsfähigkeit) von der Produktivumgebung, so besteht das Risiko, dass im Test Fehler übersehen werden oder vermeintliche Fehler gefunden werden, die gar keine Fehler sind („Phantomfehler“).

20.3 Testumgebungsmanagement

Da die Testumgebungen ein möglichst reales Abbild der Produktionsumgebungen sein müssen und in den meisten Organisationen eine komplexe Systemlandschaft vorzufinden ist, sind auch die Testumgebungen entsprechend komplex. Daher ist ein adäquates Testumgebungsmanagement für Konfiguration, Administration und Betrieb dieser Umgebungen erforderlich. Dabei sind folgende Aspekte sicherzustellen:

- **Konfigurationsmanagement:** Kenntnis der Inhalte der Testumgebung;
- **Verfügbarkeitsmanagement:** Sicherstellung der dem Plan entsprechenden Verfügbarkeit
- **Zugriffsmanagement:** Einrichtung benötigter Berechtigungen und Zugriffe und
- **Changemanagement und Releasemanagement:** geordnetes Änderungswesen und Bündelung.

Dabei ist es empfehlenswert, in der Testumgebung dieselben Werkzeuge und Konfigurationsmanagementsysteme wie in der Produktivumgebung zu verwenden. Das gilt auch für die verwendete Hardware und Infrastruktur. Test- und Produktivumgebung sollten sich, abgesehen von der Leistungsfähigkeit, lediglich in der Version des Testobjektes unterscheiden.

Die termingerechte Bereitstellung der Testumgebung (Verfügbarkeitsmanagement) und die regelmäßige Bestückung mit aktualisierten Daten und Software sind ebenfalls zu berücksichtigen. Verzögerungen durch die späte oder unvollständige Bereitstellung von Testumgebungen führen zu Verzögerungen im Testprozess und damit für den gesamten Projektverlauf und unproduktive Leerlaufzeiten. Die Testprozesse sollen dafür im Vorfeld skizziert werden, um notwendige Aktionen termingerecht durchführen zu können. Erfahrungsgemäß muss die Verfügbarkeit der Testumgebung allen Testern transparent gemacht werden, damit nicht Tests durchgeführt werden, für die die Testumgebung gar nicht ausreichend vorbereitet ist. Zur Bereitstellung der notwendigen Zugriffsrechte auf die Testumgebung ist insbesondere darauf zu achten, dass Tester keinen Zugriff auf sensible Daten erhalten, die außerhalb ihrer Befugnisse sind und durch ihre regulären Testtätigkeiten keine anderen Tests verhindern. Das gilt in besonderem Maße dann, wenn automatisierte Skripts die Daten manipulieren und deshalb für andere Testfälle nicht mehr in der benötigten Ausprägung verfügbar sind.

Ein wichtiger Punkt für das Management der Testumgebung ist, dass es keine unkontrollierten Änderungen innerhalb der Testumgebung geben darf. Änderungen sind nicht immer vermeidbar, da sich zum Beispiel die Produktionsumgebung ändert und diese Änderung in den Testumgebungen berücksichtigt werden müssen. Bei einer Ausführung unkontrollierter Änderungen an der Testumgebung kann es vorkommen, dass das Testergebnis keine verlässlichen Aussagen zu den Testzielen liefert und somit wertlos ist. Deswegen muss jegliche Änderung durch ein Changemanagement geplant, kommuniziert, gesteuert und überwacht werden, um sicherzustellen, dass Änderungen keine Auswirkung auf abgeschlossene, laufende oder geplante Tests haben. Sind durch notwendige Anpassungen Testauswirkungen aufgetreten, so müssen sie mit allen Beteiligten abgestimmt und in einem angepassten Testplan dokumentiert werden. Auch kontrollierte Änderungen an der Testumgebung können eine komplette Wiederholung aller bisher durchgeführten Testfälle nach sich ziehen, und diese Tatsache muss in der Terminplanung des Projektes berücksichtigt werden.

20.4 Externe Testumgebungen

Bei der Zusammenarbeit mit externen IT-Dienstleistern kann es dazu kommen, dass externe Testumgebungen zum Einsatz kommen. Hinzu kommt die notwendige Systemintegration bei externen Testumgebungen, wenn auch Daten mit eigenen Systemen ausgetauscht werden. In einem solchen Fall sind durch den Testmanager mit dem Betreiber der Software folgende Punkte abzustimmen:

- Art der benötigten Testumgebung(en),
- Umfang und Zeitrahmen der Tests,
- benötigte Testdaten und deren Management sowie
- Änderungen durch die neue Releaseeinführung.

Dabei ist zu beachten, dass auf allen an der Testumgebung beteiligten Systemen konsistente Daten vorliegen. Dafür sollten zusammen mit dem externen Dienstleister Prozesse definiert werden, die diese Konsistenz gewährleisten. Die Abstimmung und Einführung dieser Prozesse benötigt erfahrungsgemäß einen gewissen Vorlauf und sollte daher bereits in einer frühen Projektphase stattfinden.

Die Bereitstellung der Testumgebung erfolgt auf unterschiedlichen Wegen. Diese lassen sich in zwei Gruppen unterteilen, wobei auch Mischformen existieren:

- **Interne Bereitstellung:** Der IT-Dienstleister baut eine eigene Testumgebung beim Nutzer vor Ort auf.
- **Externe Bereitstellung:** Der IT-Dienstleister baut eine Testumgebung im eigenen Unternehmen auf. Der Nutzer greift auf eine Remoteverbindung auf diese Testumgebung zu.

Bezüglich der Verwendung von **Zugriffsrechten** lassen sich folgende Varianten (und deren Kombinationen) unterscheiden:

- **Exklusive Zugriffsrechte:** Die Testumgebung steht einem Tester bzw. Teilprojekt exklusiv zur Verfügung. Zugriffe anderer Tester oder Teilprojekte sind ausgeschlossen.
- **Teilexklusive Zugriffsrechte:** Die Testumgebung steht nur teilweise exklusiv einem Tester oder einem Teilprojekt zur Verfügung. Hierbei ist genau zu klären, welche Teile exklusiv und welche Teile nicht exklusiv zur Verfügung gestellt werden. Die Testfälle sind darauf abzustimmen. Testfälle, bei denen Kollisionen aufgrund der nicht exklusiven Nutzung entstehen können, sind entsprechend zu kennzeichnen und mit den weiteren Nutzern abzustimmen, um eine Beeinträchtigung der Testdurchführung zu verhindern. Dabei ist auch zu prüfen, inwiefern eine Kapselung der Testumgebung oder die Einrichtung unterschiedlicher Mandanten auf der Testumgebung möglich ist.

- **Institutsexklusive Zugriffsrechte:** Dieses Merkmal trifft vor allem für Tests im Bankensektor zu. Alle Tester bzw. Testteilprojekte eines Instituts müssen sich die Testumgebung teilen und die Testaktivitäten institutsintern abstimmen.
- **Nicht exklusive Zugriffsrechte:** Die Testumgebung wird durch den IT-Dienstleister vielen Kunden gleichzeitig zur Verfügung gestellt. Die Abstimmung von Testzeiten bzw. Testfenstern über viele Organisationen hinweg ist durch den IT-Dienstleister in Form eines Testkalenders und exakter Ablaufpläne zu gewährleisten. Bei dieser Form der Bereitstellung ist besonders darauf zu achten, dass keine vertraulichen oder personenbezogenen Daten untereinander einsehbar sind.

20.5 Testumgebung in der Cloud

Für das Hosting von Testumgebungen existieren einige Möglichkeiten – die wohl spannendste ist die Cloud. Vor einigen Jahren noch wenigen Leuten ein Begriff, ist die Cloud heute vor allem im privaten Bereich, wie zum Beispiel vom Smartphone (Dropbox, Evernote) oder TV (Netflix, Amazon Prime), nicht mehr wegzudenken. Im Businessbereich stoßen große Anbieter mit Endanwender-Lösungen in neue Marktbereiche vor, wie beispielsweise Microsoft mit Office365. Abseits von diesen offensichtlichen Einsatzmöglichkeiten, bietet die Cloud aber auch Lösungen für alltägliche Probleme in den Bereichen Softwaretest und Entwicklung – und dafür auch für das Hosting von Testumgebungen.

Unter „Cloud“ versteht man in diesem Zusammenhang die Services zur Anmietung von Hardware von großen Anbietern, wie Amazon oder Microsoft. Besonders bei Microsofts Azure Cloud, wird darauf Wert gelegt, gängige Microsofttools (z. B. Visual Studio, Sharepoint oder Team-Foundation-Server) zu integrieren. Folgende Vorteile ergeben sich dabei für IT-Organisationen:

- versionierte Quellcodeablage,
- automatisierte Build- und Deploymentprozesse,
- nahtlose Einbindung des Fehlermanagementsystems,
- Monitoring der eingesetzten Hardware und
- transparente Darstellung der anfallenden Kosten.

Amazon Web Services (AWS) wiederum bietet einen großen Marketplace, in dem für geringe Cent-Beträge, Templates für Maschinenkonfigurationen bezogen werden können. Hier können mit wenigen Mausklicks virtuelle Server mit einer Reihe von vorinstallierten Tools (wie z. B. Apacheserver) hochgezogen werden. Der negative Nebeneffekt ist, dass Amazon und Google auch hier wieder Informationen über das Nutzerverhalten bekommen und mit den zusätzlichen Umsatzpotenzialen zusätzliche Macht und Werbepotenziale erhalten, so wie das für zahlreiche Bereiche des privaten Lebens bereits gilt, sei es für Gesundheitsdaten oder Bewegungsprofile.

Im Vergleich zu Cloud-only-Lösungen wird in der Praxis oftmals ein „**On-Premise-Ansatz**“ gelehrt, bei dem die Hardware für Test- und Entwicklung von und im eigenen Unternehmen gehostet wird. Der große Vorteil dabei ist, dass die Organisation die gesamte Administration alles selbst in der Hand hat. Dies ist aber auch zugleich der größte Nachteil: Es entsteht viel Aufwand für Wartung und Betrieb. Mag diese Lösung also für kleine oder mittelgroße Organisationen bzw. Projekte ein gangbarer Weg sein, stößt man bei größeren Unterfangen recht schnell an Grenzen.

Eine andere Möglichkeit sind externe Rechenzentren oder Neudeutsch „**private Cloud**“, bei denen ein Drittanbieter die Betreuung der Infrastruktur übernimmt. Der Unterschied zu „echten“ Cloud-Lösungen ist, dass in diesem Fall normalerweise nicht nur der Betrieb, sondern auch die Steuerung und der Aufbau vom Drittanbieter übernommen werden. Es muss also für jeden neuen Server oder Cluster ein Antrag beim Drittanbieter gestellt werden, dessen Bearbeitung je nach Verfügbarkeit einige Zeit beanspruchen kann.

Bei der Nutzung von Cloud-Lösungen liegt aber genau hier der Unterschied: Über die einfach zu bedienenden und übersichtlichen Webinterfaces von Amazon und Microsoft, wird die Steuerung und der Aufbau von der Wartung und Pflege der Hardware entkoppelt.

Die Vorteile liegen auf der Hand – der Anbieter garantiert mehr oder weniger eine 24/7-Verfügbarkeit und Wartung, während der Nutzer es selbst in der Hand hat, wie die Hardware genutzt und kombiniert wird. Natürlich sind auch Hybridansätze möglich, bei denen in-house gehostete Testumgebungen um zusätzliche Instanzen aus der Cloud erweitert werden. Die Möglichkeiten sind hier sehr vielfältig und benötigen in der Planungsphase besondere Aufmerksamkeit, damit für die Umsetzung in der Cloud der richtige Ansatz gewählt wird und die Testumgebung am Ende dem gelebten Alltag entspricht.

Ein beispielhafter Ablauf zur Erstellung von Testumgebungen mittels AWS und [Docker](#) bzw. [Puppet](#), kann im Groben etwa so aussehen (Abb. 20.1):

Zur Bedienung der AWS-Services können, neben der Weboberfläche, auch Befehle auf Commandline verwendet werden. Somit kann auch das AWS-Setup per Script automatisiert werden. Das Setup geschieht in mehreren Schritten:

- Konfiguration: die gewünschten Hardware- und Systemparameter werden festgelegt (z. B. CPU, RAM, Betriebssystem, IP-Adresse), mit denen eine bestimmte Instanz (sprich Server) ausgestattet sein soll.
- Initiale Instanzierung: mittels dem „Run-instances-Befehl“ entsteht aus der Konfiguration der neue Server. Auf diesen verbindet man sich per Remote-Desktop-Connection, um das Setup durchzuführen.
- Setup: in einem weiteren Schritt können nun Docker und Puppet aufgespielt werden, damit dieser Server in weiterer Folge als Zielmaschine dieser beiden Tools verwendet werden kann. Eine AWS-Maschine kann aber auch für anderes zum Einsatz kommen, z. B. als:
 - Lizenzserver,
 - Repository,
 - Server für ein Drittsystem, welches in der Testumgebung benötigt wird oder
 - Monitoring-Instanz.

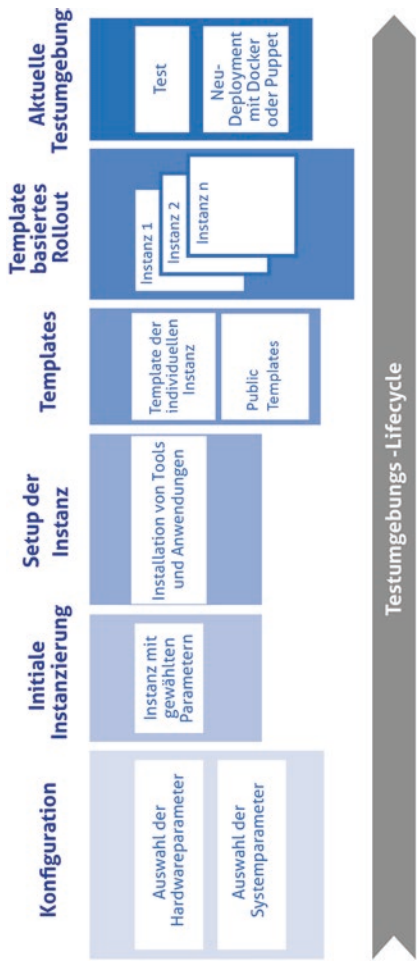


Abb. 20.1 Beispielhafter Ablauf zum Setup einer Testumgebung in AWS

Aus einer AWS-Maschine kann ein Template erstellt werden. Es sind aber bereits von der Community vorgefertigte Templates im Marketplace vorhanden. Diese Templates können in weiterer Folge wieder mit unterschiedlicher Hardwarekonfiguration als neue Instanz gestartet werden. So kann z. B. das Systemverhalten bei Last- und Performance-tests auf verschieden dimensionierter Hardware geprüft werden. Das Starten der Instanzen – basierend auf den Templates – passiert mit dem Rollout.

Am Ende dieses Prozesses erhält man eine oder mehrere aktuelle Testumgebungen. Ändern sich nun einzelne Systeme, die auf diesen Umgebungen laufen, steht einem flexiblen Neu-Deployment mittels Puppet oder Docker nichts im Weg. Alternativ dazu, muss das zugrunde liegende AWS-Template verändert und der Rollout neu gestartet werden. Bei dieser Variante ergibt sich allerdings der Nachteil, dass der Rollout von vielen Instanzen länger dauern kann als das Verändern einzelner Systeme mit Puppet oder Docker. Zusätzlich wird bei ständigem neu erzeugen von Instanzen das Management der IP-Adressen schnell unübersichtlich.

Literatur

[BOCH2010] : Bochynek, Diaz: Testen in der Finanzwelt, Diaz& Hilterscheid 2010

Verantwortlichkeiten, Zuständigkeiten und Kommunikation

21

Zusammenfassung

Eine gut funktionierende Zusammenarbeit der einzelnen Teammitglieder ist entscheidend für den Testerfolg. Die Kommunikation in Projekten wird mit steigender Teamgröße zunehmend komplex und stellt eine besondere Herausforderung für das Testmanagement dar. Die genaue Definition von Zuständigkeiten und der Einsatz einer Kommunikationsmatrix verhelfen zur strukturierten Planung des Informationsflusses innerhalb eines Projektes. Die RACI-Methode dient der Verbesserung der Kommunikation im Unternehmen.

21.1 Festlegung der Verantwortungsbereiche

Für die Verteilung der **Zuständigkeiten** auf die einzelnen Stakeholder und die Verteilung der **Verantwortlichkeiten** ist der Testprozess exakt festzulegen. Grundlage dafür ist, dass man die einzelnen Teststufen beschreibt, den Inhalt der einzelnen Arbeitspakete festlegt und die Übergänge zwischen den einzelnen Aktivitäten genau beschreibt.

Im Rahmen von Testaktivitäten sind z. B. Bereitstellung der Testumgebung, Definition der Testfälle, Automatisierung von Skripten und Durchführung der Testfälle einzelne Aktivitäten. Es empfiehlt sich, diese Schritte so fein granular wie möglich zu definieren. Jede einzelne Aktivität muss dabei grundsätzlich so beschrieben sein, dass sie auch von einem anderen als dem vorgesehenen Mitarbeiter durchgeführt werden kann. Bei sehr spezialisierten Tätigkeiten und Mitarbeitern mit langjähriger Erfahrung ist das erfahrungsgemäß problematisch, es führt aber kein Weg daran vorbei, das zu tun.

Unternehmen haben oftmals die Bedenken, Know-how aus dem Haus zu geben bzw. möchten selbst Know-how aufbauen. Das mag unschwerwiegend der Grund dafür sein, Arbeitsschritte nicht genau festzulegen. Fachliches Know-how haben aber immer nur einzelne

Mitarbeiter, nie eine Organisation als solche. Man bemerkt das besonders dann, wenn langjährige, hoch spezialisierte Mitarbeiter kündigen oder in Rente gehen.

Wenn das Zusammenspiel gut funktioniert, ist das Team gemeinsam erfolgreich. Doch viel zu oft sind die Zuständigkeiten völlig unklar. Man rennt von Pontius zu Pilatus und sucht nach der verantwortlichen Person. Das Chaos ist schließlich perfekt, wenn jemand mal krank, im Urlaub oder aus sonstigen Gründen nicht erreichbar ist. Die Folge sind Unklarheiten, unerledigte Aufgaben, Defekte sowie aufgeschobene Entscheidungen, die das Vorankommen mühsam machen. Daher ist es wichtig, in der Anfangsphase des Projektes die Zuständigkeiten im Unternehmen zu klären, um effizienter zu arbeiten. Das bedeutet auch, dass die Grundlagen wiederum im „generischen Testkonzept“ zu definieren sind, damit ein Projekt mit seinem Testkonzept darauf zugreifen kann. Oft sucht aber der Testmanager des Projektes genau diese Festlegungen vergeblich und ihm bleibt nichts Anderes übrig, als sie projektbezogen festzulegen. In vielen Fällen ist gar nicht von Anfang an transparent, was überhaupt alles zu erledigen ist.

21.2 Kommunikation im Projekt

Dazu kommt, dass die **Komplexität der Kommunikation** als zentrales Problem in größeren Projekten meist unterschätzt wird. Man sieht anhand folgender Formel deutlich, wie sehr die Komplexität steigt:

$$K = n(n-1)/2$$

wobei n die Anzahl der Teilnehmer bedeutet.

Das bedeutet, dass bei 5 Mitarbeitern 10, bei 10 Beteiligten bereits 45 und bei 20 Teilnehmern 190 Kommunikationsbeziehungen vorhanden sind. Ein größeres Projekt hat aber durchaus 50 bis 100 involvierte Stakeholder, also bei 50 Stakeholdern also rechnerisch 1225 (!) Kommunikationsbeziehungen, und auch wenn nicht jeder mit jedem verbunden ist, zeigt die Formel doch, wie viele Störeinflüsse möglich sind.

Typische Ursachen für Kommunikationsstörungen sind unterschiedliche Eigeninteressen, mangelhafte Informationen, unterschiedliche Wissensstände und Erfahrungen, innerbetriebliche Allianzen, Vorurteile oder historische Abneigung, die vielleicht gar nichts mit dem konkreten Projekt und dem konkreten Problem zu tun hat. Gerade in hektischen Phasen kann es dann zu Streitigkeiten oder gegenseitigen Schuldzuweisungen kommen, die am Ende dem Projekt und dem gemeinsamen Ziel noch weniger helfen.

Generell werden „weiche Einflussfaktoren“ für den Projekterfolg häufig unterschätzt. Für den Projekterfolg sind

- Methodik,
- Instrumente,
- Fachwissen,

- Erfahrung und
- Psychologie

wesentliche Einflussfaktoren.

21.3 Definition von Zuständigkeiten

Zuständigkeiten sind besonders dann wichtig, wenn es sich um allgemeine Tätigkeiten handelt. Ein Klassiker ist das Flipchart im Besprechungszimmer. Wer räumt es weg und sorgt dafür, dass nicht interne Informationen hängen bleiben? Angenommen, dort ist eine Kalkulation mit dem Hinweis „20 Prozent Gewinnzuschlag“ oder ein Hinweis „massive Qualitätsmängel in der Anwendung“ notiert. Wenn die Zuständigkeit für die Vorbereitung des Sitzungsraumes für das nächste Kundengespräch unklar ist und der Kunde für den Abnahmetermin schon in der Tür steht und das Flipchart sieht, dann hat er ungewollt eine Steilvorlage für ein Thema, das er dringend besprechen möchte.

Daher sollte eine Liste mit den einzelnen Personen, mit Namen, Kontaktdaten und am besten noch mit Bild, den Zuständigkeiten im Projekt und für jeden dieser Personen ein oder mehrere Vertreter, im Testkonzept mit aufgenommen werden.

Es ist auch möglich, im Testkonzept nur die Namen zu hinterlegen und die Liste der Verantwortlichkeiten unternehmensweit zu pflegen und zu verlinken. Das hat den Vorteil, dass bei Umorganisationen die Abteilungsbezeichnung zentral an einem Ort gepflegt wird oder auch Änderungen der Telefonnummern oder des Büros bei Umzügen nicht vom Testmanager gepflegt werden müssen. Dann muss aber die Beschreibung der Tätigkeiten dort ebenfalls so fein granular hinterlegt sein, dass sie auf den ersten Blick hervorgeht. Erfahrungsgemäß ist es immer wieder schwierig und eine Fehlerquelle, wenn man verschiedene Informationsquellen mit unterschiedlichen Daten zur Verfügung hat.

Für die konkreten Abläufe sind Ablaufbeschreibungen und entsprechende Checklisten für den Vertretungsfall zu erstellen.

So kann sichergestellt werden, dass die entsprechenden Aufgaben erledigt werden und keine Wartezeiten entstehen. Es ist nicht nur sinnvoll, die Tätigkeiten und Zuständigkeiten aufzulisten und zu beschreiben. Es kann auch nützlich sein, diese zeitlich zu ordnen.

Weiterhin sollte geprüft werden, ob es Tätigkeiten gibt, die in bestimmten Intervallen zu erledigen sind, z. B.

- wöchentlich,
- monatlich,
- quartalsweise,
- halbjährlich,
- jährlich,
- sporadisch,
- pro Release oder
- pro Teststufe.

Bei der Vertretung ist nachzufragen, ob sie die Einzelbeschreibung der Tätigkeiten verstanden hat und nachvollziehbar findet. Ideal ist es, einmal zusammen mit der Vertretung gemeinsam ihre Aufgaben Schritt für Schritt anhand der Beschreibung zu erledigen.

Die Klärung der Zuständigkeit ist auch für die Einarbeitung neuer Mitarbeiter wichtig. Neue Mitarbeiter sollten so schnell wie möglich produktiv werden können. Anhand der Tabelle mit den Zuständigkeiten kann für neue Mitarbeiter ein Plan für die ersten Tage erstellt werden und dabei geklärt werden, mit wem der neue Mitarbeiter sprechen muss und welches die Themen sind, die er dort lernen soll. Die entsprechenden verantwortlichen Mitarbeiter wissen am besten, was ihre Zuständigkeit ist und können diese am besten selbst erläutern und Fragen beantworten. Dadurch lernt der neue Mitarbeiter seine Kollegen und gleichzeitig die Arbeitsweise des Unternehmens kennen. Den neuen Mitarbeiter sollte man während der Einarbeitungsphase in den nächsten Wochen gezielt fragen, wo ihm die Zuständigkeiten noch nicht klar sind. Dadurch kann man die Tabelle weiter optimieren. Denn mindestens so wichtig wie die Dokumentation der Zuständigkeiten ist die Überwachung der Aktualität [BUKA2019].

Vor allem bei den Übergängen zwischen einzelnen Bereichen ist die Verantwortlichkeit festzulegen. Wenn im Rahmen einer Testverarbeitung mehrere manuelle Schritte von Mitarbeiter 1, 2 und 3 durchzuführen sind, dann muss z. B. Mitarbeiter 2 von Mitarbeiter 1 informiert werden, wenn er mit seinen Arbeiten fortfahren kann. Wenn nicht bekannt ist, wer dann wen wann informiert, kann ein Prozess schnell ins Stocken geraten. Der Testmanager als übergeordnete Instanz sollte über den Status der einzelnen Schritte daher immer informiert sein, um im Notfall eingreifen zu können. Vor allem in größeren Projekten ist dazu Kommunikation über die Grenzen der eigenen Abteilung oder des eigenen Unternehmensbereichs erforderlich, an der es häufig hapert.

Die Ursachen für teure Softwarefehler, extreme Projektverzögerungen oder große Mehrkosten sind in der Regel nicht technische Herausforderungen, sondern fehlende, fehlgeleitete oder unklare Kommunikation.

Es empfiehlt sich daher, zu Beginn des Projektes eine Kommunikationsmatrix zu erstellen:

21.4 Kommunikationsmatrix

Jeder Prozess und jedes Arbeitspaket innerhalb eines Projektes sollen einen festen Platz im Informationsaustausch haben. Je reibungsloser, zielgerichteter und störungsfreier die Kommunikation abläuft, desto erfolgreicher verlaufen Prozesse und zu erledigende Arbeiten. Das gilt nicht nur für die teaminterne Arbeit, sondern auch für die Kommunikation über verschiedene Schnittstellen wie zum Beispiel abteilungsübergreifendes Arbeiten.

Beim Versand und beim Empfang von Information setzt jedes Individuum andere Prioritäten. Kommunikationssender und Empfänger müssen selbe Information anders strukturiert aufarbeiten und über verschiedene Kanäle versenden. Weiterhin gilt es, um eine In-

formationsflut zu vermeiden, nur die wirklich relevanten Informationen zu kommunizieren. Für die Darstellung dieser Abläufe ist eine Kommunikationsmatrix sinnvoll.

Innerhalb der Kommunikationsmatrix wird nicht nur definiert, wer wann über welchen Kanal kommunizieren soll, sondern auch, welche bewussten Inhalte kommuniziert werden. Dabei können Detaillierungsgrade und Schwerpunkte auf den jeweiligen Sender und Empfänger zugeschnitten werden.

Die dabei entstehenden Vorteile des geregelten Austausches von Informationen und der Optimierung des Informations- und Kommunikationsflusses schaffen Transparenz und Klarheit im Projekt.

Als „Kommunikation“ wird allgemein der Austausch oder die Übertragung von Erkenntnissen, Erfahrungen, Wissen und Informationen definiert. Eine Kommunikationsmatrix (Abb. 21.1) bietet die Möglichkeit festzulegen, welcher Beteiligte in welchen Projektsituationen mit welcher Maßnahme informiert wird. Sie ist damit ein gutes Hilfsmittel, um die Kommunikation im Projekt zu planen und auch in zeitlich straffen Projektsituationen keinen Beteiligten zu vergessen [KOMM2019].

Eindeutige Kommunikationswege mit möglichst wenigen Schnittstellen sind dabei zu bevorzugen.

Es muss auch klar sein, wer wem welche Anweisungen erteilen kann und wer für welchen Bereich die Verantwortung trägt.

Innerhalb des Projektmanagements gibt es primär drei **Kommunikationswege**, die beidseitige nicht zielgerichtete Kommunikation (Flurfunk, Tratsch und Klatsch, Small Talk), die beidseitig zielgerichtete Kommunikation (Diskussionen, die Beantwortung einer Anfrage oder eine Problemlösung) und die einseitig zielgerichtete Kommunikation (meist **Delegation**).

WANN ?	Zum Projektstart	An Meilenstein-terminen	Bei Verzögerungen	Bei individuellen Problemen	Zum Projektabschluss
WER ?					
Auftraggeber	Start Workshop		Persönliches Gespräch		Projektabschluss-präsentation
Projektleiter	Kick-Off				
Teilprojektleiter	Kick-Off				
Team	Kick-Off				
Stakeholder 1	Info per Mail				
Stakeholder 2					

Abb. 21.1 Kommunikationsmatrix

Die Kommunikation kann in drei verschiedenen Ebenen ablaufen:

1. **Sachebene:**

Auf der Sachebene werden überwiegend reine Informationen kommuniziert. Es werden über direkte Sprachmittel wie über Schriftstücke, Grafiken oder Worte Informationen über bestimmte Kanäle, wie E-Mails oder Telefonate übermittelt. Ein Beispiel einer reinen Sachinformation ist: „Herr Meyer kam heute 20 Minuten zu spät“.

2. **Emotionale Ebene:**

Innerhalb der emotionalen Ebene, wird zu der eigentlichen Sachinformationen ein Gefühl hinzumodelliert. Dies geschieht zum Beispiel unterbewusst durch Körpersprache, Mimik, Gestik oder Tonlage wie Tonfall. Weiterhin schwingen auf dieser Ebene Zu- oder Abneigung sowie Empathie oder Ort der Kommunikation mit und laden das Gesagte emotional auf. Das gleiche Beispiel könnte auf der emotionalen Ebene wie folgt lauten: „Herr Meyer kam heute schon wieder viel zu spät, da sollte endlich mal jemand eingreifen“!

3. **Strukturelle Ebene:**

Auf der dritten Ebene verändert die Zeit, der Ort, Hierarchie, formelle Aspekte oder Werte die gesagte Information. So wird beispielsweise mit dem Partner oder der Partnerin oder einem Freund über exakt dieselbe Information anders kommuniziert als mit dem Vorgesetzten in der Firma.

Unter dem Begriff **Kommunikationsplanung** versteht man die strukturierte Planung: des Informationsflusses innerhalb eines Projektes, der Kommunikation an Schnittstellen und der Kommunikation von und zum Projekt. Weiterhin gehört zur Kommunikationsplanung die Feststellung der Kommunikations- und Informationsbedürfnisse aller Stakeholder. Die **Kommunikationsmatrix** ist der Output der Kommunikationsplanung. Innerhalb dieser Matrix oder Tabelle stehen alle Kommunikationssender und Kommunikationsempfänger, welche für das Projekt interessant sind. Eine alternative Form einer Kommunikationsmatrix ist in Abb. 21.2 dargestellt:

21.5 Auswahl der Informationsempfänger

Bevor über Berichte, Inhalte und Kanäle gesprochen wird, ist es wichtig, sich bewusst zu machen, wer überhaupt welche Informationen erhalten soll. So wäre es zum Beispiel unsinnig, mehrere Mitarbeiter einer Abteilung einzeln zu informieren, wenn auch der jeweilige Abteilungsleiter im Projekt sitzt und die Informationen an sein Team verteilen könnte.

Grundlage der Entscheidung sollte eine fertige oder zu mindestens sehr weit fortgeschrittene **Stakeholderanalyse** sein, aus der hervorgeht, wer welche Rolle und Einstellung im Projekt hält bzw. vertritt.

WER berichtet	WEM wird berichtet	WOZU wird berichtet	WAS wird berichtet	WANN wird berichtet
Projektleiter	Projektleiter	Projektstatus	Fortschritt	An Meilensteinen
Mitarbeiter A	Stakeholder	Offene Fragen	Risikosituation	Wöchentlich
Mitarbeiter B	Geschäftsleitung	Informationsaustausch	Qualitätssituation	Monatlich
...	Controlling	Abstimmungen	Aufgaben	Quartalsweise
	Auftraggeber	Überblick verschaffen	Zahlen, Daten, Fakten	...
WIE wird berichtet	Lenkungsausschuss	...	Problemstellungen	
PDF via E-Mail	...		...	
Analog via Hauspost				
Präsentation/Keynote				
Meeting				
...				

Abb. 21.2 Kommunikationsmatrix [PRMA2019]

Es gilt zu entscheiden, wer Informationen erhalten soll. Auch die konkrete unbedingt intern zu behaltende Nichtinformation kann ein Instrument der Projektleitung darstellen.

Im Rahmen der Stakeholderanalyse können zum Beispiel definiert werden:

- Verantwortliche,
- Businesspartner,
- Führungskräfte,
- Lenkungsausschuss,
- Abteilungsleiter,
- Zulieferer,
- Drittanbieter,
- Subunternehmer,
- Kunden,
- Meinungsführer,
- Steuerkreise/Stäbe,
- Geschäftsführung/Vorstand oder
- Controlling.

Dabei sind folgende Gedanken im Vorfeld anzustellen:

- Welche Person/welche Gruppe ist auf welche Informationen angewiesen, um Arbeiten auszuführen/zu planen/zu koordinieren?
- Gibt es Personen/Gruppen die explizit keine Informationen (zu bestimmten Themen) erhalten sollen/dürfen?

21.6 Definition von Kommunikationszielen

Danach sind die **Kommunikationsziele** zu definieren: Dazu sind zunächst alle Stakeholder zu dokumentieren, die informiert werden sollen. Im zweiten Schritt ist das Ziel der Kommunikation zu definieren. Dabei ist zu bedenken, warum diese Person oder Gruppe Informationen benötigt. Ohne diese Kommunikationsziele (was wird durch die Informierung bezweckt) wird zwar kommuniziert, aber nichts erreicht. Manchmal bekommt man E-Mails, bei denen man sich denkt: „Und was soll ich jetzt damit“? Dann lag entweder kein oder ein falsch definiertes Kommunikationsziel vor.

Das Kommunikationsziel für einen Abteilungsleiter einer Fachabteilung könnte beispielsweise sein: „Übermittlung der geplanten Ressourceneinsätze für die kommenden zwei Wochen.“

21.7 Festlegung von Informationsinhalten

Im dritten Schritt der Erstellung der Kommunikationsmatrix wird die Tabelle um die konkreten, eigentlichen Sachinformationen erwartet, die der Empfänger erwartet. Dazu gehören mindestens:

- aktueller Projektstatus,
- Meilensteintrendanalyse,
- aktuelle SOLL-/IST-Situation der Aufwände und Kosten,
- aktuelle SOLL-/IST-Situation der Fehler,
- Probleme in der derzeitigen Projektphase und
- Entscheidungsbedarf.

Dieser Schritt sollte so genau wie möglich beschrieben werden. Das hilft später nicht nur bei der Ausführung, sondern auch bei der Automatisierung der Informationsversendung.

Danach ist der Kommunikationskanal zu definieren. Das sind in einem Testprojekt in der Regel E-Mails mit entsprechenden Reports, es können aber auch Meetings, Präsentationen oder Newsletter sein.

Der Testmanager ist für die funktionierende Kommunikation verantwortlich, muss also als Sender die definierten Informationen zeitgerecht an die festgelegten Empfänger liefern. Der Testmanager ist aber auch Empfänger und auf Zulieferungen, Rückmeldungen und Status-Updates angewiesen.

21.8 Vorteile der Kommunikationsmatrix

Eine Kommunikationsmatrix hat folgende Vorteile:

- **Zeitersparnis:**

Gerade in größeren Projekten oder Projekten mit vielen Stakeholdern bekommt man als Testmanager fast täglich E-Mails, welche nach einem Status fragen. Selbst wenn man für die Beantwortung jeder Mail nur 10 Minuten brauchen (inklusive suchen der gewünschten Information), summiert sich das schnell auf. Daher sollte man auf bestehende und kommende Berichte verweisen bzw. den Standardbericht der letzten Woche versenden, und, situativ abhängig, dabei konsequent bleiben – so kann man sich einiges an Zeit sparen.

- **Die Kommunikation bricht nicht ab:**

Durch den geregelten und reglementierten Austausch von Informationen bricht der Informationsfluss, egal aus und in welche Richtung, nicht ab. Mit dem aktiven Einfordern und dem Weiterversand von Informationen bleibt das Projekt im Fluss.

- **Alle auf einem Stand:**

Es ist nervig, fehleranfällig und zeitaufwändig, als viele Leute mit unterschiedlichem Wissen in einem Meeting zusammensitzen zu haben. Wenn jeder erst informiert werden muss, ist gleich zu Beginn die Luft raus. Mit einer ordentlichen Informationsbasis sind jedoch alle Mitarbeiter auf demselben Stand und es kann sofort mit dem Thema begonnen werden.

- **Funktioniert auch im Vertretungsfall:**

Jeder ist mal unpässlich, wegen Krankheit, Weiterbildung, Urlaub oder durch einen Termin außer Haus. Der Empfänger der Botschaft muss diesen Umstand aber gar nicht bemerken, mit einem durchdachten Plan kann auch der Assistent oder die Vertretung alle benötigten Informationen versenden, ohne zusätzliche Telefonate oder extra Entschuldigungen liefern zu müssen.

- **Beziehungen zu Stakeholdern werden sich verbessern:**

Durch den stetigen Informationsfluss fühlen sich alle Beteiligten ernst genommen und gut informiert. Dieses Gefühl wird den Testmanager zum Projektleiter aufwerten, er wird dadurch auch bei Problemen respektiert. Weiterhin verbessert sich durch die stetigen Kontakte auch die zwischenmenschliche Beziehung zu den jeweiligen Stakeholdern.

21.9 Anforderungen an die Projektkommunikation

Folgende Grundsätze haben sich für Kommunikation im Projekt bewährt:

- **Klar und eindeutig:**

Es sollten ausschließlich Sachinformationen eingesetzt und diese klar und präzise auf den Punkt gebracht werden.

- **Effizient:**

Die Planung muss so gestaltet werden, dass alle Informationsbedarfe mit möglichst wenig Aufwand bereitgestellt werden können. Das wird leider dann schwierig, wenn in einem größeren Projekt unterschiedliche Tools eingesetzt werden oder Tools neu eingeführt werden. Jeder Toolbruch ist ein massives Projektrisiko und sollte auch als solches gekennzeichnet werden.

- **Schriftlich:**

Die Aussagen sind schriftlich zu dokumentieren. Jedes Meeting muss protokolliert werden, und es ist – wenn nicht schon von Anfang an geklärt – der Protokollführer zu Beginn des Meetings zu bestimmen. Es ist auch von hoher Wichtigkeit, das Protokoll direkt nach dem Meeting an alle Teilnehmer (ggf. im Entwurfsstadium) zu versenden und nicht tagelang damit zu warten. Wenn der Versand des Protokolls nicht unmittelbar geschieht, ist das Meeting bereits in Vergessenheit geraten und manches, was gesagt wurde, bereits überholt, sodass das Protokoll dann bereits veraltet wirkt. Es empfiehlt sich, das Protokoll während des Meetings auf einem Beamer oder online mit allen Beteiligten schon während der Erstellung direkt zu teilen. Die Protokollierung von Aussagen und Meetings hat eine hohe Priorität und wird leider oft vernachlässigt.

21.10 RACI-Methode

Die RACI-Methode dient zur Verbesserung der Kommunikation im Unternehmen.

Der Begriff RACI setzt sich aus den Anfangsbuchstaben der englischen Wörter **responsible, accountable, consulted** und **informed** zusammen.

Diese Begriffe spielen bei der Anwendung der RACI-Methode eine tragende Rolle. Die RACI-Methode eignet sich besonders, um innerhalb einer Organisation die Verantwortungen und Aktivitäten bestimmter Stakeholder klar zu definieren und zuzuordnen. Oberstes Ziel der RACI-Methode ist es, durch die übersichtliche Darstellung der Verantwortlichkeiten eine strukturierte Arbeitsweise zu erzielen und die Kommunikation im Projekt zu optimieren.

Die RACI-Methode gliedert sich in vier grundlegende Zuständigkeiten:

- „Responsible“ ordnet die Durchführung einer bestimmten Tätigkeit einer ausgewählten Person zu. Diese Person wird als verantwortlich im disziplinarischen Sinne verstanden.
- „Accountable“ beschreibt die Person, die im rechtlichen oder kaufmännischen Sinn die Verantwortung trägt. Ihr wird beispielsweise die rechtliche, politische oder auch wirtschaftliche Verantwortung der Tätigkeit übertragen.

RACI	[1] Vorbereitung				[2] Back Office															
	1.1	1.2	1.3	1.4	2.1	2.2	2.2.1	2.2.2	2.2.3	2.2.4	2.2.5	2.2.6	2.2.7	2.2.8	2.2.9	2.3	2.4			
Projektleiter																				
Name des PL	A	A	A	A	A		A	A	I	A		A	I	A	I	A	A			
Stakeholder																				
Stakeholder 1					I		I	I	A	I			A							
Stakeholder 2																				
Stakeholder 3																				
Mitarbeiter																				
Mitarbeiter 1	R	R	R	I	C		C				R			R		C	C			
Mitarbeiter 2	C	C	C	I	R		R	R	R		A	R	R	C	A	R	R			

R Responsible/ Verantwortlich

A Accountable/ Auftraggeber, Empfänger

C Consulted/ Unterstützend, Beratend

I Informed/ zu informieren

Abb. 21.3 Die RACI-Methode

- „Consulted“ bedeutet für eine Person, die nicht direkt in die Umsetzung der Tätigkeit integriert ist, aber relevante Informationen für alle Beteiligten besitzt, dass sie für den weiteren Arbeitsablauf als Fachmann hinzugezogen werden kann“.
- „Informed“ beschreibt schließlich die Kompetenz einer Person, Fortschritte und Ergebnisse im Arbeitsablauf festzuhalten und weiterzugeben.

Die klare Abgrenzung dieser Tätigkeiten stellt vorrangig sicher, dass sich alle Beteiligten auf die ihnen zugeordnete Aufgabe konzentrieren.

Zusätzlich zur klassischen RACI-Methode (Abb. 21.3) werden noch drei weitere Abwandlungen unterschieden. Es handelt sich hierbei um Varianten, die individuell an die zu erledigende Tätigkeit angepasst werden können.

RASCI beschreibt den Zusatzschritt Supportive, der eine ergänzende Arbeitskraft zur Unterstützung vorsieht. RASCI findet beispielsweise bei umfangreicheren Aufgaben Anwendung.

VARISC setzt zwei zusätzlich involvierte Personen voraus. Durch Überprüfung anhand vorgegebener Kriterien wird durch eine Person die Sicherstellung der bisher geleisteten Arbeit gewährleistet. Im Nachgang wird eine weitere Person bestimmt, die das Arbeitsergebnis bestätigt und schlussendlich freigibt.

RACIO, welches das englische Wort „omitted“, zu Deutsch „auslassen“, einschließt, beschreibt den Ausschluss einer bestimmten Person am Projekt. Hier kann es hilfreich sein, bereits im Vorfeld festzulegen, welche Personen bewusst nicht in eine Aktivität mit eingebunden werden soll.

21.11 Häufige Fehlerquellen

Folgende Fehlerquellen sind besonders häufig anzutreffen:

- **Abkürzungen nutzen und diese nicht erklären:**

Wenn Abkürzungen nicht klar sind, müssen sie definiert und erklärt werden. Der Ersteller selbst weiß, dass in seiner Kommunikationsplanung die Kürzel „MST, KÜ“ für „Meilensteintrendanalyse und Kostenübersicht“ stehen, seine Vertretung im Krankheitsfall o. Ä. jedoch nicht. Wenn Abkürzungen verwendet werden, dann welche, die ganzheitlich im Unternehmen bekannt oder mindestens auf jeder Seite in der Fußnote erklärt/ausgeschrieben sind.

- **Falsches Medium wählen:**

Mit dem Gegenüber muss über favorisierte Medien und Dateitypen gesprochen werden. Somit wartet der Empfänger nicht vergeblich auf eine E-Mail, wenn die Informationen bereits schriftlich per Post verschickt wurden.

- **Es jedem recht machen wollen:**

Im Projekt sind zum Beispiel 10 Stakeholder, jeder hat andere Präferenzen zur gewünschten Empfangsart der Informationen sowie zur Informationsdichte. Der Projektleiter bzw. bei einem Testprojekt der Testmanager entscheidet über Art und Umfang der Information, legt den grundlegenden Tenor fest und muss sich auch daran halten.

- **Zeiträume zu eng setzen:**

Ihr Projekt hat eine Laufzeit von mehr als 12 Monaten, die Zeitspanne der Lieferungen steht allerdings auf zweiwöchig – es passiert dann garantiert öfters, dass Arbeitspakete länger brauchen und ein Fortschritt auf dem Zeitstrahl kaum sichtbar wird. Daher sind von Anfang an Berichte zu erstellen, die Fortschritte ausweisen, nicht dass es so aussieht, als hätte sich gar nichts getan. Ein weiterer Nachteil kann sein, dass es zeitlich gar nicht zu schaffen ist, die Berichte überhaupt zu erstellen, sollte es keine automatische Exportfunktion einer übergreifenden Projektmanagement-Software geben.

- **Gefahr der Überplanung ignorieren:**

Die Skalierung des Planes und der Umfang des Teams sind wichtig. Gerade in kleinen Unternehmen oder Agenturen läuft vieles informell – und es läuft gut. In Konzernen gelten häufig völlig andere Vorgaben, ob intern oder wirkende Gesetzgebungen. Der eigene Plan und das eigene Testkonzept müssen gegenüber der Planung skaliert werden. Man sollte nicht mehr beschreiben, als nötig ist.

Literatur

[BUKA2019] : <https://www.buero-kaizen.de/mit-zustaendigkeiten-zu-mehr-effizienz/> zugegriffen am 09.05.2020

[PRMA2019] : <http://projektmanagement-manufaktur.de/kommunikationsmatrix>, zugegriffen am 09.05.2020

Zusammenfassung

Die einzelnen Stakeholder im Testprojekt sind zu definieren. In diesem Zusammenhang sind auch die nötige Einarbeitung und Ausbildung für neue Mitarbeiter zu beachten. Ein gutes Konfliktmanagement und Beziehungsmanagement beeinflusst maßgeblich den Projekterfolg.

22.1 Testpersonal

Im Rahmen des Testkonzeptes muss beschrieben werden, welche Stakeholder für die einzelnen Aktivitäten im Projekt (z. B. Erstellung der Testspezifikationen, manuelle Durchführung von Testfällen, Testumgebungsmanagement, Testautomatisierung) in welchem Umfang vorgesehen sind.

Die Planung der Aktivitäten ist dabei mindestens auf die Kalenderwoche herunterzubrechen, besser noch tagesgenau vorzunehmen. Eine Planung auf längere Zeiträume ist zu ungenau, um bei Verzögerungen rechtzeitig eingreifen zu können.

Das bestehende **Testpersonal** ist normalerweise (zumindest weitestgehend) schon gegeben. Wenn es Engpässe gibt oder spezielle Skills im Projekt zusätzlich benötigt werden, muss die Aufgabe präzise in einer Stellenausschreibung beschrieben werden.

Dabei ist auch zu beachten, dass jeder neue Tester eine bestimmte Phase für die Einarbeitung benötigt, in der er noch nicht 100 % produktiv sein kann. Das liegt allein schon am Onboarding-Prozess, der sich vor allem in größeren Unternehmen manchmal über Wochen hinzieht und teilweise sehr zäh und langsam abläuft. Der Arbeitsplatz muss eingerichtet sein, die Berechtigungen müssen vorliegen, und der neue Mitarbeiter muss die Anforderungen des Projektes verstehen. Zusätzlich muss der neue Mitarbeiter die Organisation, die anderen Stakeholder und die speziellen Herausforderungen seines neuen

Arbeitsumfeldes kennenlernen. Erfahrungsgemäß dauert das einige Wochen, und wenn Projekte nur sehr kurz angesetzt sind, muss man dabei bewusst Abstriche vornehmen. Dazu kommt, dass bestehende Mitarbeiter, die schon jahrelang in der Organisation beschäftigt sind, viele Dinge als bekannt voraussetzen und sich nicht in die Probleme des Neuankömmlings hineinversetzen können. Wesentliche Informationen oder Zusammenhänge fehlen und viele Details kann man in der Unternehmenskommunikation nur zwischen den Zeilen herauslesen. Die vorhandenen Dokumentationen bietet dafür meist nicht hinreichend Orientierung. Eine gute Idee ist es zum Beispiel, wenn man gerade neue Mitarbeiter zusätzlich zur Protokollerstellung in einem Meeting heranzieht. Die ersten Protokolle weisen wahrscheinlich viele Fehler und Lücken auf, dem neuen Mitarbeiter fehlt der „rote Faden“ und wesentliche Hintergrundinformationen, aber mit einer nachträglichen Besprechung des Protokolls vor der Veröffentlichung und einem Review in der Runde wird der Mitarbeiter schnell produktiv und erfährt am besten die Lücken.

Es gibt naturgemäß Unterschiede in Motivation, Ausbildung, Persönlichkeitsstruktur und Erfahrung der einzelnen Mitarbeiter.

Über die einzelnen **Persönlichkeitsmerkmale** gibt es in der Literatur zahlreiche Abhandlungen, von den Enneagrammtypen (1–9) bis zu einzelnen Tieren (Wal, Hai, Delphin, Eule) oder Farbtypen (rot, gelb, grün, blau). Projekte werden dann am ehesten erfolgreich, wenn man die einzelnen Stärken und Schwächen kennt und die Mitarbeiter entsprechend fördert. Das Thema Personalentwicklung und Konfliktmanagement ist außerhalb des Testmanagements zu behandeln, hat aber unmittelbare Auswirkungen auf den Projekterfolg. Leider ist immer wieder zu beobachten, dass Mitarbeiter auf Stellen sind, die nicht ihren Stärken entsprechen, sondern ganz im Gegenteil, ihre persönlichen Schwächen besonders gut sichtbar machen.

22.2 Konfliktmanagement

Konflikte im Projekt können erhebliche Anstrengungen erfordern. Wenn Störungen in der Teamentwicklung wahrgenommen werden, müssen die Ursachen für die beschriebenen Störungen im Team analysiert werden. Mögliche Störfaktoren sind:

- **Kompetenzprobleme:** Teammitglieder verfügen über einen höchst unterschiedlichen Kenntnisstand, etwa bezüglich der Fachkompetenz, sodass sich keine Synergieeffekte bei der Teamarbeit ergeben.
- **Entscheidungsprobleme:** Die Arbeit im Team geht nicht so recht voran, weil man sich nicht zu den notwendigen klaren Entscheidungen durchringen kann.
- **Informationsprobleme:** Probleme bei der Kooperation sind mitunter auf Schwierigkeiten beim Informationsaustausch unter den Beteiligten zurückzuführen.
- **Organisationsprobleme:** Ein vereinbarter Arbeitsplan, aus dem insbesondere ein Zeit- und Aktivitätenplan hervorgeht, fehlt. Dieser muss entsprechend nachgearbeitet werden.

- **Rollenkonflikte:** Rollen und Funktionen sind oft nicht eindeutig festgelegt. In diesem Fall muss noch einmal über das Rollenkonzept diskutiert und eine klare Festlegung der Verantwortung und Kompetenzen von Leitung und Team getroffen werden.
- **Beziehungsprobleme:** Es finden beispielsweise „Dominanzspiele“ der Teammitglieder untereinander statt. Alte Konflikte, vielleicht sogar historische Abneigungen aus früheren Projekten und vergangenen Zeiten, brechen auf. Werden Beziehungsprobleme zwischen Teammitgliedern ignoriert, kann dies zu echten Krisen führen, wo auch sachliche angemessene Informationen und Argumente nicht mehr gehört und berücksichtigt werden.

Geeignete Maßnahmen zur Bewältigung der genannten Störungen bestehen primär in einem optimalen **Beziehungsmanagement** gegenüber allen Kooperationspartnern sowie in einem sorgfältig abgestimmten **Informationsaustausch**. Insofern liegt es nahe, dass – je nach Kooperationspartner – auch ein angepasstes und differenziertes Kooperationsverhalten notwendig ist:

- Die IT-Leitung sollte versuchen, die Merkmale von Top-Teams vorzuleben und das Umfeld im Team in regelmäßigen Abständen zu motivieren. Kundenorientierung sollte der wesentliche Leitsatz des Handelns im Team sein.
- Ein weiterer Erfolgsfaktor ist die Einstellung der einzelnen Teammitglieder. Positiv ist es, wenn jedes Teammitglied versucht, eine gemeinsame Wellenlänge mit dem Kooperationspartner zu erreichen. Da die einzelnen Kooperationspartner naturgemäß höchst unterschiedlich sein können, ist dies nicht immer einfach. Teammitglieder benötigen in jedem Fall ein großes Repertoire an Verhaltensalternativen.
- Zur gemeinsamen Erreichung der Ziele ist außerdem immer wieder zu prüfen, ob die inhaltlichen Arbeitsschritte und Teillösungen dem Teamziel entsprechen. Daraufhin ist zu überlegen, ob die Art der Zusammenarbeit (Rollen- und Aufgabenverteilung, Teamklima, Machtverteilung) und die Art des Vorgehens (Methodik) zielführend sind.
- Außerdem sollte jedes Teammitglied die Wirkung, die es beim jeweiligen Gegenüber erzeugt, zur Kenntnis nehmen und als Spiegelung des eigenen Verhaltens betrachten. Dies erfordert mitunter den Mut, eingefahrene Gleise zu verlassen und die Bereitschaft, neue Verhaltensweisen zu erlernen. Solche Anstrengungen werden aber sicher belohnt, weil sich jedes Mal eine neue Gelegenheit zur Persönlichkeitsentwicklung dabei bietet.

Eine wichtige Aufgabe für Projektleitung und Teammanagement ist es daher, die Teamentwicklung hinsichtlich der zu erreichenden Ziele zu steuern. Weil die Teamarbeit zeitlich begrenzt ist, muss schnell eine funktionierende Arbeitsbasis geschaffen werden. Der IT-Leitung obliegt es, dazu in moderierender Weise das Team zu formen, Teamregeln zu vereinbaren sowie die Verantwortlichkeiten zu regeln.

Projekte scheitern in letzter Konsequenz nicht an technischen Herausforderungen oder komplexer Software, sondern an der Kommunikation, an persönlichen Differenzen, unterschiedlichen Vorstellungen, persönlichen Ambitionen, gegenseitigen Blockaden, Neid. Manchmal scheitern sie auch an der Ausblendung der Realität und unrealistischen Zielen.

Zusammenfassung

Das COCOMO-Modell ist für die Schätzung der Testdauer geeignet. Für eine detaillierte Kostenschätzung und Terminplanung sind genaue Daten erforderlich. Für die Terminplanung sind die einzelnen Testphasen und gegenseitige Abhängigkeiten zu betrachten. Die ermittelten Kosten und Termine zusammen mit den Personalauslastungsplänen müssen während der Projektlaufzeit permanent überprüft werden. Verzögerungen auf dem „kritischen Pfad“ können zu einer Verzögerung des gesamten Projektes führen. Mithilfe des COCOMO-Modells kann auch die Testproduktivität ermittelt werden.

23.1 Ermittlung der Testdauer

Im Rahmen der Testplanung soll ein genauer Plan erstellt werden, wann welche Arbeiten von wem durchzuführen sind.

Für die Schätzung der Testdauer bietet sich das **COCOMO-Modell** an.

Nach diesem Modell besteht ein algorithmisches Verhältnis zwischen dem Aufwand für ein Projekt und der Projektdauer. Die Gleichung für die Dauer einer Softwareentwicklung ist in COCOMO-I:

$$\text{TDEV} = 2,5 \times (\text{Aufwand})^{\text{Exponent}}$$

wobei der Exponent folgendermaßen gewählt wird:

- 0,38 für Standalone-Systeme,
- 0,35 für verteilte System und
- 0,32 für Embedded-Realtime-Systeme.

Nach der ursprünglichen COCOMO-I-Formel wäre die Dauer eines Testprojektes für verteilte Systeme mit dem geschätzten Aufwand von 275 Personentagen bzw. 14 Personenmonaten:

$$\text{TDEV} = 2,5 \times (14)^{0,35} = 6,3 \text{ Monate}$$

Das wäre die minimale Dauer des Projektes mit zwei Testern. COCOMO-I geht jedoch von der Entwicklung als nur bedingt teilbare Aufgabe aus. Testprojekte sind aber wie Migrationsprojekte besser teilbar. Bis auf die Anfangsphasen der Analyse und Planung können Tester besser nebeneinander arbeiten. Damit lässt sich die Dauer des Testprojektes zwar nicht ganz, aber doch erheblich komprimieren. Daher ist die COCOMO-II-Zeitformel für den Test besser geeignet. Im neuen COCOMO-II-Modell lautet die Gleichung:

$$\text{TDEV} = (C \times (\text{PM})^F) \times (1 - (\text{SCED \%} / 100))$$

wobei $F = (D + 0,2 \times (E - B))$

In dieser Gleichung sind die Parameter wie folgt definiert:

- B = Untergrenze der Skalierungskomponente = 0,91,
- C = Multiplikator = 3,67 für Neuentwicklung,
- D = Zeitbasiskoeffizient = 0,28 für Neuentwicklung,
- E = Skalierungskomponente der Entwicklung = zwischen 0,91 und 1,23,
- F = Skalierungsexponent für die Projektdauer,
- PM = geschätzter Aufwand in Projektmonaten,
- SCED % = prozentuale Kompression des Projektes,
- TDEV = Zeit für die Entwicklung.

Danach berechnet sich die Dauer des Testprojektes wie folgt:

$$F = (0,28 + 0,2 \times (0,96 - 0,91)) = 0,29$$

$$\text{TDEV} = 3,67 \times (14)^{0,29} = 3,67 \times 2,15 = 7,89 \text{ Monate (gerundet 7,9 Monate)}$$

Das ist das errechnete Ergebnis für die Testdauer, wenn man, wie im früheren Projekt (► oben), zwei Tester einsetzt. Nun kommt aber der **Schedule-Compression-Faktor**

hinzu, der beim Testen relativ hoch ist. Die Zeit zum Testen kann in dieser Annahme durch die Verteilung der Arbeit auf mehrere Tester um mindestens 60 % verkürzt werden.

$$TDEV = 7,9 \times (1 - (60 / 100)) = 3,16 \text{ Monate (gerundet 3,2 Monate)}$$

Demnach könnten z. B. 5 Tester eingesetzt werden. Anforderungsanalyse und Testplanung werden von zwei Testern durchgeführt. Nach Beendigung der Planungsphase kommen drei weitere Tester hinzu, die zusammen mit den beiden ersten die Testfallspezifikationen erstellen, Testdaten generieren, die Testumgebung zur Verfügung stellen, den Test durchführen und die Testergebnisse auswerten. Dadurch könnte das gesamte Testprojekt nach ca. 3 Monaten bereits abgeschlossen sein [SNEE2012].

Inwiefern das aber in der Praxis machbar ist, hängt von der Rollenverteilung und der Projektorganisation ab. Auch die Verfügbarkeit der Tester für das Projekt, die generelle Verfügbarkeit (Urlaub, Krankheit) und eventuelle Verzögerungen aus dem Umfeld (Entwicklung, Umgebungsmanagement ...) können sich negativ auswirken. Außerdem sind Produktivität und Einsatzmöglichkeit der einzelnen Tester unterschiedlich.

Die obige Formel setzt außerdem voraus, dass die entsprechenden Daten im Unternehmen überhaupt konsequent erhoben werden. In vielen Projekten geht es nicht nur um Softwareentwicklung, sondern in erheblichem Maße um Konfiguration und Parametrierung der Systeme. All diese Einflüsse sind in der Berechnung der Aufwände angemessen zu berücksichtigen.

23.2 Detaillierte Arbeitsplanung

Zu empfehlen ist, den Aufwand pro Aufgabe und pro Mitarbeiter und Tag in einem **Arbeitsplan** (Tab. 23.1) zu planen.

Tab. 23.1 Arbeitsplan

Tätigkeit	Tester	Montag 15.06.	Dienstag 16.06.	...
Testfallerstellung	A	1	1	
	B	0	0	
Testdurchführung	A	0	0	
	B	0	0	
Testmanagement	C	1	1	
Tätigkeit	Tester	KW 20	KW 21	...
Testfallerstellung	A	5	...	
	B	4	...	
Testdurchführung	A	0	...	
	B			
Testmanagement	C	5	5	5

23.3 Kostenschätzung

Der Testmanager sollte jederzeit einen Überblick über die geplanten und bisher angefallenen Kosten haben. Dazu muss aber auch gewährleistet sein, dass er zeitnah einen Überblick über Mehraufwände bekommen kann. Wenn es tage- oder wochenlang dauern kann, bis die angefallenen Testaufwände für den Testmanager transparent werden (etwa, weil sie erst in einem zentralen Tool erfasst werden), besteht von Anfang an ein Kostenrisiko, das im Testkonzept benannt werden soll. Zu klären ist auch von Beginn an, ob der Testmanager eine Übersicht bzw. Verantwortung über Kosten zugewiesen bekommen soll.

Vor allem bei neuen Releases (z. B. einer Funktionserweiterung einer bestehenden Software) sind Regressionstests erforderlich. Je nach Umfang und Häufigkeit dieser Regressionstests kann dadurch eine Rekalkulation der Testkosten notwendig werden. Die Kostenentwicklung ist daher während der Projektlaufzeit laufend zu überprüfen.

23.4 Planung der einzelnen Testphasen

Die einzelnen Testphasen werden im Testkonzept zusammengefasst in einer Timeline dargestellt, um den Projektverlauf transparent darzustellen. Außerdem sollte eine Timeline für die einzelnen Testphasen (Abb. 23.1) dokumentiert werden:

Zu beachten ist dabei immer der „kritische Pfad“: Was passiert, wenn eine Phase später anfängt oder nicht rechtzeitig fertig wird?

Die Methode des kritischen Pfades verwendet die Darstellungsform des Vorgangspfeils. Bei **Netzplänen (CPM-Plänen)** werden die Vorgänge als Pfeile, die Ereignisse als Knoten und die Anordnungsbeziehungen wieder als Pfeile dargestellt. Voraussetzung, um

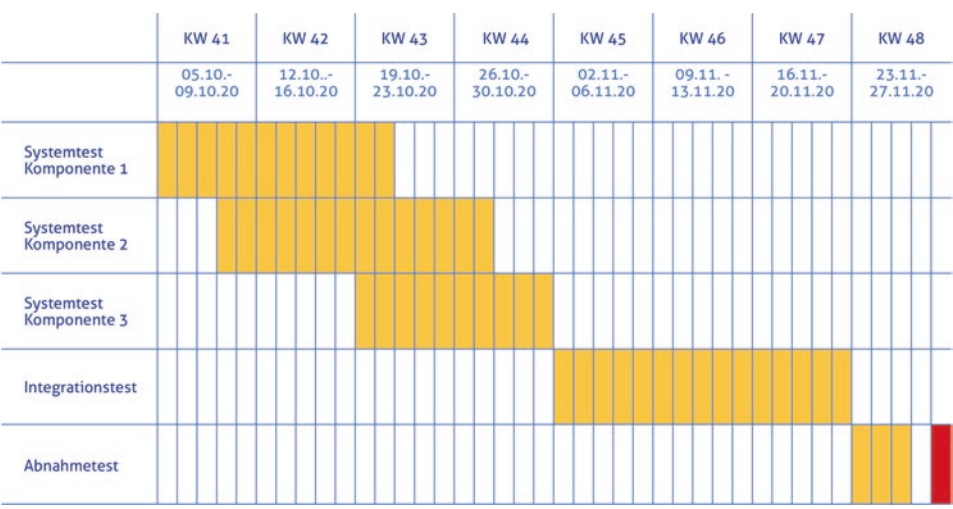


Abb. 23.1 Timeline der Testphasen

mit einem Netzplan sinnvoll arbeiten zu können, ist, dass alle Vorgänge des Projektes mit der jeweiligen individuellen Dauer richtig zueinander in Beziehung gesetzt werden.

Folgende Regeln sind dabei zu beachten:

1. Ein Vorgang kann erst beginnen, wenn alle vorhergehenden Vorgänge (Vorgänger) abgeschlossen sind. Dabei fällt das Anfangsereignis mit dem Endereignis des vorhergehenden Vorgangs zusammen (Ausnahme erster Vorgang).
2. Müssen mehrere Vorgänge beendet sein, bevor ein folgender Vorgang (Nachfolger) beginnen kann, so enden sie im Anfangsereignis des Nachfolgers.
3. Können mehrere Nachfolger beginnen, nachdem ein Vorgänger beendet ist, so beginnen sie im Endereignis des Vorgängers.
4. Haben zwei oder mehr Vorgänge gemeinsame Anfangs- und Endereignisse, so ist ihre eindeutige Kennzeichnung durch Einfügen von Scheinvorgängen sicherzustellen. Scheinvorgänge sollten als gestrichelter Pfeil, der besseren Übersichtlichkeit wegen, dargestellt werden.
5. Beginnen und enden in einem Ereignis mehrere Vorgänge, die nicht alle voneinander abhängig sind, muss die Eindeutigkeit ebenfalls durch Scheinvorgänge erreicht werden.
6. In einer Folge von Vorgängen können beliebig viele Scheinvorgänge eingefügt werden. Sie dienen der logischen Verknüpfung und können die Übersicht erhöhen. Scheinvorgänge sollten als notwendiges Übel angesehen und deshalb generell sparsam angewendet werden.
7. Kann ein Vorgang beginnen, bevor der Vorgänger vollständig abgeschlossen ist, so muss der Vorgänger unterteilt werden. Diese Regel erhöht nicht unbedingt die Klarheit des Netzplans.
8. Jeder Vorgang darf nur einmal ablaufen, es dürfen keine Schleifen auftreten.

Balkenplan: Für die Detailplanung von Projekten empfehlen sich der Balkenplan und der Personalauslastungsplan. Balkenpläne lassen sich z. B. mit Microsoft Project oder Excel erstellen. Abb. 23.2 zeigt eine grafische Darstellung und Abb. 23.3 eine Darstellung mit Excel.

Dabei empfiehlt sich eine Planung der einzelnen Aktivitäten und Verfügbarkeiten auf Tagesbasis. Eine Planung auf Wochenbasis ist in der Regel zu grob granular und stellt zu viele Leerlaufzeiten nicht ausreichend dar. Eine Planung auf Stundenbasis lässt sich im betrieblichen Alltag in der Regel nicht durchsetzen.

Alternative Darstellung:

Nachdem die logische Struktur des Projektes anhand eines Netzplans vorgenommen wurde, kann die eigentliche Terminplanung vorgenommen werden. Daraus ergeben sich Tätigkeiten, in denen der Test involviert ist.

Die Terminplanung geschieht normalerweise mit einem Balkenplan, wie er in Abb. 23.2 bzw. 23.3 dargestellt ist. Darin werden, ausgehend vom kritischen Pfad, die Intervalle mit Zeitdauer aller Vorgänge in einem geeigneten konsistenten Maßstab aufgetragen, wobei die vom Netzplan festgelegten Zusammenhänge beibehalten werden.

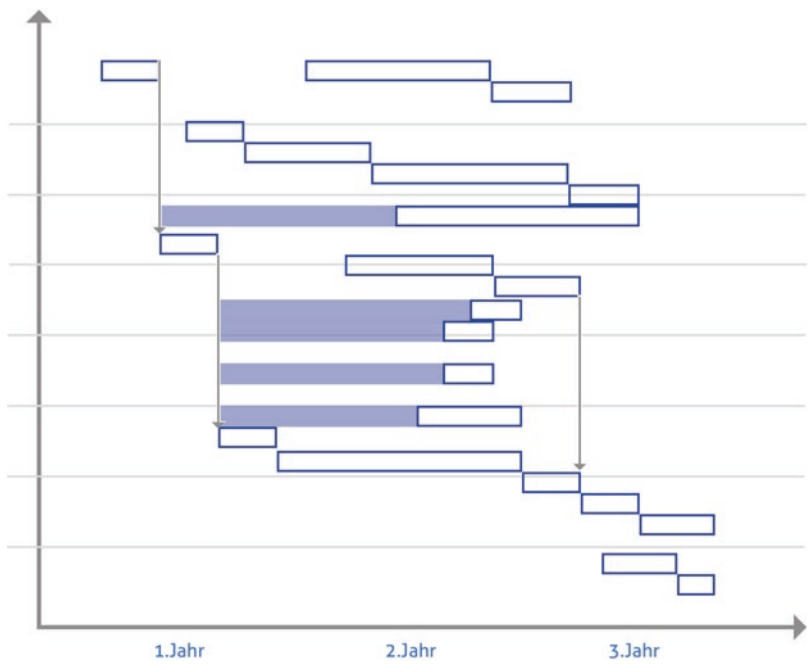


Abb. 23.2 Balkenplan

					31.08.-04.09.2020	07.-11.09.2020	14.-18.09.2020	21.-25.09.2020	28.09.-02.10.2020	05.-09.10.2020	12.-16.10.2020	19.-23.10.2020	26.-30.10.2020	02.-06.11.2020	09.-13.11.2020	16.-20.11.2020	23.-27.11.2020	30.11.-04.12.2020
Code	AP-Bezeichnung	Verant-wortung	Start	Ende	KW36	KW37	KW38	KW39	KW40	KW41	KW42	KW43	KW44	KW45	KW46	KW47	KW48	KW49
1	IT-Konzeption		31.08.2020	04.12.2020														
1.2	Ist-Analyse		14.09.2020	16.10.2020														
1.2.1	Ist-Analyse SW		14.09.2020	02.10.2020														
1.2.2	Ist-Analyse HW		21.09.2020	02.10.2020														
1.2.3	Ist-Analyse Infrastruktur		21.09.2020	16.10.2020														
1.2.4	Dokumentation Ist-Analyse		14.09.2020	16.10.2020														
1.3	Beschreibung Verfahren		19.10.2020	06.11.2020														
1.3.1	Definition Varianten		19.10.2020	21.10.2020														
1.3.2	Beschreibung Variante 1		22.10.2020	04.11.2020														
1.3.3	Beschreibung Variante 2		26.10.2020	06.11.2020														
1.3.4	Beschreibung Variante 3		28.10.2020	06.11.2020														
1.4	Marktforschung SW		09.11.2020	04.13.2020														
1.4.1	Erfassung Anwendungs-SW		09.11.2020	11.11.2020														
1.4.2	Analyse Anwendungs-SW		12.11.2020	20.11.2020														
1.4.3	Einholung Zusatzinfos		23.11.2020	27.11.2020														
1.4.4	Bewertung Anwendungs-SW		30.11.2020	04.12.2020														

Abb. 23.3 Balkenplan mit Excel [PROJ2019]

Aus dieser Darstellung sind die Projekttermine ablesbar, die für die einzelnen Vorgänge und Ereignisse gelten. Der Balkenplan kann aber auch sofort zur Personaleinsatzplanung verwendet werden. Das geschieht dadurch, dass man die im Netzplan vorgesehenen Soll-Stärken des Personaleinsatzes für die einzelnen Arbeitspakete den Balken zuordnet, die jetzt die Arbeitspakete repräsentieren, und die entsprechenden Werte in der vertikalen Achse addiert. Damit ergibt sich die „erste Näherung“ des Personalauslastungsplans im Projekt.

In der Praxis gibt es häufig Terminverschiebungen, die sich auf das gesamte Projekt auswirken. Daher muss auch die Terminplanung regelmäßig überprüft und ggf. angepasst werden. Wenn man bereits bei der Planung der einzelnen Testphasen ausreichend Pufferzeiten berücksichtigt, hat man weniger Terminverschiebungen während des Projektes und dadurch am Ende viel Aufwand und Aufregung erspart.

23.5 Personalauslastungsplan

Die erste Version des Personalauslastungsplans (Abb. 23.4) führt normalerweise sofort zu einer Umplanung des Projektes. Wenn nicht durch außerordentlich glückliche Umstände gleich beim ersten Mal ein einigermaßen ausgeglichenes „Manpower-Gebirge“ entstanden ist, wird kein Personalvorgesetzter akzeptieren, dass seine Mitarbeiter in kurzer Zeit in ein Projekt hinein- und wieder herausdelegiert werden. Abgesehen von den rein technischen

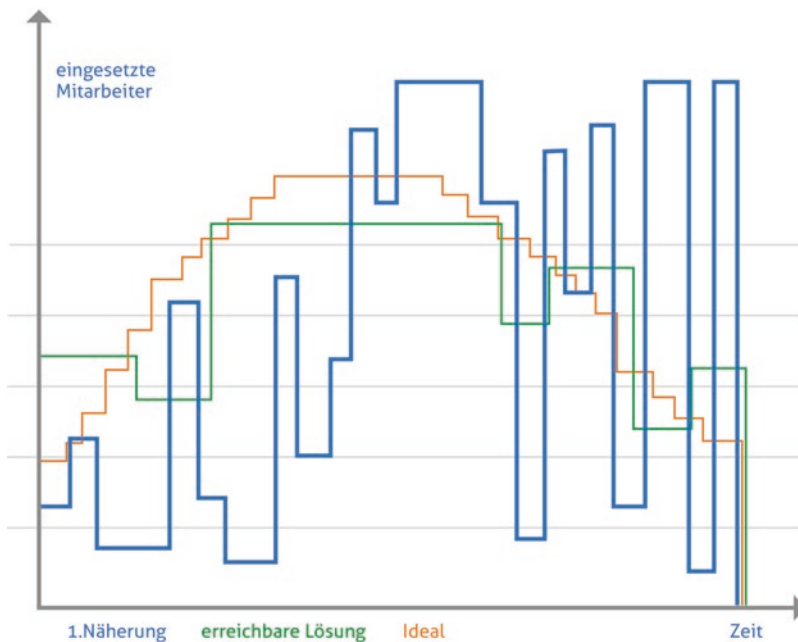


Abb. 23.4 Personalauslastungsplan

Reibungsverlusten, die dabei durch die in kurzen Abständen auftretenden Lernphasen bei Mitarbeitern auftreten, würde sich kein qualifizierter Mitarbeiter an einem solchen Projekt lange beteiligen.

Folglich muss wieder der Balkenplan diskutiert werden, was zur Überprüfung der Annahmen im Netzplan führt usw. Das Ganze ist also ein iterativer Prozess.

In vielen Fällen laufen im Unternehmen aber mehrere Projekte zeitlich parallel, die alle auf dieselben Ressourcen zugreifen. Für den Linienverantwortlichen, der die Tester auf einzelne Projekte aufteilt, ist es erforderlich, die Anforderungen aus den einzelnen Projekten zu erheben und dann die Tester entsprechend aufzuteilen. Bei einem Einsatz eines Testers für mehrere Projekte kann es immer wieder zu Abhängigkeiten und Leerlaufzeiten kommen, weil dann der Einsatz entweder auf dem kritischen Pfad liegt oder Verschiebungen es verhindern, dass ein Testobjekt zur Verfügung steht. Das ist besonders während der Testdurchführungsphase zu bedenken.

23.6 Detaillierte Testaufwandsplanung

Für eine detaillierte Testaufwandsplanung empfiehlt es sich, die Testaufwände pro Tag in einer Tabelle (► Tab. 23.2) zu planen.

Bei Verschiebungen einzelner Testphasen ist dieses Dokument anzupassen.

Pro Tag wird ermittelt, zu welchen Anteilen ein Mitarbeiter für das Projekt zur Verfügung steht. Dabei sind Feiertage, verteilte Standorte und Abhängigkeiten zwischen den Testern zu berücksichtigen. Der kritische Pfad ist zu beachten.

Es sollten für die Auslastung pro Arbeitstag nie Werte über 0,8 angesetzt werden, um genügend Puffer für unvorhergesehene Ereignisse zu haben und genügend Zyklen für erforderliche Nachtests. Auch unproduktive Zeiten, etwa für die laufende Aktualisierung der Testumgebung, sind dabei zu berücksichtigen.

Tab. 23.2 Arbeitsplan pro Mitarbeiter

		KW 29					...	Summe
		Mo 13.07.	Di 14.07.	Mi 15.07.	Do 16.07.	Fr 17.07.		
Tätigkeit	Mitarbeiter							
Testfallerstellung	Meier	0,8	0,8	0,8	0	0	...	
Testfallerstellung	Müller	0	0	0	0,5	0,5	...	
Testdurchführung, Analyse der Ergebnisse und Nachtest	Meier	0	0	0	0,8	0,8	...	
Testdurchführung, Analyse der Ergebnisse und Nachtest	Müller	0	0	0	0	0	...	
Last- und Performancetest	Schmidt	0	0	0	0,2	0,2	...	
Testkoordination	Huber	0,8	0,8	0,8	0,8	0,8	...	

Bei mehreren Testern in unterschiedlichen Teams empfiehlt sich, eine weitere Spalte für das Team zu führen, dem die einzelnen Mitarbeiter zugeordnet sind. Damit kann man darstellen, wann welcher Mitarbeiter ins Projekt eingebunden wird. Im Beispiel aus Tab. 23.2 wird Tester Müller erst später hinzugezogen und fängt in KW30 mit der Testdurchführung noch nicht an. Über die einzelnen Werte wird am Ende der Periode eine Summe pro Zeile und eine Gesamtsumme gebildet.

Sollten beim Test unterschiedliche Komponenten zu unterschiedlichen Zeiten zugeliefert werden, kann man das auch durch die Tabelle visualisieren.

Sollte sich das Projekt über mehrere Releases erstrecken und der Aufwand auf die einzelnen Releases zuzuordnen sein, so kann man das ebenfalls in der Tabelle abbilden und Summen für bestimmte Zeitintervalle bilden, aus denen der Testaufwand auf das einzelne Release heruntergebrochen werden kann. Der Anteil der Aufwände für die Regression wird in den späteren Releases steigen, vor allem dann, wenn die Testautomatisierung noch nicht ausreichend fortgeschritten ist. Aufwände der Testautomatisierung sind in dieser Tabelle ebenfalls zu erfassen.

Wie schwierig eine Testplanung zuweilen sein kann, zeigt folgendes Beispiel:

Es stehen 3 Tester und 3 Bausteine zum Test an.

- Tester 1 testet Baustein A in Woche 1;
- Tester 2 testet Baustein B in Woche 2;
- Tester 3 testet Baustein C in Woche 3.

C ist davon abhängig, dass Baustein A und B fertiggetestet sind.

Jeder Baustein soll 1 Woche zum Test benötigen. Die Tester haben die Tests spezifiziert und kennen das Projekt.

Anbei ein paar mögliche Störungen:

1. Tester 1 ist in Woche 1 krank, Tester 3 ist noch in einem anderen Projekt gebunden. Tester 2 kann Baustein A übernehmen.
2. Tester 1 ist in Woche 1 krank, Tester 2 und 3 sind noch in einem anderen Projekt gebunden. Tester 4 steht zur Verfügung, der jedoch das Projekt nicht kennt, die Testfälle nicht kennt und sich erst einarbeiten muss. Es kommt dadurch zu einer Verzögerung, die aber noch aufgefangen werden kann: Tester 4 testet in Woche 2 Baustein A einige Tage weiter, Tester 3 kann in Woche 3 rechtzeitig mit Baustein C beginnen.
3. Baustein B wird nicht rechtzeitig in der Entwicklung fertig. Tester 2 steht in Woche 2 zur Verfügung, kann aber mit der Testdurchführung am Montag nicht rechtzeitig beginnen, sondern erst am Mittwoch. Hiermit befindet sich das Projekt auf dem „kritischen Pfad“, weil der Test in Summe gefährdet ist. Der Test von Baustein C kann in der Woche 3 auch erst zeitlich verspätet beginnen. Bei Baustein C sollen deswegen noch Tester 1 und Tester 2 hinzugezogen werden, um rechtzeitig fertig zu werden. Tester 1 ist zu diesem Zeitpunkt aber bereits in einem anderen Projekt gebunden, das eine noch höhere Priorität als das betrachtete Projekt hat, und scheidet für die Verstärkung der Testaktivitäten also aus.

4. Baustein B hat eine derart schlechte Qualität, dass eine Teilintegration unmöglich wird und er keine hinreichende Voraussetzung zum Test von Baustein C hat. Die Nachbesserungsarbeiten in der Entwicklung dauern die gesamte Woche 3 an. Der Test von Baustein C ist sinnvoll erst in Woche 4 möglich, wobei der kritische Pfad verlassen wurde und es bereits eine Verzögerung des Projektes ergibt.

Die hier beschriebenen Beispiele erscheinen dabei noch relativ trivial. Gerade bei mehreren parallelen Projekten addieren sich diese Probleme. Bei gegenseitigen Abhängigkeiten wird die Projektplanung dabei häufig vor komplexe Herausforderungen gestellt. Daher empfiehlt es sich, schon bei der Personaleinsatzplanung bestimmte Leerlauf- und Pufferzeiten zu berücksichtigen. Jeder Mitarbeiter sollte in der Planung daher generell nur zu 60 % oder 70 % ausgelastet werden.

23.7 Schätzung der Testproduktivität mit der COCOMO-II-Methode

Das Verhältnis zwischen Aufwand und Zeit ist nicht linear, d. h., man kann ein Projekt von 12 Mannmonaten nicht auf 6 Mannmonate reduzieren, indem die Anzahl der Projektbeteiligten verdoppelt wird. Die steigende Anzahl an Beteiligten erhöht signifikant die Aufwände für Kommunikation, Administration und Organisation. Die Produktivität pro Mitarbeiter sinkt, teilweise sogar erheblich. Ein Schlüssel in der gegenseitigen Beeinflussung von Zeit und Aufwand ist die Ermittlung der Produktivität. In nachfolgender Skizze (Abb. 23.5) ist die Ermittlung der Testproduktivität grafisch dargestellt.

Zur Bildung einer Produktivitätskennzahl werden hier zuerst die sogenannten „**Testpoints**“ eingeführt. Sie erlauben es, jedes Testobjekt gemäß seiner Komplexität zu bewerten und ermöglichen so eine Differenzierung nach Schwierigkeiten einzelner Testtätigkeiten. Ein Testpoint wird dabei mit 1,5–3 Arbeitsstunden bewertet.

Man unterscheidet dabei zum einen die dynamischen Testpoints, also die Anzahl der ausgeführten logischen Testfälle gemäß den Anforderungen. Führt man beispielsweise 1000 Testfälle (1000 dynamische Testpoints) in 150 Arbeitstagen aus, so ergibt sich eine Testproduktivität von 6,6 Testpoints pro Tag.

Diese dynamische Produktivität kann wie in Abb. 23.5 aufgezeichnet und langfristig ermittelt werden. Zum andere existieren die statischen Testpoints. Diese ergeben sich aus der Anzahl der Testobjekte. Im Systemtest sind dies zum Beispiel die zu testenden Schnittstellen, Benutzeroberflächen oder Datenbanken. Die Formel zur Berechnung der Testpoints ergibt sich wie folgt

$$\text{Testpoints} = N_{\text{TestCases}} + (N_{\text{Panels}} \times 4) + (N_{\text{Reports}} \times 2) + N_{\text{Services}}$$

Da die Produktivität wesentlich von der Erfahrung der Tester und dem Grad der Testautomatisierung abhängt, muss die Statistik bisheriger Projekte (Abb. 23.5) bemüht wer-

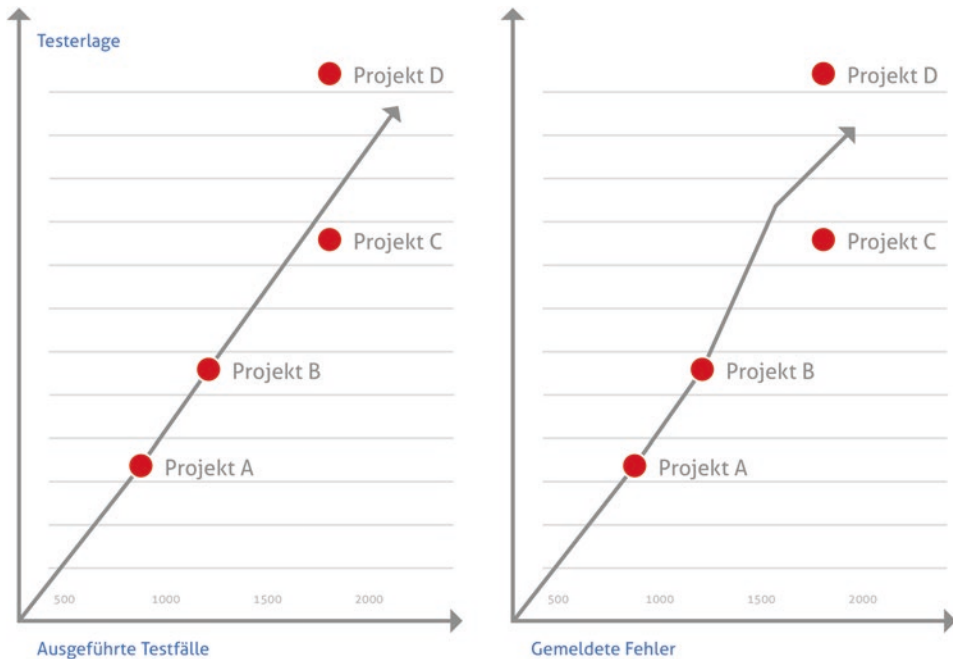


Abb. 23.5 Ermittlung der Testproduktivität

den. Hinzu müssen spezielle Einflussfaktoren und Testumstände berücksichtigt werden, sodass sich die **Testproduktivität** wie folgt ergibt:

$$\text{Testproduktivität} = (\text{Testpoints} / \text{Testertage}) \times \text{Einsatztage}$$

Die Idee dieser noch wenig detaillierten Formel wird in der sogenannten COCOMO-II-Gleichung zur Einschätzung des Testaufwands mit Größen neben der Testproduktivität weiter aufgegriffen.

Die drei weiteren berücksichtigten Faktoren sind hierbei:

- Projektbedingungen,
- Produktqualität und
- Systemtyp.

Die **Projektbedingungen** lassen sich wiederum in folgende fünf Bedingungen unterteilen:

- Grad der Testwiederholbarkeit,
- Stabilität der Testumgebung,
- Kenntnis der Anwendung,
- Zusammengehörigkeit des Testteams und
- Reife des Testprozesses.

Der finale Skalierungsexponent ist das arithmetische Mittel aus den Bewertungen der fünf Projektbedingungen. Jede dieser Bedingungen wird anhand nachfolgender Skala bewertet:

- sehr niedrig (1,23),
- niedrig (1,10),
- mittel gut (1,00),
- hoch (0,96) und
- sehr hoch (0,91) [IWTB2017].

Literatur

[SNEE2012] : Sneed, Baumgartner, Seidl: Der Systemtest, Carl Hanser Verlag München 2012

[PROJ2019] : <https://dieprojektmanager.com/terminplanung-balkenplan-gantt-diagramm-chart/>,
zugegriffen am 09.05.2020

Zusammenfassung

Bei Terminverschiebungen sind die Zielbeträge der Handlungsalternativen deutlich zu machen und mögliche alternative Entscheidungswege zu bewerten. Dazu eignen sich Methoden wie Investitionsrechnung und Argumentenbilanz. SWOT-Analysen, Konsequenzentabellen und Entscheidungsbäume, Nutzwert- und Risikoanalysen sind weitere hilfreiche Methoden zur Bewertung und Steuerung von unvorhergesehenen Ereignissen im Laufe von Projekten.

24.1 Unvorhersehbare Risiken

Im Kap. 16 wurde bereits darauf eingegangen, dass Risiken, die für den Test im Vorfeld bereits erkannt werden, zu bewerten sind.

Es kann durch erkannte Risiken oder durch Probleme, die während des Projektes auftreten, zu Verschiebungen in Bezug auf Termin, Kosten und Qualität kommen.

Terminverschiebungen während der Projektlaufzeit sind manchmal nur dann hinnehmbar, wenn der Funktionsumfang des Testobjektes im Nachhinein verringert wird.

Es kommt aber immer wieder vor, dass man im Rahmen der Planung auftretende Probleme noch gar nicht erkannt hat oder sie erkannt hat, aber sowohl das Eintrittsrisiko als auch den daraus entstehenden Verzug für gering erlassen hat. Wenn sich die Rahmenbedingungen im Unternehmen ändern, wenn sich Prioritäten verschieben oder neue Anforderungen von außen an die Organisation bzw. das Projekt herangetragen werden, muss man die gesamte Planung hinterfragen und neu aufsetzen.

In diesen Fällen ist es in jedem Fall erforderlich, dem Management den Umstand zu berichten und Vorschläge zu unterbreiten, die unterschiedliche Handlungsalternativen

aufzeigen und diese Handlungsalternativen bewerten. Wichtig ist, dass in solchen Fällen schnell und flexibel reagiert wird.

Dazu empfiehlt sich folgendes Vorgehen:

24.2 Zielbeiträge der Handlungsalternativen sichtbar machen

Mit der Beschreibung der Entscheidungssituation werden Ziele und Rahmenbedingungen benannt, die für eine gute Entscheidung wichtig sind. Mit den Lösungen und Handlungsalternativen werden die Möglichkeiten aufgezeigt, unter denen der Entscheider auswählen und für oder gegen die er sich entscheiden kann. Nun müssen im nächsten Schritt mögliche Folgen, Konsequenzen oder Ergebnisse aufgezeigt werden, die eine einzelne Handlungsalternative hat, wenn es darum geht, die Ziele und Ergebnisse zu erreichen und die Rahmenbedingungen zu beachten.

Die möglichen Folgen, Konsequenzen und Ergebnisse, zu denen die Wahl für eine Handlungsalternative führen kann, soll dabei möglichst nachvollziehbar, verständlich, plausibel, begründet und belegt herausgearbeitet werden. Dazu müssen Informationen zusammengestellt werden, die Entscheider bewerten können. Diese Darstellung soll dem Testmanager oder anderen Entscheidern helfen, die richtige Entscheidung zu treffen im Sinne von: diese Auswahl trägt am besten zur Zielerreichung bei.

24.3 Methoden und Werkzeuge als Entscheidungstechniken

Dabei gibt es eine Fülle von Methoden und Werkzeugen, die dabei helfen, die Informationen so zusammenzustellen, auszuwerten und darzustellen, dass die Entscheider in der Lage sind, die Folgen und Konsequenzen zu bewerten. Diese Entscheidungstechniken haben alle den Zweck, Transparenz in die Entscheidungssituation zu bringen und sichtbar zu machen, welche Alternative vor dem Hintergrund der Ziele und Bewertungskriterien die beste wäre. Sie helfen, die Unsicherheit beim Entscheider zu verringern.

Welche Entscheidungstechniken, Methoden und Werkzeuge sich am besten eignen, ergibt sich vor allem aus nachfolgenden Fragen:

- Worüber soll entschieden werden?
- Welche Ziele spielen die wichtigste Rolle?
- Welche Bewertungskriterien sind deshalb zu beachten und zu bewerten?
- Was sind die wichtigen Rahmenbedingungen?
- Wer sind die Entscheider?
- Welche Interessen haben weitere Stakeholder?
- Wie komplex ist die Entscheidung?

- Wie weitreichend sind die Entscheidung und die damit verbundenen Folgen? (Zeithorizont, Risiko- oder Chancenpotenzial)
- Welche Art von Lösungen gibt es?
- Welche Informationen liegen vor?

Vor diesem Hintergrund werden in der Unternehmenspraxis meistens die folgenden Entscheidungstechniken, Methoden und Werkzeuge eingesetzt – einzeln oder in Kombination, um verschiedene Aspekte der Entscheidungsfolgen besser sichtbar zu machen.

24.4 Methoden der Investitionsrechnung

Bei diesen Methoden stehen betriebswirtschaftliche Kalküle im Vordergrund. Dazu gehören Gewinn, Rentabilität, Wirtschaftlichkeit, Profitabilität oder Produktivität. Die Handlungsalternativen sollen dazu beitragen, diese zu verbessern. Inwieweit dies im Einzelfall gelingt und welche Entscheidung deshalb betriebswirtschaftlich sinnvoll ist, machen diese Methoden sichtbar:

- Kostenvergleichsrechnung,
- Gewinnvergleichsrechnung,
- Rentabilitätsrechnung,
- Amortisationsrechnung sowie
- Cashflow-Rechnungen (mit unterschiedlichen Varianten wie: Kapitalwertermittlung, Net Present Value, Discounted Cashflow oder Economic Value Added).

Dahinter stehen Berechnungsverfahren, die davon ausgehen, dass Kosten (oder Ausgaben und Auszahlungen) auf der einen Seite sowie Umsatz und Erlöse (oder Einnahmen und Einzahlungen) auf der anderen Seite bekannt, berechenbar oder zumindest zuverlässig abschätzbar sind.

24.5 Argumentenbilanz

Wenn keine verlässlichen Zahlen zur Wirtschaftlichkeit, Rentabilität, Gewinn, Umsatz oder Kosten vorliegen, müssen qualitative Verfahren herangezogen werden, um Handlungsmöglichkeiten und ihre Folgen zu bewerten. Hier geht es darum, Argumente zu finden, die für oder gegen eine Lösung sprechen. Diese Argumente müssen nachvollziehbar und schlüssig sein. Das gelingt, wenn geeignete Belege, also Beispiele, Indikatoren, Kennzahlen oder Expertenerfahrungen dazu angeführt und benannt werden.

Eine einfache Methode für eine entsprechende qualitative Bewertung ist die **Argumentenbilanz**. Sie zwingt den Anwender dazu, seine wesentlichen Bewertungskriterien zu benennen und transparent zu machen – gerade auch für andere Personen. Implizite Bewertungskriterien werden explizit dargestellt. Dadurch werden subjektive Meinungen sichtbar und zur Diskussion gestellt.

Wie bei einer betriebswirtschaftlichen Bilanz mit Aktiva und Passiva werden für jede Handlungsmöglichkeit, die zur Entscheidung steht, zwei Seiten einander gegenübergestellt: Pro und Kontra. Kurz und knapp werden in beiden Spalten die wesentlichen Argumente benannt, die für oder gegen eine Alternative sprechen. Die Argumente sollten explizit Bezug nehmen zu den Zielen und gewünschten Ergebnissen, wie sie mit der Entscheidungssituation herausgearbeitet wurden. Der Nutzer der Argumentenbilanz muss dann selbst entscheiden, welche Argumente er zur Gesamtbewertung der Lösung heranziehen will und welche ihn überzeugt. Dabei wird zwischen Pro und Kontra abgewogen. Jede der Alternativen kann dabei Punktzahlen erhalten, und die Alternative mit der höheren Punktzahl gewinnt.

Bei einer Argumentenbilanz sollten möglichst sachlich und neutral alle Argumente für beide Seiten zusammengestellt werden und nicht eine vorgefasste Meinung in die Auswahl und Formulierung der Argumente einfließen. Das ist oft ein schwieriges Unterfangen. Wenn die Argumentenbilanz im Team erstellt wird, wird die Gefahr verringert, dass eine einseitige Sichtweise vorherrscht und sich eine Person mit ihrer Meinung durchsetzt. Dazu müssen aber die Alternativen von allen Teilnehmern (wie in einem Brainstorming) notiert werden, sodass nicht nur die Wortführer sich durchsetzen.

24.6 SWOT-Analyse

Die **SWOT-Analyse** (englische Abkürzung für Strengths (Stärken), Weaknesses (Schwächen), Opportunities (Chancen) und Threats (Risiken)) ist ein Instrument der strategischen Planung. Sie dient der Positionsbestimmung und der Strategieentwicklung.

Argumente für oder gegen eine Handlungsalternative können unterschiedliches Gewicht haben. Das entsteht vor allem dadurch, wie zuverlässig sie gelten und wie sicher oder unsicher die entsprechenden Folgen und Ergebnisse eintreten. Diese Differenzierung kann mithilfe einer SWOT-Analyse dargelegt werden. Mit der Analyse der Stärken, Schwächen, der Möglichkeiten und Chancen sowie der Gefahren und Risiken werden die Güte der Alternativen in Bezug auf die Gegenwart (weitgehend sichere Folgen und Ereignisse) und die Zukunft (eher unsichere Folgen und Ereignisse) beleuchtet. So werden mit dieser SWOT-Analyse Argumente, die für oder gegen eine Alternative sprechen, differenzierter herausgearbeitet und dargestellt. Die Stärken und Schwächen werden vor allem dann sehr bedeutsam, wenn sie mit Chancen oder Risiken im Umfeld zusammenfallen. Nachfolgende Abbildung (Abb. 24.1) zeigt die Fragen einer SWOT-Analyse:

SWOT-Analyse Beispiele
Portfolio-Analyse

SWOT Portfolio-Analyse	Chancen	Risiken
Stärken	Stärken/Chancen Wie die Stärken einsetzen, um die Chancen optimal zu nutzen?	Stärken/Risiken Wie die Stärken einsetzen, um die Risiken zu minimieren?
Schwächen	Schwächen/Chancen Wie an den Schwächen arbeiten, um die Chancen besser zu nutzen?	Schwächen/Risiken Wie an den Schwächen arbeiten, um die Risiken besser zu meistern?

Abb. 24.1 SWOT-Analyse

24.7 Konsequenzentabelle

Grundlage für die Bewertung und Entscheidung ist, dass Sie alle Folgen oder Konsequenzen, die mit einer Handlungsmöglichkeit verbunden sind, möglichst gut sichtbar machen. Dabei hilft die Konsequenzentabelle. Sie stellt möglichst anschaulich, übersichtlich und umfassend alle Konsequenzen zusammen und macht die Alternativen so auf „einen Blick“ vergleichbar. Die Tabelle zeigt auf: Welche Konsequenzen oder Folgen hätte es, wenn sich der Entscheider für die jeweils betrachtete Alternative entscheidet?

Die Angaben in der Tabelle können zusätzlich erläutert und begründet werden. Oft sind Beispiele oder Kennzahlen als Belege notwendig. Diese müssen für die Zukunft fortgeschrieben oder abgeschätzt werden. Manchmal gelingt es, mit einer **Konsequenzentabelle** die Handlungsmöglichkeiten direkt zu identifizieren, die in jedem Fall schlechter sind als andere Alternativen. Die folgende Tabelle (Tab. 24.1) ist ein Beispiel für eine Konsequenzentabelle.

In nachfolgendem Beispiel steht ein Integrationstest einer Software unmittelbar bevor. Die Komponente B soll neu entwickelt werden und dabei neue Funktionen bekommen, sie ist aber mit der Entwicklung noch nicht fertig. Es besteht allerdings eine schon bestehende Komponente B mit geringerem Funktionsumfang, die sofort integriert werden könnte, die aber nicht innovativ ist und den Softwarestand aus dem vorigen Produkt hat. Man könnte ggf. die neue Funktion aus Komponente B abkündigen, es wäre aber dem Image des Unternehmens abträglich.

Tab. 24.1 Konsequenzentabelle IT

Ziele oder Bewertungskriterium	Alternative			
	1	2	3	4
	Mehr Entwickler für Fertigstellung der Komponente B einsetzen, Integrationstest etwas später starten	Beginn des Integrationstests verschieben, weil Komponente B noch nicht fertig ist	Bestehende Komponente B verwenden, neue in einem künftigen Produkt (Projektescope verkleinern)	Integrationstest mit bestehender Komponente B starten, neue Komponente B später austauschen und neu integrieren, aber ausliefern
Kosten	Höhere Kosten ggü. Kalkulation	fast kostenneutral	geringere Kosten ggü. Kalkulation	Höhere Kosten ggü. Kalkulation
Termine	Können noch gehalten werden	Kann zu Projektende kritisch werden	Terminziele können sicher gehalten werden	Kann zu Projektende kritisch werden
Qualität/technisches Risiko	Evtl. leichter Qualitätsverlust wegen späterem Teststart	Kommt darauf an, wie viele Tester zum Integrationstest herangezogen werden können	Qualitätsziele sind unkritisch	Kommt darauf an, wie viele Tester zum Integrationstest herangezogen werden können
Image beim Kunden	Neutral	Neutral aber riskanter Weg	Negativ	Neutral aber riskanter Weg
...				

24.8 Entscheidungsbaum

Der Entscheidungsbaum gibt ebenfalls eine Übersicht über mögliche Handlungen (Alternativen) und ihre wahrscheinlichen oder weniger wahrscheinlichen Konsequenzen, Folgen und Wirkungen – insbesondere in Bezug auf die Ziele und Erwartungen. Mit dem Entscheidungsbaum wird ein Bild entwickelt, in dem ausgehend von der Ist-Situation und der Entscheidungsfrage die Handlungsalternativen und möglichen Entscheidungen als Verzweigungen dargestellt werden und danach mit möglichen Konsequenzen, Folgen und Wirkungen verknüpft werden.

Dabei können die jeweils möglichen Folgen mit einer Wahrscheinlichkeit für ihr Eintreten bewertet werden. Darüber lässt sich mithilfe von Erwartungswerten ermitteln, mit welcher Wahrscheinlichkeit ein Ziel erreicht wird, wenn die Entscheidung für eine Alternative getroffen wird. So können anhand von Eintrittswahrscheinlichkeiten und Erwartungswerten die einzelnen Alternativen bewertet und dadurch die favorisierte Alternative ermittelt werden. Die folgende Abbildung (Abb. 24.2) zeigt dies an einem beispielhaften Entscheidungsbaum.

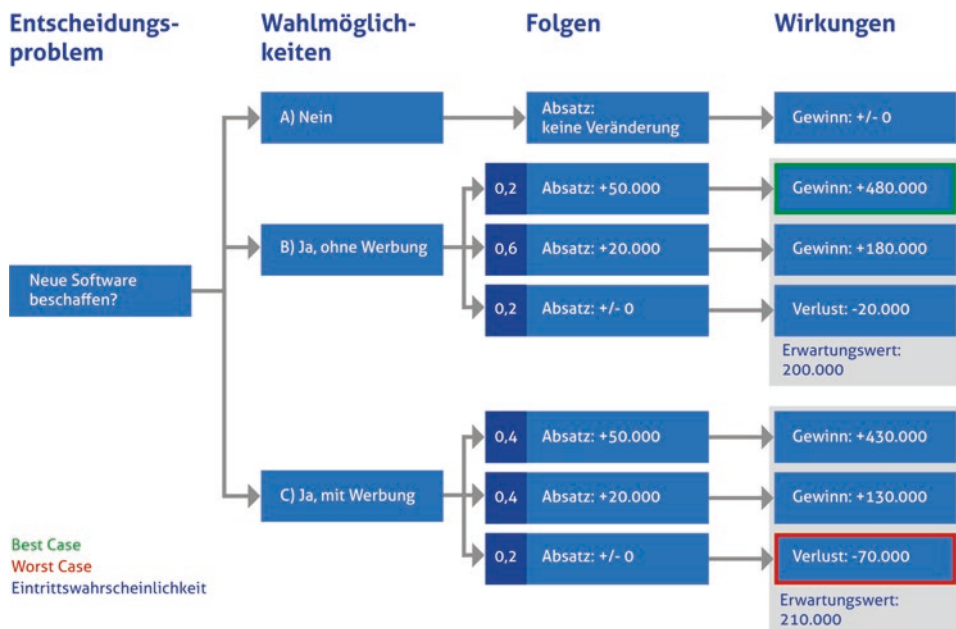


Abb. 24.2 Entscheidungsbaum für Beschaffung einer Software

In diesem Beispiel wird sichtbar, dass die Wahl der Alternative C die beste wäre, weil sie mit dem höchsten Erwartungswert für das Ziel „Gewinn maximieren“ verbunden ist. Allerdings ist damit auch der Worst Case verbunden, der maximal mögliche Verlust. Nun muss der Entscheider abwägen, welcher Faktor für ihn wichtiger ist. Wenn z. B. der maximal mögliche Verlust in jedem Fall zu vermeiden ist, kann im Resultat auch bewusst die zweitbeste Alternative gewählt werden.

Mit dem Entscheidungsbaum kann dargestellt werden, wenn es Konsequenzen gibt, die auf keinen Fall zulässig sind. Dann scheiden alle Handlungsmöglichkeiten aus diesem Zweig aus. Allerdings kann sich im Rahmen der Bewertung eine bevorzugte Wirkung ergeben. Dann sollte versucht werden, diese bevorzugte Wirkung in die entsprechenden Handlungen umzusetzen.

24.9 Nutzwertanalyse

Wenn mehrere Ziele und Kriterien bei der Bewertung von Handlungsmöglichkeiten eine Rolle spielen, diese aber einen sehr unterschiedlichen Charakter haben, können diese mithilfe der Nutzwertanalyse vergleichbar gemacht werden. Bei dieser Methode wird für jede Alternative geprüft, welchen Beitrag sie zu einem einzelnen Ziel oder Bewertungskriterium leistet. Dieser Zielbeitrag wird zunächst in Form eines Indikators angegeben, der den Zielbeitrag angemessen darstellt. Dieser Indikator wird dann in einem zweiten Schritt in einen Nutzwert (Punkte oder Scorewert) überführt; je mehr Punkte, desto besser ist die

Tab. 24.2 Nutzwertanalyse

Bewertungs- kriterium	Gewich- tung	Alternativen										
		1	2	3	4							
		Mehr Entwickler für Fertigstellung der Komponente B einsetzen, Integrationstest etwas später starten	Beginn des Integrations- tests verschieben, weil Komponente B noch nicht fertig ist	Bestehende Komponente B verwenden, neue in einem künftigen Produkt (Projektescope verkleinern)	Integrationstest mit bestehender Komponente B starten, neue Komponente B später austauschen und neu integrieren, aber ausliefern							
		Ausprä- gung Indikator	Nutz- wert	Gewich- teter Nutzwert	Ausprä- gung Indikator	Nutz- wert	Gewich- teter Nutzwert	Ausprä- gung Indikator	Nutz- wert	Gewichteter Nutzwert		
Kosten	20	Höher	3	60	Ähnlich	7	140	Neutral	10	200	3	60
Termine	40	Neutral	10	400	Risikant	5	200	Neutral	10	400	7	280
Qualität/ technisches Risiko	30	Leichter Verlust	8	240	Risikant	5	150	Neutral	10	300	7	210
Image beim Kunden	20	OK	10	200	Etwas riskant	8	160	Negativ	0	0	9	180
Summe	100		29	720			670			900		730

Alternative in Bezug auf das betrachtete Ziel. Am Ende werden alle Punkte über alle Ziele und Bewertungskriterien addiert. Die Summe entspricht der Güte der Alternative; diejenige mit der höchsten Punktzahl ist die beste.

Dabei kann auch berücksichtigt werden, dass nicht alle Ziele und Bewertungskriterien für die Entscheider gleich wichtig sind. Sie erhalten damit ein Zielgewicht, das in die Berechnung des Punktwerts einfließt. Tab. 24.2 zeigt den Aufbau einer Nutzwertanalyse. Dabei wurde wieder das obige Beispiel mit den 4 Alternativen zur Komponente B herangezogen.

Im Beispiel aus Tab. 24.2 ist der gewichtete Nutzwert von Alternative 3 am höchsten. Fazit ist demnach, zunächst die bestehende Komponente zu integrieren und damit weiterzutesten und die neue Komponente B in einem künftigen Produkt zu verwenden. Der Projektscope wird verkleinert, um die Zielerreichung gewährleisten zu können.

24.10 Risikoanalyse

Wenn keine verlässlichen oder nachvollziehbaren Indikatoren vorliegen, die zeigen, wie gut eine Handlungsalternative zur Zielerreichung beiträgt, ist eine Bewertung sehr schwierig. Die Alternativen können nicht unmittelbar bewertet werden. Allenfalls lassen sich qualitativ Möglichkeiten oder Chancen einerseits sowie Gefahren oder Risiken andererseits benennen. Das wird mit der SWOT-Analyse geleistet. Manche Entscheider schauen dabei besonders auf die Chancen, andere besonders auf die Risiken.

Bei weitreichenden und schwer einschätzbaren Wirkungen und Folgen kann es hilfreich sein, sich die möglichen Risiken genauer anzuschauen und diese zu analysieren – um sichtbar zu machen, was im „Worst Case“ passieren kann. Damit werden Aspekte sichtbar wie:

- das Risiko muss in jedem Fall vermieden werden oder
- selbst im Worst Case sind die Folgen so überschaubar, dass die möglichen Chancen genutzt werden sollten.

24.11 Methoden und Werkzeuge zur Entscheidung auswählen und nutzen

Am Beginn sollten zunächst die Entscheidungssituation und die Entscheidungsfrage aufgeschrieben werden: Worüber soll entschieden werden?

Danach sollten möglichen Alternativen und Lösungen zusammengestellt werden, aus denen der Entscheider eine Wahl treffen soll. Dabei wird analysiert, welche Effekte, Wirkungen und Folgen mit diesen Alternativen in Bezug auf Ziele, Bewertungskriterien und Rahmenbedingungen verbunden sein können. Dazu müssen Informationen recherchiert

und Indikatoren benannt werden, die diese Zusammenhänge sichtbar machen. Dabei werden Kennzahlen gemessen oder Effekte abgeschätzt.

Anschließend wählt man eine oder mehrere Entscheidungstechniken sowie die Methoden und Werkzeuge. In diese gehen die ermittelten Informationen, Indikatoren oder Kennzahlen ein, werden verarbeitet und ergeben ein Bild von der Güte der Alternative. Schließlich wird geprüft, welche Entscheidungstechnik für die zugrunde liegende Fragestellung hilfreich ist. Zum Beispiel:

- betriebswirtschaftliche Kennzahlen (Kosten, Gewinn, Rentabilität, Wirtschaftlichkeit, Cashflow, Amortisation usw.),
- Argumentenbilanz,
- SWOT-Analyse,
- Konsequenzentabelle,
- Entscheidungsbaum,
- Nutzwertanalyse oder auch
- Risikoanalyse.

Diese Entscheidungstechniken können auch kombiniert werden, um so ein umfassenderes Bild über die Alternativen und mögliche Folgen zu zeichnen. Die Entscheider müssen aber immer in der Lage sein, alle diese Informationen zu verarbeiten und für ihre Entscheidung zu bewerten.

Geeignete Vorlagen und Templates für die einzelnen Techniken sind im Internet zu finden.

24.12 Entscheidungsprobleme, Handlungsalternativen und mögliche Folgen aufzeigen

Für das Entscheidungsproblem oder die zugrunde liegende Fragestellung können die einzelnen Handlungsalternativen und Wahlmöglichkeiten im Hinblick auf ausgewählte Kennzahlen bewertet werden. Dies lässt sich in Form eines Entscheidungsbaums anschaulich darstellen. In den folgenden Vorlagen ist dabei festzuhalten:

- Szenarien als mögliche Handlungsalternativen;
- mögliche Folgen dieser Szenarien in Form von quantitativ messbaren Indikatoren oder Kennzahlen;
- Erwartungswerte in Prozent für diese Folgen;
- mögliche Wirkungen für eine einzelne Folge – ebenfalls in Form einer adäquaten Kennzahl (Wirkungsfaktor).

Dann wird daraus ein Erwartungswert des jeweiligen Szenarios in Bezug auf den **Wirkungsfaktor** berechnet. So lässt sich der „Wert des Szenarios“ quantitativ bestimmen.

Mit den vorgestellten Entscheidungstechniken und ihren Methoden und Werkzeugen werden den Entscheidern die Folgen und Wirkungen sichtbar gemacht und aufgezeigt, welche Beziehungen zu seinen Zielen, Erwartungen und Wünschen bestehen. Das erleichtert die Entscheidung und macht sie auch für andere, Betroffene, besser nachvollziehbar. Mit diesen Verfahren werden vor allem die Argumente herausgearbeitet, die für oder gegen eine Handlungsmöglichkeit oder Alternative sprechen. Genau darauf kommt es in diesem Entscheidungsprozess an.

Doch bei aller Systematik und auch beim methodischen Vorgehen können immer falsche Entscheidungen getroffen werden. Denn jede Entscheidung basiert auf einer subjektiven Bewertung. Wichtig ist, sich diese bewusst zu machen und mögliche Fallen zu kennen, die dabei auftreten können.

Je länger Testprojekte dauern und je umfangreicher sie sind, desto höher sind die Risiken für unvorhergesehene Ereignisse.

Zusammenfassung

Im Testabschlussbericht erfolgt eine Freigabeempfehlung, danach wird die Genehmigung der Freigabe erteilt. Der Freigabeprozess mit der Empfehlung des Einsatzes der getesteten Software ist im Vorfeld zu definieren.

25.1 Freigabeempfehlung

Die für den Test verantwortliche Abteilung gibt im Allgemeinen nur eine **Freigabeempfehlung**, keine abschließende **Freigabe** nach Testabschluss für ein bestimmtes Testobjekt oder Projekt. Aufgrund der Testaktivitäten wird eine bestimmte Qualitätsstufe ermittelt, aus der sich ableitet, ob der Einsatz der Software empfehlenswert ist oder nicht. Dafür ist ein Testabschlussbericht erforderlich.

In der Regel wird zum Ende der Teststufen Integrationstest, Systemtest und Abnahmetest jeweils ein Testabschlussbericht erstellt.

Der **Testabschlussbericht** ist wie folgt aufgebaut:

- Bezeichnung des Projektes, des Testobjektes und der Testphase;
- Angaben, ob Wartungsmaßnahmen betroffen sind;
- Releases und Umgebungsstand;
- Abweichungen zum Testkonzept;
- bekannte Fehler: alle Fehler, die während des Projektes aufgetreten sind (mit ID, Bezeichnung und Status) und Fehler, die zu Ende der Testphase noch offen, also noch nicht erfolgreich nachgetestet wurden;
- ausgewählte Metriken, z. B. zur Testentwicklung, Fehler nach Status und Auswirkung, Fehlertrend;

- Testabdeckung und Überdeckungsgrade mit Referenz zur Traceability-Matrix;
- Freigabeempfehlung und Begründung, warum die Freigabe uneingeschränkt, bedingt oder keine Freigabe empfohlen wird;
- verbleibende Risiken für die Produktiveinführung bzw. die nächste Teststufe;
- Reproduzierbarkeit der Arbeitsergebnisse, Status der Testautomatisierung;
- gewonnene Erkenntnisse, Zusammenfassung der während der Testphase gewonnenen positiven und negativen Erfahrungen zur besonderen Beachtung in zukünftigen Testaktivitäten.

Der **Freigabeprozess** muss generell festgelegt werden, wer über den Einsatz der getesteten Software entscheidet. Wer die Freigabe erteilt, muss zu Beginn des Projektes in der Aufbauorganisation festgelegt werden. Je nach hierarchischer Bedeutung der Vorgänge kann dies ein Arbeitspaketverantwortlicher, ein Teilprojektleiter, der Projektleiter selbst oder der Lenkungsausschuss sein [PRMA2020].

Wenn eine Software eingesetzt wurde, deren Einsatz nicht empfohlen wurde, hat sich das Testmanagement mit der Dokumentation im Testabschlussbericht dahingehend abgesichert, dass die Risiken beschrieben sind und eventuelle Probleme im Falle der Erteilung einer Freigabe transparent sind.

Das Testkonzept ist ebenfalls freizugeben. Es empfiehlt sich, eine Version 0.1 zu erstellen, die einem Review unterzogen wird. Beim Review wird eine Befundliste erstellt, aus der hervorgeht, welche Passagen in welcher Form geändert werden sollen. Nach der Überarbeitung des Testkonzeptes wird eine Version 1.0 erstellt, mit der die Testaktivitäten starten. Laufende Überarbeitungen können mit weiteren Versionen 1.x erstellt werden, eine freigegebene Version wäre dann 2.0 oder höher.

Beim Review sollten Stakeholder aus dem Test, der Entwicklung, der Projektleitung und weiteren Kernfunktionen beteiligt werden. Ein eigenes Review-Meeting ist dabei unbedingt zu empfehlen. Danach ist das Testkonzept von der Projektleitung freizugeben.

Es ist im Vorfeld organisatorisch festzulegen, wer die Freigabe zur Software erteilen darf und damit auch die Konsequenzen dieser Entscheidung tragen muss.

Im Falle einer nicht erteilten Freigabe sind Kriterien zu definieren, welche Funktionen bis wann nachgebessert sein müssen und welche Arbeiten im Einzelnen durchzuführen sind, damit die Freigabe erfolgen kann. Dabei sollte man auch festlegen, dass es bei diesen Kriterien bleiben muss, also nicht bei der nächsten Freigabe weitere Punkte herangezogen werden, die beim ersten Mal nicht bemängelt wurden, da sonst die Freigabe weiter herausgezögert wird und es zu unendlich wirkenden Freigabeschleifen kommen kann.

In dem Fall, dass der Kunde eine Abnahme erteilt, an die Zahlungsverpflichtungen gebunden sind, kann es vorkommen, dass der Kunde gar kein Interesse daran hat, die Abnahme schnell zu erteilen, sondern eher das Projekt in die Länge zieht, da es dann auch finanziell gestreckt werden kann. Irgendeinen Grund, eine Abnahme zu versagen, lassen sich bei genügend tiefer Analyse sicher finden. Daher sind vertragliche Vereinbarungen mit dem Kunden und eindeutig definierte Abnahmekriterien im Vorfeld wichtig.

Literatur

[PRMA2020] <https://www.projektmagazin.de/glossarterm/freigabe>, zugegriffen am 29.05.2020

Zusammenfassung

In der Praxis sind unterschiedliche Formen der Projektorganisation anzutreffen, die sich auf die Organisation des Testprojektes auswirken. Die Matrixorganisation ist dabei die in der Praxis am meisten angetroffene Organisationsform. Unterschiedliche Indices können zur Bewertung des Projektfortschritts herangezogen werden.

In der Regel ist die Form der **Projektorganisation** bereits vorgegeben. Folgende typische Ausprägungen von Organisationsformen sind dabei in der Praxis anzutreffen:

26.1 Stablinien-Projektorganisation

Die Stablinien-Projektorganisation (Abb. 26.1) wird auch als Einflussorganisation bezeichnet. Der Projektleiter verfügt weder über fachliche noch über disziplinarische Weisungsbefugnis und hat oft kein festes Projektteam.

Der Projektleiter kann dabei seinen Einfluss geltend machen. Oft werden derartige Projekte durchgeführt, die direkt von der Geschäftsleitung oder zumindest von einer hoch aufgehängten Abteilung unterstützt werden. Diesen Einfluss kann der Projektleiter nutzen, um im Projekt voranzukommen.

Beispiel: Im Unternehmen soll ein neuer Prozess zum Qualitäts- und Testmanagement eingeführt werden. Dieser Prozess soll abteilungsübergreifend implementiert und die Mitarbeiter geschult werden. Dieses Projekt wird von der Geschäftsführung hoch priorisiert. Der Projektleiter fungiert nun als Berater bzw. Koordinator in den betroffenen Abteilungen. Hierbei kann er flexibel auf Ressourcen zugreifen und Zuarbeiten mit der Rückende-

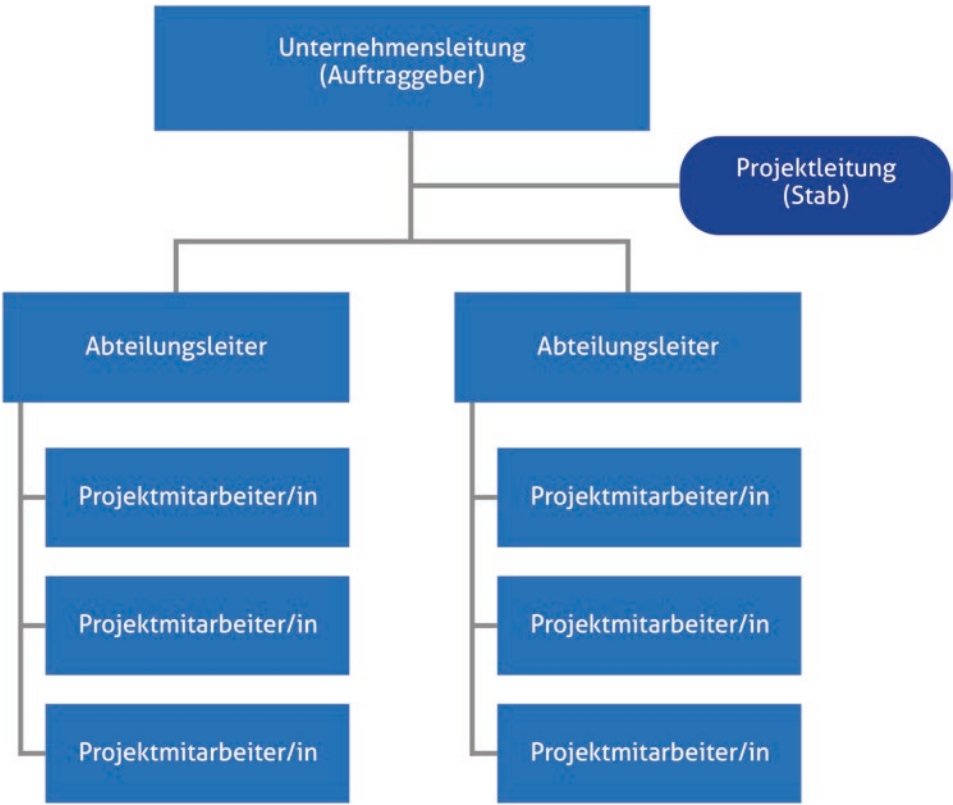


Abb. 26.1 Stablinien-Projektorganisation

Tab. 26.1 Vor- und Nachteile der Stablinien-Projektorganisation

Vorteile	Nachteile
• Die Organisationsstruktur wird nicht verändert • Die Projektorganisation kann schnell eingerichtet und wieder aufgelöst werden • Projektmitarbeiter verbleiben in ihren Abteilungen und können ihr Fachwissen von dort beziehen	• Der Projektleiter verfügt über keine direkten Weisungsrechte • Abstimmung und Koordination ist sehr aufwändig • Maßnahmen sind häufig schwierig durchzusetzen

ckung der Geschäftsführung einfordern. Die Stablinien-Projektorganisation hat folgende Vor- und Nachteile (Tab. 26.1).

- Einsatzgebiete der Stablinien-Projektorganisation:
- kleine und nichtkritische Projekte und
 - strategische und abteilungsübergreifende Projekte.

26.2 Reine Projektorganisation

Die reine Projektorganisation (Abb. 26.2) bezeichnet man auch als **autonome Projektorganisation**. Für diese Organisationsform werden Mitarbeiter für die Dauer des Projektes komplett aus ihrer Abteilung herausgelöst und sind ausschließlich für das Projekt tätig. Für diesen Zeitraum übernimmt der Projektleiter nicht nur das fachliche, sondern auch das disziplinarische Weisungsrecht. Bei dieser Form der Projektorganisation handelt es sich um die stärkste Abgrenzung zur Stammorganisation.

Beispiel: Ein Unternehmen baut zu Expansionszwecken eine eigene Entwicklungs- und Testabteilung für eine Produktlinie im Ausland auf. Die Infrastrukturanbindung ist Teil des Projektes.

Das Projektteam wird aus verschiedenen Experten zusammengestellt, die für die Projektlaufzeit ihre Position in den jeweiligen Abteilungen verlassen und ausschließlich für das Projekt arbeiten. Die reine Projektorganisation ist durch folgende Vor- und Nachteile charakterisiert (Tab. 26.2).

Einsatzgebiete der reinen Projektorganisation sind vor allem:

- besonders große Projekte,
- Auslandsprojekte oder
- terminkritische Projekte.

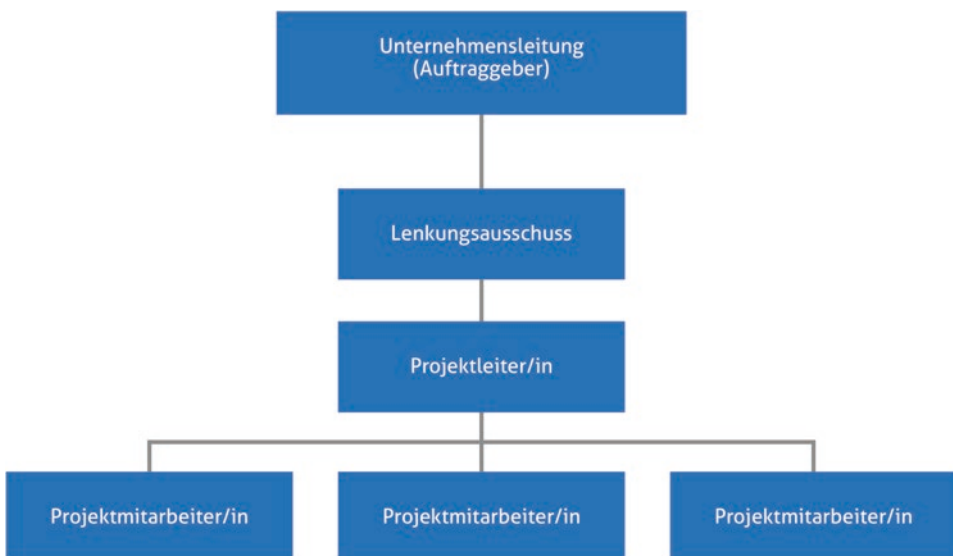


Abb. 26.2 Autonome Projektorganisation

Tab. 26.2 Vor- und Nachteile der reinen Projektorganisation

Vorteile	Nachteile
• Hohe Identifikation der Projektmitarbeiter mit dem Projekt • Klare Strukturen und sowohl fachliches als auch disziplinarisches Weisungsrecht durch den Projektleiter • Direkte Kommunikation	• Hoher Aufwand bei Wiedereingliederung der Projektmitarbeiter in die Linienorganisation • Wissensaustausch mit Fachabteilungen schwierig

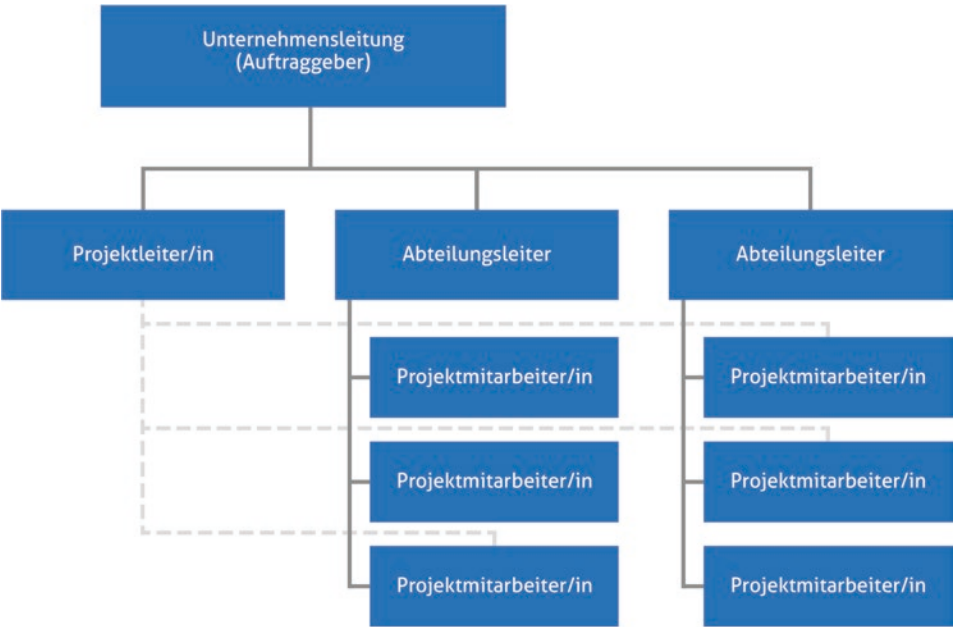


Abb. 26.3 Matrixorganisation

26.3 Matrixorganisation

Die Matrixorganisation (► Abb. 26.3) wird in der Praxis am häufigsten angewendet und stellt eine Art Mischform zwischen Stabs- und reiner Projektorganisation dar.

Die Projektmitarbeiter verbleiben in ihren Fachabteilungen und stehen dem Projekt zu einem gewissen Prozentsatz zur Verfügung. Der Abteilungsleiter übt weiterhin das disziplinarische Weisungsrecht und das fachliche Weisungsrecht für Anforderungen der Linienorganisation aus. Der Projektleiter verfügt über ein fachliches Weisungsrecht, was die Projektinhalte betrifft.

Beispiel: In einem Softwareunternehmen soll ein neues Produkt entwickelt werden. Mehrere Mitarbeiter unterschiedlicher Forschungs-, der Entwicklungs- und der Qualitätssicherungs-Abteilungen stehen dem Projekt zu unterschiedlichen Anteilen ihrer Arbeits-

zeit zur Verfügung. Während der Projektlaufzeit berichten sie sowohl an ihren Projektleiter als auch ihren Abteilungsleiter, was ihre Aufgaben des Tagesgeschäfts betrifft. Dabei kann es zu Konflikten und Ressourcen-Überlagerungen oder Fragen der Priorisierung von Aufgaben kommen, sodass die Kommunikation in dieser Organisation von großer Bedeutung ist. Die Matrixorganisation hat folgende Vor- und Nachteile (Tab. 26.3).

Einsatzgebiete der Matrixorganisation:

- mittelgroße bis große Projekte sowie
- terminkritische Projekte.

Zusammenfassend verfügt der Projektleiter je nach Organisationsform über unterschiedliche Weisungsbefugnisse und muss sich mit den Vorgesetzten der einzelnen Abteilungen abstimmen [ERFP2019].

26.4 Ausbalancierte Matrixorganisation

Um die möglichen Nachteile einer Matrixprojektorganisation – wie beispielsweise Konflikte zwischen Abteilungsleiter und Projektleiter – zu entschärfen, kann es sinnvoll sein, dem Projektleiter einen Lenkungsausschuss vorzuschalten. Dieser ist dann u. a. für die Ressourcenzuteilung zuständig (► oben) [PROR2019].

Wenn in diesem Lenkungsausschuss dann auch die relevanten Abteilungsleiter vertreten sind, verringert das meist die Ressourcenkonflikte zwischen Projektaufgaben und Tagesgeschäft, da die Abteilungsleiter ja an diesen Entscheidungen beteiligt sind (► Abb. 26.4).

26.5 Befugnisse und Zieleverantwortung des Projektleiters

Tab. 26.4 zeigt einen Überblick über die Befugnisse und Zieleverantwortung des Projektleiters für alle 3 Organisationsformen.

Tab. 26.3 Vor- und Nachteile der Matrixorganisation

Vorteile	Nachteile
• Mitarbeiter verbleiben in ihren Abteilungen, sodass eine Wiedereingliederung nach Projektende nicht erforderlich ist • Mitarbeiter können vom Wissen und dem Austausch in ihrer Fachabteilung profitieren • Flexibler Zugriff auf Ressourcen	• Konflikte zwischen Projekt- und Abteilungsleitung möglich (Mitarbeiter haben zwei Vorgesetzte) • Probleme bei der Priorisierung zwischen Linien- und Projektarbeit

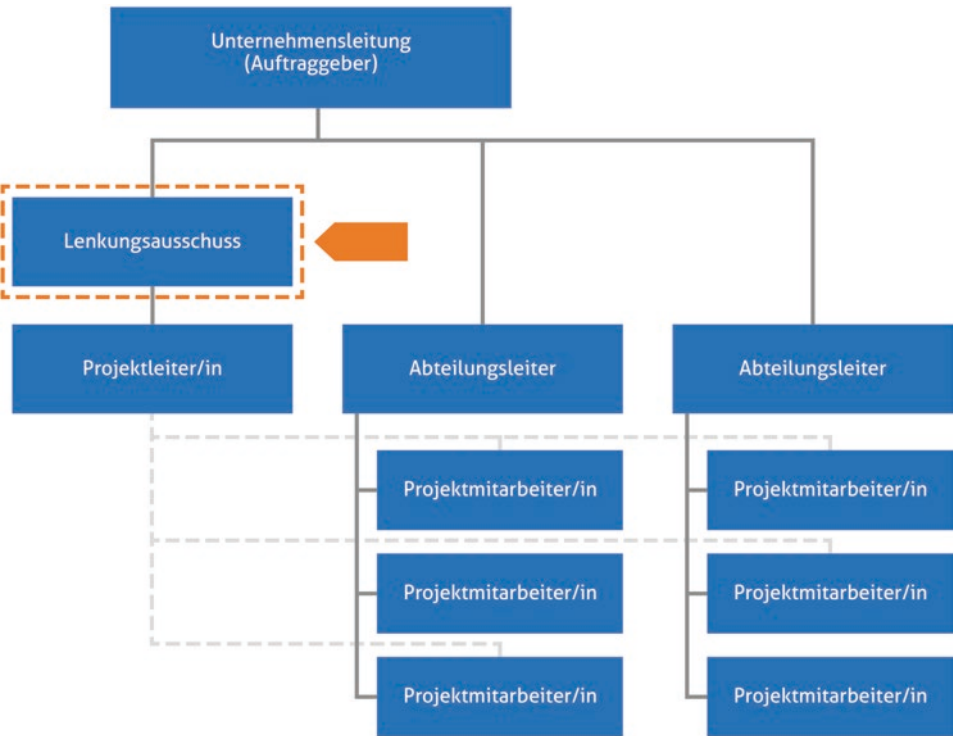


Abb. 26.4 Ausbalancierte Matrixorganisation

Tab. 26.4 Befugnisse und Zieleverantwortung des Projektleiters

Projektleiter	Projektorganisation		
	Autonom	Matrix	Stablinien
Fachliche Weisungsbefugnis	Ja	Ja	Nein
Disziplinarische Weisungsbefugnis	Ja	Nein	Nein
Zieleverantwortung	Volle	Volle	Keine

26.6 Formeln zur Bewertung des Projektfortschritts

Der Projekterfolg lässt sich im Projektmanagement in unterschiedlichen Formeln bewerten. Einige von ihnen sind besonders auch für Testprojekte interessant, da sie den Status in Zahlen ausdrücken können (► Tab. 26.5).

Tab 26.5 Formeln zur Bewertung des Projektfortschritts

Kostenentwicklungsindex Cost-Performance-Index (CPI)	Fertigstellungswert (EV)/Ist-Kosten (AC)	Ein Wert >1 bedeutet, dass die Kosten geringer waren als sie ursprünglich geplant waren. <1 bedeutet folglich das Gegenteil
Terminentwicklungsindex Schedule-Performance-Index (SPI)	Fertigstellungswert (EV)/Plankosten	Ein Wert >1 bedeutet, dass der Projektfortschritt schneller als der Plan ist. Ein Wert <1 bedeutet dementsprechend das Gegenteil
Erwartete Restkosten Estimate to Complete (ETC)	BAV – EV (geht weiter wie bisher, CPI ist 1) oder (BAC-EV)/CPO, Planung wird um CPI korrigiert	Bei Variante 2 wird der CPI in die Restkostenschätzung mit aufgenommen. Wenn der CPI bisher z. B. 1,2 betragen hat, wird davon ausgegangen, dass dieser Wert auch in Zukunft erreicht wird und korrigiert die Restkosten entsprechend
Zeitabweichung Schedule Variance (SV)	Fertigstellungswert (EC) – Plankosten (PV) Earned Value – Planned Value	Bei der Ermittlung wird bestimmt, wie weit der aktuelle Fertigstellungsgrad vom geplanten Fertigstellungsgrad abweicht. Dabei wird die zeitliche Abweichung in einem monetären Wert bestimmt. Beispiel: Zum Zeitpunkt x sollen Arbeitspakete im Wert von 120.000 € erstellt, tatsächlich wurden aber in der geplanten Zeit Leistungen im Wert von 160.000 € erstellt. In der geplanten Zeit wurden Leistungen im Umfang von 40.000 € mehr erbracht als geplant.
Kostenabweichung Cost Variance (CV)	Leistungswert (EV) – Ist-Kosten (AC) Earned Value (EV) – Actual Costs (AC)	Hier wird ermittelt, inwieweit die Ist-Kosten vom Fertigstellungswert abweichen. Eine negative Kostenabweichung gibt an, dass mit den angefallenen Kosten weniger Leistungen erbracht wurden als ursprünglich geplant. In der Regel hat man hier einen monetären Wert (Euro, Dollar ...)

Literatur

[PROR2019]_: <https://projekte-leicht-gemacht.de/blog/pm-methoden-erklart/projektorganisations-formen/>, zugegriffen am 09.05.2020

[ERFP2019] : <https://erfolgreich-projekte-leiten.de/projektorganisation/>, zugegriffen am 09.05.2020

Zusammenfassung

Mehrere Modelle, wie z. B. TMMI, TPI oder TMap, unterstützen das methodische Vorgehen während des Testprozesses. Dadurch können Testprozesse zielgerichtet und mit den richtigen Prioritäten optimiert werden. Unterschiedliche Reifegrade definieren dabei den Fortschritt der entsprechenden Testorganisation.

27.1 Einsatz von Testmethoden

In Softwareprojekten schluckt das Testen einen erheblichen Anteil der zur Verfügung stehenden Zeit. Um diese Zeit zu verkürzen, verfallen IT-Manager häufig in Aktionismus. Es wäre stattdessen erheblich zielführender, den Reifegrad der Testorganisation zu verbessern.

Ein essenzielles Merkmal reifer Testorganisationen ist ein methodisches Vorgehen, wie es beispielsweise bei der Qualitätssicherung von Flugzeugbauern und Automobilherstellern längst Standard ist. Bei Softwareentwicklern kommt dieser Ansatz noch zu kurz. Der Grund heißt „Time-to-Market“. Dazu kommt, dass der Reifegrad der IT generell noch nicht so weit ist wie in klassischen Industrien. Häufiger als in anderen Branchen wird noch eher kreativ und künstlerisch vorgegangen. Der Einsatz geeigneter **Testmethoden** ist aber gerade wegen der zunehmenden Industrialisierung des Entwicklungsprozesses und des Softwaretests für einen nachhaltigen Erfolg der Testmaßnahmen erforderlich.

Die Entwicklungszyklen bis zum nächsten Release haben sich in den letzten Jahren rapide verkürzt. Gleichzeitig sind IT-Projekte komplexer geworden, und die IT spielt im Unternehmen heute überall eine geschäftskritische Rolle. Fallen Systeme aufgrund von Softwarefehlern aus, hat das in der Regel eine geschäftsschädigende Wirkung.

Um für die Softwarequalität zu sorgen, nimmt heute das Testen mehr Projektzeit in Anspruch. In Großprojekten schluckt es mindestens 30 % der Gesamtzeit für das Pro-

jekt; häufig summiert sich der Anteil auf 40–50 Prozent. Die Folge: Um die Deadline des Kunden zu halten und im Budget zu bleiben, spielen Softwareentwickler auf Risiko und lassen ihr Testmanagement oft schleifen. Gerade dann, wenn die Zeit für die Entwicklung ohnehin knapp bemessen ist, wird eher Zeit in den Testphasen verkürzt und Aufwand eingespart als geplante Features wegzulassen oder die Entwicklungsmannschaft zu verstärken. Der Ausweg aus diesem Dilemma aus Zeitdruck und steigenden Qualitätsansprüchen kann nur lauten, das Testen effizienter zu gestalten, beispielsweise die Abläufe zu beschleunigen.

Bei der Verbesserung der Prozesse verfallen IT-Manager allerdings häufig in Aktionismus anstatt zunächst genau zu analysieren, wie es um das eigene Testmanagement bestellt ist. Beliebte Ad-hoc-Maßnahmen sind die Automatisierung von Softwaretests, um Testkosten einzusparen, sowie das Auslagern bestimmter Tests an Spezialisten. Diese Maßnahmen sind zwar im Ansatz richtig. Für beide sollte das Testmanagement allerdings bereits einen bestimmten Level an strategischem und methodischem Vorgehen aufweisen. Man muss sich auch darüber bewusst sein, dass auch eine Testautomatisierung permanent gepflegt werden muss und keineswegs einen Selbstläufer darstellt.

Für eine wirkungsvolle Testautomatisierung sollten die Testfälle beispielsweise reproduzierbar und standardisiert sein. Und um Outsourcing effektiv einzusetzen, unterstützt eine umfassende Dokumentation aller mit dem Testen zusammenhängenden Abläufe die Zusammenarbeit mit Dienstleistern.

27.2 Geeignete Testmethoden und Teststrategie

Auch Begrifflichkeiten sollten eindeutig definiert sein und in einer **Teststrategie** festgehalten werden. Auftraggeber und Testdienstleister sprechen damit dieselbe Sprache, wodurch Unternehmen unnötige Kosten durch Kommunikationspannen vermeiden.

Fehlt dieses strategische Vorgehen, steigt das Risiko, dass die Kosten für zusätzliche Abstimmungsrunden, überflüssige Trainings und unnötige Tool-Lizenzen steigen. Schlimmstenfalls übersteigen die Investitionen sogar die anvisierten Einsparungen. Denn eine Automatisierung von Chaos führt eher zu weiterem Chaos, anstatt es zu beseitigen.

Ein wesentlicher Grund für diesen Aktionismus ist bei vielen Unternehmen, dass ihnen häufig gar nicht bekannt ist, in welchem Stadium sich ihre Testprozesse befinden. Inzwischen gibt es allerdings verschiedene Bewertungsmethoden, die Unternehmen an die Hand nehmen und einen Rahmen liefern, um den Status-quo des Testmanagements zu ermitteln und es schrittweise zu verbessern.

Die Methoden, beispielsweise TMMi, TPI oder TMap, verstehen sich dabei als Leitfaden. Testmanager können anhand definierter Kriterien den Reifegrad ihrer Prozesse einschätzen und erkennen, ob geplante Investitionen in bestimmte Verbesserungsmaßnahmen aktuell überhaupt sinnvoll sind oder ob nicht zunächst andere Schritte viel wichtiger sind [CIOD2019].

27.3 Testing Maturity Model (TMMI)

Das Testing Maturity Model Integration (TMMi) ist eine dafür oft verwendete Methode. Das Assessmentverfahren ist die Weiterentwicklung des mittlerweile als veraltet geltenden TMM, einem Reifegradmodell für Testprozesse, das von Ilene Burnstein und ihrem Team am Illinois Institute of Technology in Anlehnung an SW-CMM entwickelt wurde.

TMMi umfasst fünf Ebenen, von denen jeder einen Reifegrad des Testmanagements beschreibt (Tab. 27.1).

Auf der untersten Stufe, Level 1, ordnen sich beispielsweise Unternehmen ein, die Software nach dem Prinzip „Try and Error“ oder „Debugging“ testen. Softwaretests sind vor allem dazu da, Programmier- und Funktionsfehler zu finden und zu beseitigen. Auf der obersten Stufe, Level 5, haben Unternehmen die Prozessverbesserungen, die sich aus den Ebenen darunter ableiten, in die Testabläufe integriert. Der Test trägt dann nicht nur dazu bei, dass Fehler gefunden, sondern dass sie auch aktiv vermieden werden [CIOD2019].

Da TMMi eng an CMMI angelehnt wurde, gelten (mit Ausnahme von GG1, da TMMi nur als Staged Representation existiert) alle generischen Ziele und generischen Praktiken wie es sie auch in CMMI gibt. Somit wird das Thema Reviews in der GP 2.9 PROZESS-EINHALTUNG OBJEKTIV BEWERTEN und GP 2.10 UMSETZUNG MIT DEM HÖHEREN MANAGEMENT PRÜFEN angesprochen. Reviews sind dabei im allgemeinen

Tab. 27.1 Die fünf Ebenen des Maturity-Modells

Ebene	Level	Eigenschaften	Notwendige Verbesserungen
5	„Optimizing“	Rückwirkung der Verbesserungen auf den Prozess	• Organisation der Produktion auf optimaler Ebene • Kontinuierliche Prozessverbesserung etabliert
4	„Managed“	Quantitativ erfasster und verstandener Prozess	• Wechsel in der Technologie • Problemanalyse • Vermeidung von Problemen • Prozessverbesserungsvorschläge auf Basis von Kennzahlen
3	„Defined“	Prozess qualitativ definiert und institutionalisiert	• Analyse und „Messung“ des Prozesses • Quantitative Quartalspläne • Einheitliche Prozesse etabliert
.2	„Repeatable“	Intuitive, personenabhängige Beherrschung des Prozesses	• Ausbildung • Technische Praktiken (Reviews, Tests) • Konzentration auf Normen und Teams
1	„Initial“	Prozessablauf „chaotisch“ und Management „ad hoc“	• Projektmanagement • Projektplanung • Konfigurationsmanagement • Software-Qualitätssicherung

Sinne, d. h. als Überprüfung in beliebiger Form durch ein oder mehrere Personen gemeint. Diese Reviews können eher mit Audits oder anderen Prüfungen der Prozesskonformität und als Managementreviews und nicht mit der Überprüfung von Dokumenten gleichgesetzt werden.

Zwei Prozessgebiete von TMMi sind ausschließlich dem Thema **Peer Reviews** gewidmet. Dazu gehört ein Prozessgebiet, das die Durchführung informeller Reviews sowie von Walkthroughs, Technical Reviews und Inspektionen entsprechend der IEEE Std 1028-2008 beschreibt und ein Prozessgebiet zu fortgeschrittenen Peer Reviews im Reifegrad 4, der die Verankerung dieser Peer Reviews als strategischen Part des gesamten Testprozesses (also Teststrategie, Testplanung und Testansätze) mit entsprechenden Metriken darstellt. Die Beschreibung der Peer Reviews entspricht weitestgehend den Kriterien der IEEE Std 1028-2008 für formale Reviews. Die geforderten spezifischen Ziele und spezifischen Praktiken sind unabhängig vom Reviewtyp und konzentrieren sich auf die Besonderheiten von Reviews im Bereich des Testens. Dies zeigt sich vor allem an der spezifischen Praktik SP 2.2 „Tester reviewen die Dokumente der Testbasis“ mit den typischen Arbeitsprodukten „Testbarkeits-Fehler“ und „Testbarkeits-Reviewbericht“.

Die Institutionalisierung von Peer Reviews, also die Anwendung der generischen Ziele und generischen Praktiken sind eine gute Quelle für die Aktivitäten, die Organisationen bei der Einführung und Optimierung von Reviews beachten müssen. Die organisationsweise Politik bezüglich Reviews, die gerade auch die allgemein gültigen Ziele und Planungen der Reviews festlegt und die Trainings verlangt, ist eine Grundlage für die erfolgreiche Einführung von Reviews in Unternehmen. Zusätzlich werden mit TMMi die wichtigsten Inhalte für die Trainings von Gutachtern und für die Trainings von Reviewmoderatoren. Besonders hervorzuheben ist dabei die klare Anforderung, dass die bereitgestellten Checklisten für das Review ihrerseits einem Review unterzogen sein müssen durch die „Peers“ und die potenziellen Nutzer der Checklisten. Problematischer ist es, die für das Unternehmen wichtigen Kennzahlen zu ermitteln und auch auszuwerten. Dazu muss man aus der Menge der Kennzahlen erst prüfen, ob sie überhaupt ohne weiteres vorhanden und wie sie zu generieren sind. Außerdem muss man sich darüber im Klaren sein, welche Indexwerte welche Interpretationen ergeben, und was daraus im Einzelnen für die betriebliche Praxis folgt.

Immer wieder entstehen im Laufe des Testprozesses Arbeitsprodukte, und konsequenterweise fordert TMMi das Review dieser Arbeitsprodukte auch in passenden spezifischen Praktiken. In TMMi haben Reviews eine hohe Bedeutung, die bereits in den Forderungen für die Erreichung des Reifegrades 3 deutlich wird. Die Inhalte der spezifischen Praktiken für Peer Reviews sind konform zu IEEE Std 1028-2008 und deutlich formal. Die Reviewdurchführung ist unabhängig vom Reviewtyp und unabhängig von den zu prüfenden begleitenden Dokumenten dargestellt und formal beschrieben. In einzelnen spezifischen Praktiken werden die Besonderheiten dargestellt, wenn Tester als Reviewer fungieren [QUAL2010].

27.4 TMap (Test Management Approach)

TMap ist ein Modell im Bereich des Testens und der Qualitätssicherung von Software in der alle anfallenden Aspekte, das Umfeld und die Vorgehensweise strukturiert werden.

Damit ist TMap (1995 entwickelt, eingetragenes Warenzeichen von Sogeti Nederland B.V.) und Standard in vielen Organisationen weltweit. Es kann **TPI (Test Process Improvement)** aus derselben Unternehmensgruppe gegenübergestellt werden. Während TMap die Tests selbst strukturiert, will TPI den gesamten Testprozess optimieren. Damit befindet sich TPI auf der Managementebene, während TMap im konkreten Projekt eingesetzt werden soll. Dabei basiert TMap auf praktischer Erfahrung und stellt damit keine theoretische, sondern eine pragmatische Methode dar. Die Methode ist universell und branchenübergreifend einsetzbar und für die konkreten Bedürfnisse in dem jeweiligen Testprojekt bzw. in der jeweiligen Testsituation anpassbar.

Der Testprozess wird dabei in 4 Bereiche, die sogenannten Kernbausteine unterteilt:

1. geschäftsbasiertes Testmanagement,
2. vollständig strukturierter Testprozess,
3. vollständiger Werkzeugsatz und
4. adaptiver, flexibler Testprozess.

Geschäftsbasiertes Testmanagement (**BDTM: Business Driven Test Management Approach**) bedeutet dabei, dass der **Geschäftsvorfall (Business Case)** die Rechtfertigung für ein Projekt bildet und die Fragen beantwortet, warum das Projekt durchgeführt wird, welche Investitionen erforderlich sind und was der Kunde mit den Ergebnissen erreichen will.

Während des Projektes wird der Business Case zu definierten Zeitpunkten verifiziert, um sicherzustellen, dass die daraus folgenden Ergebnisse für den Kunden immer noch gültig sind. TMap unterstützt die Rechtfertigung der IT und überträgt sie auf die Testaktivitäten. TMap geht davon aus, dass ein Projektansatz, der auf einem Geschäftsfall basiert, folgende Merkmale aufweist:

- Der Ansatz zielt darauf ab, ein vorher festgelegtes Ergebnis zu erreichen.
- Das Gesamtprojekt, mit dem dieses Ergebnis erreicht werden soll, wird innerhalb der verfügbaren Laufzeit realisiert.
- Die Kosten des Projektes, mit dem dieses Ergebnis erzielt werden soll, sind gegenüber dem Gewinn, den die Organisation erzielen will, angemessen.
- Die Risiken bei der Inbetriebnahme sind bekannt und so gering wie möglich. All das bewegt sich innerhalb der Rahmenbedingungen, die durch die oben genannten Merkmale festgelegt werden.

Die vier oben beschriebenen Faktoren der **IT-Governance** (Ergebnis, Risiko, Zeit und Kosten) finden sich in diesen Merkmalen. Für die erfolgreiche Durchführung eines Projektes ist es wichtig, dass der Testprozess mit dem Business Case im Einklang steht. Die Beziehung zwischen dem Business Case und dem Testprozess wird durch den BDTM-Ansatz hergestellt. Der BDTM-Ansatz hat folgende Schritte:

- Formulierung des Auftrags und Zusammenstellung der Testziele: Testziele müssen immer eine Relevanz sowohl für den Kunden als auch für die Auftraggeber bzw. deren Vertreter haben. Testziele werden häufig in Bezug auf IT-unterstützte Geschäftsprozesse, umgesetzte Benutzeranforderungen oder definierte Risiken formuliert.
- Bestimmung der Risikoklasse für jede Kombination von Merkmal und Teilobjekt: In einem generischen Dokument („**Mastertestplan**“) wird die Festlegung über vorgesehene Teststufen getroffen. Dabei wird für jedes Testziel eine Risikoklasse definiert.
- Ermittlung, wie gründlich die Kombination eines Merkmals mit einem Teilobjekt getestet werden muss. Aus der zuvor ermittelten Risikoklasse wird das Ergebnis für jede Teststufe in einer Strategietabelle dokumentiert.
- Die Gesamtschätzung für alle Tests wird durchgeführt und die Testplanung erstellt. Nach Präsentation vor dem Kunden und anderen Stakeholdern werden ggf. die Risikoklasse und die Strategietabelle entsprechend angepasst.
- Schließlich wird aus der Kombination aus Merkmal und Teilobjekt die richtige Testtechnik ausgewählt. Wenn der Kunde und die anderen Interessenvertreter sich über die Abschätzung und die Planung einig sind, stellt der Testdesigner die Testdesigntabelle fertig. Dabei werden die Entscheidungen über mehr oder weniger gründliches Testen in konkrete Aussagen über die gewünschte Testabdeckung übersetzt.
- Während des Testprozesses berichtet der Testmanager über den Fortschritt des Testprozesses, die Qualität und Risiken des Testobjektes sowie die Qualität des Testprozesses.

TMap unterstützt einen strukturierten Testprozess. Im Mastertestplan wird festgehalten, welche Funktionen mit welcher Intensität getestet werden müssen. Sowohl für den Mastertestplan als auch für die einzelnen Teststufen (Modul-, Integrations-, System-, Abnahmetest) muss dazu ein gut funktionierender Prozess für die Aktivitäten der Planung, Vorbereitung, Durchführung und Steuerung aufgesetzt werden. Zusätzlich zu diesen Prozessen sind bestimmte Aspekte zentral zu verwalten und nicht einzeln für jedes Projekt. Dies umfasst unterstützende Prozesse für

- Testrichtlinien: Festlegung, wie Menschen, Ressourcen und Methoden in Verbindung mit den Testprozessen eingesetzt werden und Definition des zentralen Supports.
- Permanente Testorganisation.
- Testumgebungen.
- Testwerkzeuge.
- Testexperten.

TMap unterstützt die korrekte Durchführung des strukturierten Testprozesses mit einem vollständigen Werkzeugsatz. Dieser konzentriert sich auf die Themen

- Techniken für Testaufwandsschätzung, Fehlermanagement, Erstellung von Metriken, Produktrisikoaanalyse, Testdesign und Produktprüfung. Diese Themen beschreiben, wie getestet wird.
- Techniken für die Infrastruktur, die definieren, wo und womit getestet wird. Dazu gehören Testumgebungen, Testwerkzeuge und Arbeitsplätze. Bei den Testwerkzeugen unterscheidet man dabei Testplanungs- und Teststeuerwerkzeuge, Testdesignwerkzeuge, Werkzeuge zur Testdurchführung und Werkzeuge zur Fehlersuche und Codeanalyse.
- Techniken für die Organisation beschreiben, wer testet. Dazu müssen Testrichtlinien, permanente Testorganisation, Testorganisation in Projekten, Testexperten und Testrollen festgelegt werden.

TMap ist ein Ansatz, der in allen Testsituationen und in Kombination mit allen Systementwicklungsmethoden angewendet werden kann. Der Tester wählt in Abhängigkeit von der Situation die Elemente von TMap, die er einsetzen will. Das macht TMap zu einer **adaptiven Methode**. Adaptiv bedeutet, dass man ein Element in Teilelemente aufspalten kann, die in einer anderen Situation ein neues, wertvolles Element für die spezifische Situation ergeben.

Die Anpassungsfähigkeit von TMap kann durch vier Eigenschaften ausgedrückt werden:

- Auf Änderungen reagieren: Wenn die Teststrategie und die daraus abgeleitete Schätzung und Planung für den Kunden nicht akzeptabel sind, wird der Plan angepasst. Das gibt dem Kunden die Kontrolle über den Testprozess und ermöglicht es ihm, diesen auf Grundlage einer Abwägung zwischen Ergebnis und Risiko einerseits und der Zeit andererseits zu steuern. Der Testmanager kann entscheiden, verschiedene Aspekte des Testplans in Abstimmung mit dem Kunden anzupassen.
- Produkte und Prozesse (wieder)verwenden: Dank einer großen Menge von Werkzeugen, die in Form von Testdesigntechniken, Checklisten und Vorlagen vorhanden sind, können Produkte und Prozesse schnell eingesetzt werden. TMap bietet verschiedene Formen der permanenten Testorganisation, um die Wiederverwendung von Produkten und Prozessen in der Organisation zu verankern.
- Aus Erfahrungen lernen: Die Bewertung des Testprozesses ist als Aktivität bei TMap eingebaut. Ein weiteres wichtiges Instrument ist der Einsatz von Metriken. Für den Testprozess sind Metriken über die Qualität des Testobjektes und über Fortschritt und Qualität der Testprozesse wichtig, um den Testprozess zu steuern, Testempfehlungen zu rechtfertigen und Systeme oder Testprozesse miteinander zu vergleichen.
- Erst probieren, dann verwenden: Die Annahme der Testbasis (unter Verwendung eines Testbarkeitsberichtes), der Testinfrastruktur und des Testobjektes ermöglichen es, diese vor der Verwendung zu testen [[TMAP2008](#)].

27.5 Vorteile von TMap

- Die Methode beruht auf der Erfahrung aus einer Vielzahl von Projekten.
- TMap berücksichtigt aktuelle Trends.
- Der Fokus liegt auf dem Testprozess.
- TMap optimiert Risikoabdeckung und Testtiefe.
- Die Einbindung der Stakeholder wird maximiert.

Literatur

- [CIOD2019] : <https://www.cio.de/a/test-methoden-tmimi-und-tpi-im-vergleich,2301883,3>,
zugegriffen am 09.05.2020
- [QUAL2010] : <https://thequaliteers.de/nachrichten-leser/items/reviews-in-test-maturity-model-in-integration-tmimi.html>, zugegriffen am 09.05.2020
- [TMAP2008] : Koomen van der Aalst Broekmann Vroon, TMap Next, dpunkt Verlag Heidelberg 2008

Zusammenfassung

TPI Next bewertet den Reifegrad von Testprozessen und stellt geeignete Best Practices zur Verfügung, um eine Testorganisation nachhaltig zu verbessern. Dazu wird der Testprozess in verschiedene Kernbereiche unterteilt. Für jeden der Reifegrade müssen bestimmte Kriterien erfüllt sein.

Das international anerkannte TPI-Modell hat sich als Standard zur Bestimmung und Verbesserung des Reifegrads von Testorganisationen etabliert. Daher soll es im Folgenden als Testmethode besonders hervorgehoben werden.

28.1 Ermittlung des Reifegrads des Testprozesses

Ziel von **TPI Next** ist es, Testprozesse nachhaltig zu verbessern und dafür geeignete Best Practices zur Verfügung zu stellen. Mit dem **TPI-Modell** wird der Reifegrad des Testprozesses ermittelt.

Allgemein sollte der Testprozess folgende Eigenschaften aufweisen:

- **Transparenz:** Der jeweils aktuelle Stand der Tests soll abrufbar sein. Alle Mitarbeiter haben ein Bild vom Gesamtprozess und wissen, was als nächstes zu bearbeiten ist und wann die aktuellen Aufgaben abgeschlossen sein müssen.
- **Handhabbarkeit:** Die Testaktivitäten sind in einzelne Arbeitspakete aufgeteilt.
- **Nachvollziehbarkeit:** Entscheidungen und Ergebnisse sollen auch im Nachhinein von Dritten nachvollzogen werden können.

- Planbarkeit: Prozessmeilensteine werden einbezogen und Erfahrungswerte für die Aufwandsschätzung der Arbeitspakete auf Testobjektebene werden herangezogen.
- Effizienz: Redundanzen werden vermieden und ein optimaler Ressourceneinsatz wird gestaltet [[HETT2019](#)].

TPI dient also zur Bewertung der übergeordneten Sichtweise über alle Projekte der betrieblichen Organisationseinheit und nicht nur für eine einzelne Projektmaßnahme. Die ermittelten **Reifegrade** wirken sich aber auf das einzelne Projekt unmittelbar aus. Viele in Projekten erkannten Schwächen und Probleme der Testprozesse können mit den Kriterien von TPI Next untermauert und erklärt werden. Mit dem Einsatz der TPI-Next-Methode soll eine schrittweise Verbesserung der Testprozesse ermöglicht werden.

Das TPI-Modell unterteilt den Testprozess hierfür in 20 verschiedene Kernbereiche, sogenannte Key Areas, denen jeweils Ebenen zugeordnet sind. Anhand der Ebenen lässt sich der Reifegrad eines Kernbereichs bestimmen – ähnlich wie bei TMMi. Jede höhere Ebene ist in zeitlicher, finanzieller oder qualitativer Hinsicht besser als die vorherige. Damit die Benutzer feststellen können, ob ein Kernbereich die Anforderungen einer Ebene erfüllt, definiert das TPI-Modell für jede Ebene eine Reihe von Kontrollpunkten. Sie dienen als Checkliste, die eine objektive und transparente Bewertung eines Kernbereichs ermöglichen.

Ein strukturierter Testprozess hat folgende Vorteile:

- Die Verwendung des Ansatzes ist in jeder Situation möglich und unabhängig von Geschäftsbereich, der Organisationsstruktur oder der gewählten Vorgehensweise der Systementwicklung.
- Der Ansatz liefert Aussagen über die Qualitätsrisiken des getesteten Systems.
- Wichtige Fehler werden in einer frühen Phase des Entwicklungsprozesses gefunden, sodass die Fehler früh behoben und die Kosten dadurch erheblich gesenkt werden können.
- Fehler in zukünftigen Entwicklungsprodukten werden durch Testbarkeitsreviews vermieden.
- Kosten, die durch materielle oder immaterielle Schäden verursacht werden, werden verhindert.
- Testfälle und Testprodukte sind wiederverwendbar.
- Das Testen befindet sich so kurz wie möglich auf dem kritischen Pfad des gesamten Entwicklungszyklus und kann dadurch die Gesamtlaufzeit des Projektes verkürzen.
- Der Testprozess ist verständlich und beherrschbar.

Dafür wurden zunächst das TPI-Modell und später das TPI-Next-Modell entwickelt. Im TPI-Next-Modell werden gegenüber dem TPI-Modell die Reifekategorien stärker betont, die Kontrollpunkte in einer **Testreifematrix** dargestellt und die Kernbereiche überarbeitet. Abhängigkeiten und Prioritäten zwischen Kernbereichen wurden durch Gruppen

ersetzt und das Modell wurde um die Testunterstützung in iterativen und agilen Entwicklungsprozessen erweitert.

Darüber hinaus wurden Verbesserungen durchgeführt, die den geschäftsbasierten Charakter des neuen Modells ausmachen:

- Gruppierung der **Kontrollpunkte**: Eine Gruppe besteht aus einer Anzahl zusammenhängender Kontrollpunkte verschiedener Kernbereiche. Auch wenn TPI Next im Rahmen einer allgemeinen Initiative zur Softwareprozessverbesserung verwendet wird, ist die Nutzung dieser Basisgruppierung zu empfehlen. Alle Situationen, in denen das klassische TPI-Modell genügt, werden durch die Basisgruppierung des TPI-Next-Modells erfasst. Wenn aber die Testprozessverbesserung auf ein spezifisches Resultat hin ausgerichtet werden soll – z. B. Reduktion der Testkosten oder verbesserte Time-to-Market – lassen sich die Gruppen neu strukturieren, was bedeutet, dass die effektivsten Optimierungsmaßnahmen ganz oben auf der Liste stehen. Dadurch ist TPI Next sehr vielseitig einsetzbar.
- **Enabler**: Das Testen ist ein integraler Bestandteil der Softwareentwicklung, der in Wechselbeziehung zu einer Vielzahl angrenzender Prozesse steht. Diese gegenseitige Beeinflussung von Testen, Entwicklung und Betrieb wird durch die Einführung von Enablern verdeutlicht: Prozesse, die in enger Wechselbeziehung mit dem Testen stehen und die bei angemessener Berücksichtigung einen positiven Einfluss auf die Testreife haben, und umgekehrt. Bei unzureichender Berücksichtigung sind die Einflüsse jedoch negativ. Dadurch spielen Enabler eine wichtige Rolle bei der Erkennung möglicher „Quick Wins“ sowie der Gewährleistung einer engen Anpassung an allgemeine Modelle zur Verbesserung von Softwareprozessen wie z. B. CMMI oder SPICE.

28.2 Testorganisation im Reifeprozess nach TPI Next

TPI Next bewertet den **Testprozess** anhand unterschiedlicher Reifegrade. Dabei werden u. a. Testprozessmanagement, Berichtswesen, Kommunikation, Metriken und methodisches Vorgehen bewertet. Eine der Kriterien bei dieser Bewertung ist die Testorganisation, auf die im Folgenden näher eingegangen wird.

Der Begriff der Testorganisation im Sinne von TPI Next umfasst alle Testfunktionen, Einrichtungen, Abläufe, Aktivitäten, Verantwortlichkeiten und Befugnisse sowie deren jeweilige Beziehung untereinander. Eine Testorganisation setzt Mittel und Wege ein, um Testprodukte und/oder Testdienstleistungen anzubieten. Sie verfolgt gemeinsame Ziele, kontrolliert ihre eigene Leistung und trägt die Verantwortung für ihre eigenen Ergebnisse.

Um sich schnell an veränderte Gegebenheiten anpassen zu können, muss die Testorganisation flexibel aufgebaut und an den Testprozess angepasst sein. Das ist vor allem dann erforderlich, wenn zusätzliche Projekte anstehen oder eine innerbetriebliche Umorganisation durchgeführt wird. Eine Testorganisation bietet verschiedene Produkte und Dienstleistungen und ist in verschiedenen Reifegraden ausgeprägt.

28.3 Einteilung der Reifegrade

Jeder Testprozess kann in einem bestimmten Reifegrad eingestuft werden.

Beginnend beim Reifegrad „Initial“ kann sich ein Testprozess über die Reifegrade „Kontrolliert“ über „Effizient“ bis hin zu „Optimierend“, dem höchsten Reifegrad, entwickeln. Mit Erreichen eines höheren Grads nimmt die Reife des Testprozesses zu und der geschäftliche Mehrwert steigt.

- **Initial:** Der Testprozess besteht aus unzulänglich dokumentierten Ad-hoc-Aktivitäten. Der Blick ist darauf gerichtet, ob eine Software überhaupt funktioniert und nicht darauf, ob die Anforderungen und Spezifikationen eingehalten wurden. Ein solcher Testprozess bietet nur unzureichende oder unsystematische Informationen über bestehende Risiken und die Qualität des Testprozesses selbst.
- **Kontrolliert:** Alle wichtigen Tätigkeiten des Testprozesses werden geplant und durchgeführt. Der Prozess wird kontrolliert und gesteuert. Der Testprozess erlaubt dadurch einen früheren und besseren Einblick in die Qualität und lässt rechtzeitige Korrekturmaßnahmen zu. Das Risiko von Verzögerungen aufgrund mangelhafter Produktqualität wird dadurch verringert.
- **Effizient:** Die Aktivitäten des Testprozesses werden derart aufeinander abgestimmt, dass es zu einem günstigeren Kosten-Nutzen-Verhältnis kommt. Die Art der Testdurchführung wird verbessert und die kritischen Fehler werden bei minimalen Kosten und einem minimalen Zeitaufwand ermittelt.
- **Optimierend:** Der erreichte Zustand des Testprozesses wird gepflegt, analysiert und stetig den sich ändernden Erfordernissen angepasst. Das Niveau der Testaktivitäten wird auch in Zukunft gehalten. Kontrollpunkte des Reifegrads „Optimierend“ enthalten die Maßnahmen, die notwendig sind, um die Prozessqualität auch bei sich verändernden Situationen und Rahmenbedingungen stabil zu halten.

28.4 Reifeprozess für die Testorganisation

Zur Bewertung des Reifeprozesses werden einzelne Kriterien bewertet:

- Erfahrung der Tester, Ausstattung mit Testkompetenz und spezifischer Expertise, z. B. im Anwendungsbereich oder im testspezifischen Wissen:
 - Unterstützung der Produktrisikoaanalyse,
 - Entwicklung einer Testvorgehensweise für das gesamte Projekt (Mastertestplan),
 - Erfahrung mit Testaufwandsschätzungen,
 - Koordination von Testprojekten und
 - Testautomatisierung.

- Beschreibung des Testprozesses, Vorlagen und Schulungsunterlagen für strukturiertes Testen.
- Testwerkzeuge, Werkzeuglizenzen und Skripte.
- Kommunikations- und Berichtswesenstrukturen.
- Grundsätzliche Verfahren für das Fehlermanagement.
- Testprozessverbesserung.

Viele der aufgeführten Testprodukte und Testdienstleistungen sind in gewissem Umfang auch für andere Kernbereiche von Bedeutung. Die Testorganisation stellt dadurch die Verbindung zwischen den einzelnen Kernbereichen her.

28.5 Reifegrad „Initial“

Die am Testen beteiligten Personen sind über größtenteils fachlich organisierte Abteilungen verteilt. Ihre Rollen betreffen meist spezifische Teilaufgaben. Gemeinsame Ansichten zu Test und Qualitätssicherung sind kaum vorhanden. Die Weitergabe von Informationen ist mangelhaft, die Ergebnisse der Testaktivitäten werden kaum verglichen und die Personen sind eher zufällig zusammengestellt und widmen nur einen nicht exakt definierten Teil ihrer Arbeitszeit dem Test von Systemen.

28.6 Reifegrad „Kontrolliert“

Innerhalb des Softwarelebenszyklus können unterschiedliche Testaktivitäten und die zugehörigen Verantwortlichkeiten identifiziert werden. Entwurf, Entwicklung und Wartung sind zu einem gewissen Umfang organisiert und als spezifische Funktionen anerkannt. Der Test ist zu einem gewissen Grad organisiert, jedoch stark vom Reifegrad der anderen Prozesse und deren Organisation abhängig.

Kontrollpunkte:

- Alle Beteiligten wissen, wer bzw. welche Abteilung für die Testdienstleistungen Verantwortung trägt.
- Innerhalb der Testorganisation ist eine Struktur der Kontrolle und Verantwortlichkeit vorhanden.
- Testaufgaben und Verantwortlichkeiten sind definiert und dokumentiert und einer Person oder Organisationseinheit zugewiesen.
- Die Produkte und Dienstleistungen der Testorganisation sind deren Kunden bekannt.

Enabler:

- Im Unternehmen ist ein Wissensmanagement etabliert, das die Testorganisation für die Wissenskonsolidierung nutzen kann.

Verbesserungsvorschläge:

- In einem kleinen Unternehmen sollte eine für das Testen verantwortliche Person benannt werden, die als Ansprechpartner für alle testbezogenen Fragen fungiert.
- In einer größeren Organisationseinheit kann, basierend auf Art der etablierten Testorganisation, über die bestmögliche (örtliche) Platzierung des Testteams entschieden werden.
- Der Austausch von Wissen, Erfahrungen und Best Practices zwischen Testern und anderen Personen, die Testaufgaben wahrnehmen, wird durch regelmäßige Workshops und Besprechungen organisiert.
- Die Stakeholder erfahren die Notwendigkeit verschiedener Rollen und Verantwortlichkeiten für die Beauftragung von Tests, Produktabnahme und Testdurchführung zu unterscheiden.
- Produkte und Dienstleistungen der Testorganisation sind dokumentiert (z. B. „Erstellen der Testplanung“, „Ausführung einer Teststufe“)
- Ein Marketing- und Kommunikationsplan zur Öffentlichkeitsarbeit (Was verkaufen wir? Wen spricht unser Produkt an?) liegt vor.

28.7 Reifegrad „Effizient“

Am Testprozess können unterschiedliche Abteilungen und Teams beteiligt sein, die teilweise gar nicht wissen, welche Testaktivitäten an anderen Stellen im Unternehmen ausgeführt werden. Sie können eine völlig andere Arbeitsweise haben und unterschiedliche Werkzeuge verwenden. Um zum Reifegrad „Effizient“ zu gelangen, sollte man die separaten Aktivitäten ganzheitlich koordinieren.

In einer effizienten Testorganisation werden den Projekten Testressourcen in Form von Dienstleistungen zur Verfügung gestellt, die folgende Aktivitäten abdecken:

- Durchführung,
- Support (Coaching und Schulungen),
- Kontrolle (Audit und Review),
- Wartung sowie
- Forschung und Entwicklung.

Testdienstleistungen können auch von externen Dienstleistern zur Verfügung gestellt werden. Bei Outsourcing kann die Testverantwortung ganz oder teilweise extern übernom-

men werden. Bei einem geänderten Ressourcenbedarf der einzelnen Projekte findet eine Neuverteilung der Ressourcen auf die einzelnen Projekte statt.

Auf diesem Reifegrad ist die Testorganisation stärker als im Reifegrad „Kontrolliert“ auf die tatsächlichen Anforderungen eines Kunden ausgerichtet. Dadurch stehen weniger die Aktivitäten als die Resultate im Vordergrund. Die Testaktivitäten werden dahingehend identifiziert, organisiert und überwacht.

Um Umfang und Qualität der Testergebnisse besser planen zu können, werden bestimmte Verfahren definiert und durch Schulungen vermittelt. Der Bedarf an Werkzeugen und anderen Hilfsmitteln wird analysiert und entsprechend abgedeckt.

Die Teststrategie der Organisationseinheit wird von der Unternehmensführung und der Testorganisation definiert und in einer Testrichtlinie dokumentiert. Die Unternehmensführung achtet auf die Einhaltung und Umsetzung dieser Testrichtlinie.

Kontrollpunkte:

- Verschiedene Personen oder Abteilungen, die Testdienstleistungen zur Verfügung stellen, stimmen die Organisation ihrer Arbeit untereinander ab.
- Die Testorganisation stellt den Projekten die vereinbarten Testressourcen und Testdienstleistungen zur Verfügung.
- Die Entscheidung, wo und wie die Testdurchführung positioniert werden soll, ist gut durchdacht.
- Die Testrichtlinie wird befolgt.

Enabler:

Organisationseinheiten, die Dienstleistungen in den Bereichen Personal, Entwicklungsumgebungen, Werkzeuge und Prozesse anbieten, können auch Dienstleistungen und Ressourcen im Bereich Testen zur Verfügung stellen.

Vorschläge zur Optimierung:

Eine Untersuchung des Bedarfs der verschiedenen Testprojekte wird durchgeführt. Der Bedarf wird nach Arten von Ressourcen und Arten von Dienstleistungen pro Ressource kategorisiert. Ein Leistungsverzeichnis mit verschiedenen Arten von Ressourcen und Dienstleistungen wird zusammengestellt. Auf Grundlage eines Business Cases wird entschieden, welche Dienstleistungen die Testorganisation zur Verfügung stellen soll. Eine permanente Verbesserung der Testorganisation wird anhand konkreter Maßnahmen und Ziele aufgezeigt.

28.8 Reifegrad „Optimierend“

Verbesserung und Innovation und Beziehungen zu Gruppen und Personen außerhalb der Testorganisation sind von großer Bedeutung. Die Testorganisation funktioniert auf allen Stufen und bei den wichtigsten Prozessen.

Auf dem Reifegrad „Optimierend“ führt die Konzentration auf die interne und externe Qualität zu besseren Ergebnissen. Hierzu zählt auch die Pflege des Leistungsverzeichnisses, insofern als es den Anpassungsbedarf des Angebots an die Nachfrage widerspiegelt. Die Testorganisation konzentriert sich darauf, bei Bedarf das geeignete Personal mit entsprechender Erfahrung und Expertise zur Verfügung zu stellen.

Die Testorganisation verbessert sich kontinuierlich selbst, indem interne Prozesse bewertet und angepasst werden. Es werden Vergleiche mit externen Dienstleistern durchgeführt und zur Unterstützung verwendet, wenn sie bessere Leistungen anbieten können.

Kontrollpunkte:

- Die Produkte und Dienstleistungen der Testorganisation werden regelmäßig bewertet. Neue Dienstleistungen werden hinzugefügt, wenn sie wirtschaftlich sind.
- Die Testorganisation trägt Verantwortung für das Gelingen und Misslingen von Testaufträgen.
- Die Leistung der Testorganisation wird regelmäßig mit der Leistung externer Dienstleister oder ähnlichen Testorganisationen verglichen.

Enabler:

Eine Testorganisation kann ihre Leistung steigern, wenn die in Wechselbeziehung zueinanderstehenden Teile des Unternehmens den gleichen Reifegrad haben und proaktiv zusammenarbeiten.

Optimierungsvorschläge:

- Die Nachfrage nach den angebotenen Dienstleistungen wird beobachtet und regelmäßig an Bedarfsänderungen angepasst.
- In regelmäßigen Abständen wird geprüft, ob eine Nachfrage nach neuen Testdienstleistungen besteht und entschieden, ob diese Dienstleistungen in das Leistungsverzeichnis aufgenommen werden sollen.
- Die Kosten der Dienstleistungen werden überprüft und mit externen Dienstleistern und/oder Testorganisationen ähnlicher Reifegrade verglichen.
- Die internen Prozesse der Testorganisation werden überprüft und nach Möglichkeit optimiert. [TPIN2011].

28.9 Unterschiede TMMi und TPI

Beim Vergleich beider Methodenbaukästen fällt auf: TMMi hat Vorteile, wenn ein Unternehmen bereits für die Verbesserung der Softwareentwicklung insgesamt auf CMMi setzt. Die Testmanager erhalten so eine 360-Grad-Sicht auf die IT-Organisation, die Softwareentwicklung und Testing gleichermaßen abdeckt.

Bei der Verbesserung der Prozesse lassen sich so Synergien und eine erweiterte Sicht auf sämtliche Abläufe erreichen. Wenn eine Testorganisation beispielsweise nach TMMi

die Ebene 3 erreicht, ist das vergleichbar mit Level 2 im Bereich „Verification and Validation“ bei CMMi.

Obwohl TMMi schon sehr ins Detail geht, deckt es Bereiche nicht ab, die TPI berücksichtigt. Dazu gehören das **Office Environment**, **Reporting** und **Testware Management**. Alle drei Bereiche sind allerdings wichtige Bestandteile im Testprozess. Das verschafft TPI einen gewissen Vorteil gegenüber der TMMi-Methode. Ein weiterer Unterschied beider Ansätze ist die Abstufung zwischen den Reifegraden. Jeder Kernbereich bei TMMi erfordert einen bestimmten Reifegrad. Das bedeutet: Um Level 3 zu erreichen, müssen unter anderem alle Anforderungen der Key Area „Test Training“ erfüllt sein. [CIOD2019].

Literatur

- [TPIN2011] : Sogeti: TPI Next – Geschäftsbasierte Verbesserung des Testprozesses, dpunkt Verlag Heidelberg 2011
- [CIOD2019] : <https://www.cio.de/a/test-methoden-tmmi-und-tpi-im-vergleich,2301883,3>, zugegriffen am 09.05.2020
- [HETT2019] : <https://www.hettwer-beratung.de/konzepte/testkonzept/testrollen/>, zugegriffen am 09.05.2020

Besonderheiten der Testorganisation in agilen Projekten

29

Zusammenfassung

Das Testmanagement in agilen Projekten muss die stärkere Dynamik der Testprozesse berücksichtigen. Organisationsformen, Rollen und für die Agilisierung nötige Methoden wirken sich auch auf den Softwaretest aus. Für das agile Testen sind daher besondere Grundsätze zu beachten.

29.1 Agile Projekte

Handelt es sich bei der Aufgabenstellung zur Testplanung und Testorganisation um ein **agiles Projekt**, nehmen die Herausforderungen für das Testmanagement nochmals deutlich zu. Parallel ablaufende Prozesse und Änderungen der Funktionalität innerhalb eines Zeitfensters führen zu erheblich stärkerer Dynamik.

Effizientere Kommunikation zwischen allen Projektbeteiligten, optimierte Time-to-Market, Beherrschung der steigenden Komplexität und vor allem Sicherstellung der geforderten Qualität sind wesentliche Gründe zur Agilisierung von Prozessen, die sich auf die Testplanung und Testorganisation auswirken.

Statt am Ende des Projektes ein finales Resultat zu präsentieren, beziehen agile Teams ihre Kunden in den Prozess mit ein und liefern ihnen regelmäßig neue Softwareversionen, die diese bewerten können. Es reicht aber nicht allein aus, Kundenwünsche frühzeitig zu erkennen – eine effiziente Umsetzung ist mindestens genauso wichtig. Die Software ist damit häufigen und oft auch tief greifenden Änderungen unterworfen. Die Vorteile liegen auf der Hand: kurze Entwicklungszyklen mit schnellen Realisierungserfolgen.

Doch um mit dem hohen Tempo mithalten zu können, muss sich das Testing den agilen Arbeitsweisen anpassen, da es mittlerweile zu einem unverzichtbaren Bestandteil agiler Softwareentwicklung geworden ist. Längst wird nicht mehr darüber diskutiert, ob agile

Methoden eingesetzt werden sollen, sondern wie sie richtig eingesetzt und gegebenenfalls skaliert werden können.

29.2 Agile Methoden

Der Test ist also untrennbar mit der Entwicklung verwoben. Eine der bekanntesten agilen Methoden ist **Scrum**, das die Prinzipien des agilen Manifests umsetzt. Allerdings werden sowohl im agilen Manifest als auch in der Scrum-Beschreibung weder Testaktivitäten noch die Testorganisation tiefergehend behandelt. Es werden zwar Ansätze des **Test Driven Developments** oder der **Continuous Integration** empfohlen, doch ein systematisches Testvorgehen wird nicht beschrieben. Es mag zwar Tester geben, also Spezialisten mit dem Schwerpunkt Qualitätssicherung und Akzeptanztest, aber eine separate Testschiene kommt in den meisten Vorgangsmodellen nicht vor. **Kanban**, die dem agilen Vorgehen zugrunde liegende Methode in der Softwareentwicklung, lässt dies bewusst offen. Die Testaktivitäten und die Testorganisation spielen aber eine entscheidende Rolle für den Erfolg agiler Methoden. Das Testing in agilen Entwicklungsprojekten unterscheidet sich von klassischen Methoden in erster Linie dadurch, dass es eine präventive Maßnahme ist und dass die Tests in sogenannten **Sprints** viel häufiger ausgeführt werden müssen als im klassischen Testmodell (wie z. B. im V-Modell). In der Regel beträgt die Dauer eines Sprints ein bis zwei Wochen, an dessen Ende die getestete Software stehen soll, die anschließend an den Kunden ausgeliefert wird. In regelmäßigen Retrospektiven werden die Erfolge und Probleme festgehalten und evaluiert, was die Qualität sowie die erzielte Testabdeckung weiter erhöht.

Für agile Teams hat sich aber inzwischen ein Modell durchgesetzt, dass die verschiedenen Testperspektiven etwas vereinfachend, aber umso wirkungsvoller darstellt. Die sogenannten vier Testquadranten von Crispin & Gregory (Abb. 29.1) teilen die Testaufgabe in eine eher IT-technische und eine eher fachliche Perspektive bei den Tests. Außerdem werden Tests unterschieden, je nachdem, was man im Vorhinein schon ausgearbeitet hat („**supporting the team**“ – Team unterstützend) und auf welche Probleme man erst im Laufe der Fertigstellung stößt („**critique the product**“ – Produkt hinterfragend).

Damit lassen sich ebenfalls verschiedene Testarten (z. B. Unittest, Benutzbarkeitstest) unterteilen, für die es ggf. verschiedene Mitarbeitergruppen gibt, die sich jeweils darum kümmern. Gerade bei Last-/Performancetests, Sicherheitstests und Zuverlässigkeitstests sind oft Hilfestellungen von Partnern außerhalb des Projektteams nötig.

Der Regressionstest wird hierbei nicht betrachtet, weil er kein eigener Test ist. Der Regressionstest ist immer Teil eines Tests und sollte bei jedem Test mitgedacht werden, auch wenn man ihn evtl. organisatorisch (z. B. Nutzung von Offshore-Mitarbeitern) oder technisch (durch automatisierte Tests) auslagert.

Wenn man die Perspektiven des Tests und damit ggf. auch die Arbeitsteilung im Test geklärt hat, sollte man sich vor Augen führen, was überhaupt der Testgegenstand ist.

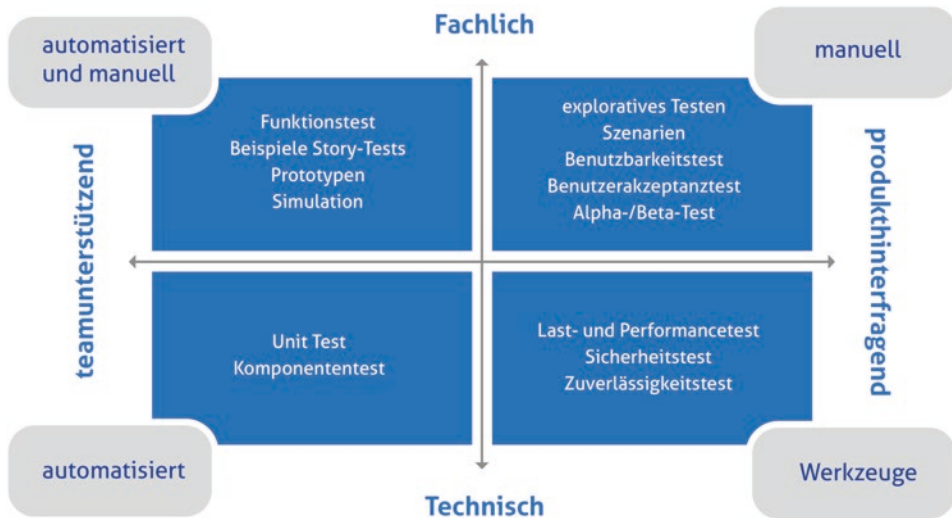


Abb. 29.1 Testquadranten als Modell für Testaktivitäten in agilen Projekten

Zum einen sind in diesem Zusammenhang die Funktionen/Features (oder wie sie jeweils im Projekt genannt werden) zu nennen. Hier ist genau zu unterscheiden, welche Funktionen zu testen sind und welche Funktionen nicht. Manchmal ist eine Funktion besser dadurch abzugrenzen, dass man ausschließt, was nicht zum Funktionsumfang gehört. Bei einem Webshop kann man beispielsweise gerne die Abbruchfunktion für das Einkaufen versprechen, aber es macht die Sache doch für den Auftraggeber klarer, wenn man einschränkt, dass der Kunde dann (noch) nicht in der Lage ist, mit demselben Warenkorb erneut wieder zur Kasse zu gehen.

Zum anderen sollte man aber neben den einzelnen Features die nichtfunktionalen Qualitätseigenschaften betrachten. Zuverlässigkeit, Sicherheit und Benutzbarkeit sind bei IT-Projekten zentrale Kriterien. Hier empfiehlt sich ein Blick auf das Qualitätsmodell der **ISO 25010**, welches man als Checkliste der Reihe nach durchgehen sollte [PERF2017].

Für den Erfolg eines Projektes ist es unerlässlich, dass alle für ein Projekt relevanten Testverfahren (Unit-, Funktions- und Last-Performancetests) früh eingeplant und auf allen Ebenen umgesetzt werden. Das unterstützt den Tester dabei, Probleme rechtzeitig zu erkennen und zu beheben. Durch die Automatisierung von Tests ist es möglich, diese mehrmals täglich oder in regelmäßigen Testläufen nachts und am Wochenende durchführen zu lassen. Regressionstests sorgen dafür, dass neue Funktionen nicht bereits bestehende Funktionen beeinträchtigen. Da die automatisierten Tests nach jeder Iteration an das überarbeitete digitale Produkt angepasst werden müssen, muss laufend abgewogen werden, wie effizient die Methode ist.

Zusätzlich zur Durchführung von Testfällen, automatisiert oder nicht, ist es wichtig, diese mit explorativen Tests, bei denen die Software ohne vorgegebenen Testplan geprüft wird, zu kombinieren. Menschlicher Scharfsinn und Kreativität sind nötig, um sowohl

oberflächliche als auch tief gehende Fehler zu finden, die vorab nicht im Labor planbar sind. Die Qualität der Testaktivitäten entscheidet letztlich über die Qualität des Endproduktes.

29.3 Organisation und Ziele agiler Projekte

Im Mittelpunkt der Teamorganisation steht der **Product Owner**, der das große Ganze im Blick behält, Ziele definiert und die fachliche Vertretung des Kunden ist. Neben dem Product Owner gibt es die einzelnen Testteams. Die funktionsübergreifende Organisation und Zusammenarbeit gibt agilen Entwicklungsteams die Chance, freier zu arbeiten und auch mal über den Tellerrand hinauszuschauen. Entwickler und Product Owner sprechen regelmäßig miteinander und tauschen sich direkt mit den Testverantwortlichen aus. Dies ermöglicht bei Bedarf schnelle Anpassungen und Reaktionen.

Die Festlegung der Ziele gibt den Teams eine klare Vorstellung davon, woran sie arbeiten. Nur wenn diese klar definiert sind, können die Teams effizient testen und die einzelnen Sprints erfolgreich abschließen. Obwohl es sich beim Testen um Qualitätssicherung handelt, können die Tester niemals alle Fehler finden. Das eigentliche Ziel der Testmaßnahmen sollte es daher sein, die Software laufend zu verbessern. Darüber hinaus können die Ziele zu ambitioniert sein, wenn Entwickler zu viele Funktionen implementieren möchten. Die Folge: Die Ziele sind nicht aus Nutzersicht gedacht und die Teams testen am Kern der Probleme vorbei. Was nutzt ein Angebot, das keiner wahrnimmt? Es ist daher elementar, die eigenen User zu kennen und beispielsweise auf Geräten zu testen, die auch die Zielgruppe verwendet. Wie hoch ist die CPU-Auslastung, wie viel Datenvolumen verbraucht die App und welche Ladezeiten sind akzeptabel? Diese und weitere Faktoren sollten während der Testdurchführung unbedingt geprüft werden.

Hier kommen Methoden wie **Crowdsourced Testing** oder **Crowdtesting** ins Spiel. Diese Art der Qualitätssicherung einer Website, einer mobilen App oder eines Connected Device stützt sich für die Testdurchführung auf eine Community aus Testern außerhalb des Unternehmens. Mit Crowdtesting werden Produkte nicht in einer sterilen Laborumgebung getestet, sondern durch echte Personen, auf echten Geräten und in gängigen Alltagssituationen auf der ganzen Welt. Die Ergebnisse reflektieren die tatsächliche oder mögliche Nutzererfahrung wesentlich besser, als dies durch Inhouse- oder Laborlösungen möglich wäre. Ein weiterer Vorteil dieser Methode: Sie ist kompatibel mit der Geschwindigkeit agiler Prozesse, da viele Tester zur gleichen Zeit testen und so hunderte Teststunden innerhalb eines einzigen Tages generieren können. Problematisch ist dabei allerdings die Fehlererfassung, da die Fehler meist nicht so exakt und umfassend beschrieben sind wie bei erfahrenen, ausgebildeten Testern und die Fehler oft mehrfach von unterschiedlichen Testern erfasst werden. Das bedeutet ein umfassenderes Defect-Management, mit mehr Rückfragen und Eliminieren doppelter Fehlereinträge. Die Organisation und die Arbeitsvorbereitung können dabei ebenfalls kompliziert und aufwändig werden. Dazu kommt, dass sich dieses Vorgehen in aller Regel nur eignet, wenn es sich um eine Internetapplikation

oder eine App handelt, die einen breiten Kundenkreis anspricht (also etwa eine Online-Shopping-Plattform), die Methode ist aber tendenziell ungeeignet bei sehr spezialisierten Anwendungen, die eine spezielle Infrastruktur voraussetzen.

29.4 Agile Projekte und traditionelle Organisation

Testdatenmanagement wird häufig unterschätzt, ist aber essenziell. Die Tester fördern wertvolle Informationen und Erkenntnisse zutage. Wie viele Testfälle wurden durchgeführt, wie hoch ist die Fehlerrate usw.? Alles Fragen, die Material für ausführliche und detaillierte Berichte hergeben und mit Sicherheit von hoher Relevanz für den Projektleiter und den Produktmanager sind. Die Frage ist aber: Wen könnten diese Daten darüber hinaus noch interessieren? Denn im Kern dreht sich bei agilen Methoden alles darum, Arbeitsabläufe zu optimieren und nur so viel Zeit wie unbedingt nötig für Reportings aufzuwenden. Es ist daher ratsam, Daten zu sammeln, die auch für andere Abteilungen nützlich sind. Die Aufbereitung der Daten in Form von Best-Practices bzw. Cases kann dabei helfen, die Leistungen der Teams zu steigern und die Entwicklung der Software zu verbessern.

Derartig umfassende Veränderungen stellen Unternehmen – insbesondere das Management – natürlich vor Herausforderungen, die es zu meistern gilt: Die Teams stoßen immer wieder auf Widerstände an den Schnittstellen zur traditionellen Organisation. Eine segmentierte Planungsweise, der erhöhte Abstimmungsaufwand oder die reduzierte Dokumentation sorgen nicht selten für Unsicherheiten. Um dem entgegenzuwirken, bedarf es der vollen Rückendeckung durch das Management, eindeutige Festlegung von Entscheidungskompetenzen, Zeitrahmen und Budgets sowie der Klärung der Schnittstellen mit Funktionseinheiten außerhalb des agilen Prozesses [ENTW2019].

29.5 Notwendige Voraussetzungen zur Testorganisation in agilen Projekten

Bei der Testorganisation muss man in agilen Projekten folgende Punkte beachten:

- Man muss sich einen Überblick über das Testobjekt, die Testziele und die Abgrenzung zu den Schnittstellen (Umfang) definieren.
- Die Funktionen des Testobjektes müssen in einer geeigneten Struktur abgebildet werden.
- Für jede Komponente muss zu jedem Zeitpunkt die jeweilige Testabdeckung erkennbar sein.
- Jedes einzelne Arbeitspaket wird der verbleibenden Zeit gegenübergestellt und die entsprechenden Anpassungen für Zeit, Geld und Leistung durchgeführt.

In der Praxis empfiehlt sich zur Umsetzung ein Dokument mit der Zusammenstellung testrelevanter Informationen, die in funktionelle Komponenten gegliedert ist. Folgende Komponenten sollten im Dokument enthalten sein:

- Grafik der gewünschten Darstellung,
- Akzeptanzkriterien,
- Bemerkungen aus Besprechungen,
- testrelevante Umsetzungen mit Verweis auf zugehörige Testfälle,
- Berechtigungen der Rollen sowie
- weitere relevante Informationen.

Ziel des Arbeitsdokuments ist es, jederzeit einen Überblick über den aktuellen Status der Testaktivitäten im agilen Team zu erhalten. Dadurch kann man auf Änderungen im agilen Projekt mit kurzen Reaktionszeiten gezielt reagieren und notwendige Anpassungen kurzfristig identifizieren. Die Komponente soll schnell auffindbar, angepasst und die betroffenen Testfälle ersichtlich sein. Diese Testfälle können dann entsprechend geändert, erweitert oder entfernt werden. Testfälle, die zusätzlich erstellt werden sollten, können leicht ermittelt werden [BLDS2019].

29.6 Grundsätze für agiles Testen

Folgende Grundsätze sollten beim agilen Testen beachtet werden:

- Einplanung von Testaktivitäten, Qualitätssicherung als Kernaufgabe: Oft sind in agilen Teams Entwicklungsarbeiten verzögert, was dazu führt, dass dem Test nicht die nötige Aufmerksamkeit gewidmet wird. Im agilen Team sind nicht immer Testexperten vorhanden. Nur regelmäßiges, frühes Testen bringt aber die Vorteile kurzer Feedbackschleifen in Scrum voll zum Tragen.
- Methodische Testfallerstellung: Unittests sollten am meisten, komplexe End-to-End- oder Systemtests am wenigsten vertreten sein. Gerade bei komplexen Anwendungen und Sachverhalten liegen oft zu viele Variationen vor, um alle Fälle zeitgerecht untersuchen zu können. Die sinnvolle Auswahl und ausreichende Testabdeckung sind daher besonders zu beachten.
- Koordination der Testaktivitäten: Gerade im Kontext von Scrum und interdisziplinären Teams ist es von Vorteil, Teams mit qualifizierten Testexperten zu durchmischen. Dabei sollte, je nach Arbeitsgegenstand und Anzahl der Teams, ein Testmanager benannt werden, der die teaminterne aber auch teamsübergreifende Koordination übernimmt. Es sollte in jedem Fall darauf geachtet werden, Test-Know-how langfristig in allen Teams zu manifestieren. Zertifizierungen im ISTQB-Bereich sind in diesem Zusammenhang ebenfalls zu empfehlen. In agilen Teams wird die Qualitätssicherung vor allem dann

problematisch, wenn die Entwicklung eine mächtigere Position als der Softwaretest einnimmt und das Thema Test nur als Anhängsel betrachtet wird.

- Agile Testcharta: Die **Testcharta** ist ähnlich eines Testkonzeptes zu betrachten, sie wird aber iterativ während des Projektes überarbeitet. Es ist auch möglich, Aspekte des Testens in die **Definition of Done** einfließen zu lassen, z. B. zu jedem **Product Backlog Item** (häufig: User Story) liegt ein automatisierter Testfall vor. Folgende Punkte sollten in der Testcharta vorhanden sein: Verteilung der Testfälle auf Teststufen, Anwendung von Methodik und Metrik, Nutzung von Testumgebungen, Herleitung und Umgang mit Testdaten.
- Einsatz von Testautomatisierung: Die Erstellung automatisierter Testskripte sollte von Anfang an, beginnend mit dem Unittest, fokussiert werden. Nur durch sukzessives Erstellen und Pflege geeigneter automatischer Testfälle (z. B. auch durch Überführen manueller Tests) ist eine kontinuierliche Auslieferung des Produktes im Sprintrhythmus (Continuous Delivery & Integration) möglich. Testautomatisierung ist eine notwendige Voraussetzung, um Projekte erfolgreich agilisieren zu können [SCRU2019].

In manchen Teststufen, wie Unittest oder Integrationstest, treten die Vorteile der agilen Methoden stärker hervor als beim Systemtest. Vor allem, wenn die Anforderungen erst im Lauf der Entwicklung herausgearbeitet werden, sind agile Methoden den klassischen Methoden überlegen.

Daher gibt es auch Mischformen, bei denen auf den höheren Teststufen nach wie vor nach V-Modell und bei den entwicklungsnahe Teststufen nach agilen Methoden getestet wird. Das bewirkt eine bessere Systematik der Testdurchführung. Diese Mischformen bedeuten aber einen besonders hohen Bedarf für die Kommunikation unter den Stakeholdern und die Organisation von Testprozessen.

Literatur

- [BLDS2019] : <https://blog.doubleslash.de/testmanagement-im-agilen-projekt-ein-erfahrungsbericht/>,
zugegriffen am 09.05.2020
- [SCRU2019] : <https://www.scrum.de/5-tipps-fur-agiles-testen/>, zugegriffen am 09.05.2020
- [ENTW2019] : <https://entwickler.de/online/agile/agiles-testing-579801504.html>, zugegriffen am
09.05.2020
- [PERF2017] : <https://www.informatik-aktuell.de/entwicklung/methoden/das-perfekte-testkonzept-in-6-schritten.html>, zugegriffen am 09.05.2020

Zusammenfassung

Neue Trends, wie z. B. künstliche Intelligenz und der Einsatz kognitiver Technologien, werden den Softwaretest verstärkt beeinflussen und zu neuen Testverfahren und veränderten Prozessen führen.

30.1 Künstliche Intelligenz

Künstliche Intelligenz (KI) ist ein Trendthema, das in den kommenden Jahren einen wesentlichen Umbruch im Alltag und bei der täglichen Arbeit nach sich ziehen wird. Künstliche Intelligenz weist bereits viele kognitive Eigenschaften aus. Kognition aber hat sich der Mensch vor allem sich selbst zugesprochen, sodass künstliche Intelligenz zuerst als Konkurrenz für die menschliche Arbeit betrachtet wird. Die neuen Trends der Nutzung kognitiver Technologien werden sich auch auf den Softwaretest auswirken.

Durch den Einsatz künstlicher Intelligenz wird es möglich sein, komplexer werdende Softwaresysteme in kürzerer Zeit zu verifizieren. Außerdem wird eine Überwachung der produktiven Software in Echtzeit und Prognosen über zukünftige Veränderungen und Abweichungen möglich, bevor es zu Fehlern kommt. Weiterhin sind Analysen großer Datenmengen möglich, um beispielsweise Fehlertaxonomien zu identifizieren oder die Testabdeckung zu steigern.

Viele Werkzeuge bieten bereits heute einen automatisierten visuellen Abgleich der Benutzeroberfläche und können somit für den Akzeptanztest verwendet werden. Einige wenige Tools sind bereits dazu in der Lage, nach einem Training Testfälle autonom zu entwerfen, durchzuführen und zu pflegen. Weiterhin überwachen die meisten KI-gestützten Testwerkzeuge Änderungen der Performance der Software (z. B. Laufzeiten, Speicherauslastung) und prüfen regelmäßig die Informationssicherheit. Aufgrund historischer Daten

und einem gezielten Training können diese Werkzeuge frühzeitig Warnhinweise geben, bevor es zu Einschränkungen in der Qualität kommt. Dadurch kann man die Fehlerbehebung noch früher einsetzen lassen bzw. durch Erkennung spezieller Muster schon im Vorfeld erkennen, bevor sich Fehler betrieblich auswirken.

Nach risikobasiertem und agilem Testen bedeutet KI erneut ein Umdenken für Softwaretester. Zukünftig wird die Zusammenarbeit mit KI-gestützten Werkzeugen noch erheblich wichtiger werden. Dazu müssen diese Systeme aber mit sinnvollen Daten, z. B. einer ausreichenden Anzahl Bilder zu einem vorgegebenen Begriff, trainiert werden. Ein grundlegendes Verständnis der mathematischen und algorithmischen Grundlagen ist daher eine wesentliche Voraussetzung. Softwaretests werden somit wieder ein Stück technologieorientierter. Künstliche Intelligenz übernimmt wesentliche Aufgaben des Softwaretests, beispielhaft die Testdurchführung oder den Entwurf der Testfälle. Sie wird aber auf absehbare Zeit nicht den Menschen als Softwaretester vollständig ersetzen können. Wie immer in der Geschichte werden bestimmte, eher reproduktive Tätigkeiten wegfallen, andere, kreative und geistig anspruchsvolle Aufgaben, den Softwaretest prägen und die Notwendigkeit organisierter und gut konzipierter Testaktivitäten und durchdachter Testprozesse eher steigern [SQMA2018]. KI wird nicht dazu führen, dass Tester überflüssig werden. Berufsbilder wandeln sich im Laufe der Zeit, aber genauso wie gewisse Tätigkeiten in Zukunft nicht mehr nötig sind oder automatisiert werden können, werden neue berufliche Herausforderungen entstehen, die heute noch gar nicht genau vorhersehbar sind.

Ist von KI die Rede, gibt es mitunter wilde Assoziationen wie etwa allmächtige Supercomputer, die wild entschlossen sind, die Menschheit zu zerstören, sprachgesteuerte Assistenten in der Art von Alexa oder Siri, Computer als Schachgegner oder selbstfahrende Autos. Die Schlagworte werden meist in politischem Kontext verwendet, ohne sie genau zu definieren und um teilweise Science-Fiction-Szenarien aufzuzeigen, die mit der Realität sehr wenig zu tun haben.

30.2 Anwendungsbereiche künstlicher Intelligenz

Wichtig ist auch sich klarzumachen, dass sich der Umfang dessen, was man unter KI versteht, mit der Zeit verändert. Beispielsweise gab es eine Zeit, in der die optische Zeichenerkennung (Optical Character Recognition, OCR) als der Stand der Technik im Bereich der KI angesehen wurde. Auch die Fragen und Antworten von Siri und Alexa galten einmal als der neueste Stand der Technik, werden inzwischen aber weitgehend als Selbstverständlichkeit hingenommen und nicht in jedem Fall als KI betrachtet. Dasselbe wird bei den Softwaretesttools passieren: Heutige Innovationen im Bereich der Automatisierung werden zur Selbstverständlichkeit, sobald neue Fähigkeiten bereitstehen.

Noch steckt die Nutzung von künstlicher Intelligenz in der Softwareentwicklung in den Kinderschuhen, der **Autonomiegrad** ist hier noch weit niedriger als auf weiter entwickelten Gebieten wie den selbstfahrenden Systemen oder der Sprachsteuerung. Dennoch zielt die Entwicklung klar in Richtung des autonomen Testens. Der KI-Einsatz bei den

Software-Testwerkzeugen konzentriert sich darauf, den Softwareentwicklungs-Lebenszyklus einfacher zu gestalten. Durch die Anwendung von Schlussfolgerung, Problemlösung und in einigen Fällen auch maschinellem Lernen lässt sich KI nutzen, um bei der Automatisierung zu helfen und den Aufwand an mühseligen Routinearbeiten beim Entwickeln und Testen von Software zu verringern. An dieser Stelle wäre die Frage angebracht, ob Testautomatisierungstools das nicht bereits tun. Die Antwort ist ein klares Ja, wenn auch mit ein paar Einschränkungen. Man denke beispielsweise daran, dass die meisten (oder sogar alle) Testautomatisierungstools dafür genutzt werden, Tests für Entwickler durchzuführen und Ergebnisse zu liefern. Weil die meisten Tools nicht wissen, welche Tests durchzuführen sind, starten einfach alle oder eine vorgegebene Auswahl. Wie wäre es also, wenn ein KI-fähiger Bot den aktuellen Teststatus, neuere Codeänderungen, die Codeüberdeckung und andere Metriken auswerten könnte, um auf dieser Basis zu entscheiden, welche Tests noch notwendig sind, und diese Tests anschließend auszuführen? Das Einbringen einer Entscheidungsfindung auf der Basis veränderlicher Daten ist ein Beispiel für den Einsatz von KI. Damit ist die Software effektiv in der Lage, den Entwickler bzw. Tester in diesem Entscheidungsprozess zu ersetzen.

KI kann in der Softwareentwicklung immer dann glänzen, wenn es darum geht, diese Einschränkungen zu beseitigen, oder wenn die Tools zum Automatisieren von Softwaretests den Entwicklern und Testern einen noch größeren Nutzen bieten sollen. Der Nutzen der KI resultiert daraus, dass die direkte Einbeziehung der Entwickler oder Tester in die besonders profanen Aufgaben reduziert wird. (Überaus gefragt ist die menschliche Intelligenz dagegen nach wie vor u. a. bei der Anwendung von Geschäftslogik.) [EMBD2020]

Literatur

- [SQMA2018] : SQ Magazin, ASQF, Ausgabe 48/2018, Intelligence of Things, Markus Höber „Kognitives Testen“
- [EMBD2020] : <https://www.embedded-software-engineering.de/was-bedeutet-kuenstliche-intelligenz-beim-software-testen-a-817057/>, zugegriffen am 09.05.2020

Nachwort

Eine Vielzahl von Herausforderungen und Problemen, die auf den Testmanager im Projekt hinzukommen werden, lassen sich in der Phase der Konzepterstellung bereits einschätzen.

Daher ist es wichtig, frühzeitig, mehrere Wochen bevor der erste Tester in das Projekt eingebunden wird, mit der Erstellung eines Testkonzeptes zu beginnen. Viele Erfahrungen und Risiken sind bereits bekannt, bevor das Projekt überhaupt startet.

Man kann sich natürlich fragen, warum nicht von Anfang an mehr Ressourcen, mehr Zeit, weniger Funktionen für ein Projekt geplant werden. Wenn man ausgewertet hat, dass grundlegende Voraussetzungen schwierig sind – sei es die Verfügbarkeit der Testumgebung, die Qualität des Anforderungsmanagements oder die Struktur der Projekte – könnte man diese Dinge proaktiv und gezielt optimieren. Von dieser Optimierung würde jedes weitere Projekt profitieren.

Nach der erfolgreichen Einführung eines Projektes ist es oftmals schwierig, die Defizite, die während der Projektphase aufgekommen sind, direkt zusammenzufassen und sich zu motivieren, diese zu analysieren. Nachdem durch die nachträgliche Korrekturlieferung die größten Probleme behoben wurden und die Einführung bzw. der Rollout geschafft sind, tritt meist eine gewisse Projektermüdung bei den Beteiligten auf. Nun wollen die Erfolge gefeiert werden.

Man müsste eigentlich eine längere Phase zur Verfügung haben, in dem gar kein Projekt durchgeführt wird und man den betrieblichen Ablauf und die Testprozesse zielgerichtet optimiert, bevor man in das nächste Projekt startet. Nur Projekte bringen aber das notwendige Geld, um die Existenz der IT-Abteilung überhaupt zu rechtfertigen. Für externe Unterstützung, mit deren Hilfe Prozesse angepasst und erneuert werden könnten, gilt das im Besonderen. Also bleiben längere Phasen, in denen „erst einmal Ruhe im Unternehmen einkehrt“, in denen die notwendigen Anpassungen durchgeführt werden, ein frommer Wunsch. Das sieht man sehr deutlich immer wieder in den Wirtschaftskrisen – wie um die Jahrtausendwende, in der Finanzkrise um 2010 und in der Corona-Krise 2020. Eigentlich wäre jetzt ausreichend Zeit vorhanden, endlich einmal die Prozesse zu verbessern, die Dokumentation nachzuziehen und aufzuräumen.

Wenn aber aufgrund einer nachlassenden Auftragslage oder schwierigeren Konjunktur mehr Zeit dazu vorhanden wäre, elementare Probleme anzugehen, kommt es schnell zu finanziellen Problemen, Personalabbau, Sparen mit dem Rotstift und noch stärkere Beschränkung auf das Nötigste. Die Optimierung der Prozesse ist dann eher nachgelagert, wenn man gar nicht weiß, ob das Unternehmen in dieser Form weiter bestehen kann oder eine Insolvenz droht.

Die entscheidenden Fehler machen Unternehmen meistens dann, wenn es ihnen (zu) gut geht: Man hat ohnehin genügend Umsatz und erwartet ansehnliche Rendite, da kann man sich auch weniger produktive Abläufe leisten. Ob man dabei alle Fehler entdeckt oder nicht, ist auch nicht so entscheidend, zur Not wird eben eine verbesserte Softwareversion ausgerollt, es stehen ja ohnehin genügend finanzielle Ressourcen bereit. Der Engpass besteht dann eher im verfügbaren Personal, das aufgrund von Arbeitsüberlastung nicht termingerecht fertig wird. Sicherlich arbeitet man vielleicht nicht effizient, aber das wird in diesem Fall nicht als so wichtig betrachtet. Effizienz ist erfahrungsgemäß sowieso ein nachgelagertes Kriterium, wenn es wirklich darum ginge, würden Prozessverbesserungen wesentlich offensiver angegangen. Wenn im Unternehmen jedoch gerade der Umsatz zurückgeht und der Gewinn einbricht, denkt man stärker darüber nach, wie man Optimierungspotenziale heben kann und ist dadurch eher bestrebt, keine Fehler zu machen und nicht zusätzlichen Aufwand zu generieren.

Man spricht vom Gleichgewicht zwischen „brain“ und „capital“: In Krisenzeiten, wo weniger Kapital zur Verfügung steht, kommt es ganz besonders darauf an, durch Gedanken, Wissen und Innovation Verbesserungen zu erzielen und sich auf seine Stärken zurückzubedenken, anstatt in den Zeiten in denen Defizite eher durch Geld ausgeglichen werden können. Das kann man bei Projekten beobachten, aber auch im täglichen Leben. Ein Land, das über sehr viele Rohstoffe verfügt, muss sich über Innovation weniger Gedanken machen als ein rohstoffarmes Land. Wer reich geerbt hat, muss sich weniger anstrengen, Geld zu vermehren, sondern nur aufpassen, dass er es nicht durch unbedachte Funktionen verliert. Wenn man rohstoffreiche Länder wie Russland oder Saudi-Arabien mit einem rohstoffarmen Land wie Deutschland vergleicht, wird das besonders deutlich.

Da man an den Rahmenbedingungen wenig ändern kann, ist es umso wichtiger, darauf hinzuweisen und sich auch selbst nicht einem zu hohen Erfolgsdruck auszusetzen. Ich habe schon erlebt, dass Projekte, die mit großem Elan begonnen wurden, gestoppt oder gestreckt wurden. Der Sinn erschließt sich dem Betrachter dann nicht, und rein wirtschaftlich ist es zumindest mittelfristig auch eine Fehlentscheidung. Da aber ein Quartalsdenken und Shareholder Value in den Unternehmen Einzug gehalten hat, kommt es meist vor, dass manche sinnvollen Verbesserungen, die erst über einen längeren Zeitraum Wirkung entfalten können, gar nicht vorgenommen werden, sondern hauptsächlich „low hanging fruits“ geerntet werden. Daher ist es auch nicht außergewöhnlich, wenn man die Risiken aus Projekt A im Jahr 1 fast unverändert für Projekt B in Jahr 2 übernehmen kann. Der Vorteil bei Projekt B ist lediglich, dass man aus der Vergangenheit schon weiß, wie man damit umgehen muss und im besten Fall die Kalkulation für Projekt B genau diese Beschränkungen bereits beachtet. In vielen Fällen aber erbt man als Testmanager ein Projekt und war

an den Aufwandsermittlungen gar nicht beteiligt. Man bewegt sich oft in einem engen vorgegebenen Rahmen.

Wenn man Testaktivitäten generisch betrachtet und als Stabsabteilung oder Qualitätsabteilung neue Prozesse einführen kann, ist man eher in der Lage, diesbezüglich neue Akzente zu setzen. Dafür ist aber eine schonungslose Analyse der betrieblichen Wirklichkeit eine unbedingte Voraussetzung. Dazu sollten auch die einzelnen Testabteilungen stark eingebunden werden. Oft erlebt man die Qualitätsabteilung als eine theoretisch und abstrakt agierende Gruppe. Normen und Standards müssen aber für den betrieblichen Alltag übersetzt werden. Die Qualitätssicherung wird nur dann Akzeptanz finden, wenn sie konkret anhand von Beispielen und realen Funktionen zeigt, wie innerbetriebliche Abläufe optimiert werden können, ohne für die beteiligten Abteilungen (Entwicklung, Test, Systemdesign ...) nur Zusatzaufwände für die Erstellung neuer Dokumente zu erzeugen.

Vom Konzept kommt man zur Umsetzung, indem man die einzelnen Aktionen, die während des Tests zu erledigen sind, in einem Plan erfasst und die Erledigung trackt. Oft weiß man allerdings gar nicht, wer von wem abhängig ist, welche Aktionen zusammenhängen oder aufeinander aufbauen. Die Dokumentation gibt das meist nicht her, und Workflow-Dokumente sind selten. Deswegen muss man sich hier häufig durchfragen. Vor allem in Konzernen und großen Projekten mit zahlreichen internen und externen Beteiligten kann diese Übersicht sehr komplex werden. Es ist aber unbedingt zu empfehlen, sich als Testmanager die Abhängigkeiten bewusst zu machen und zu einem gewissen Grad die fachlichen Anforderungen nachzuvollziehen, da man sonst nicht von den Testern ausreichend akzeptiert werden wird.

Es wird immer wichtiger, in Prozessen zu denken. Prozessstruktur versus Funktionsstruktur – dieser Gegensatz hemmt die Entwicklung von **Process Excellence** in Unternehmen. Erfolgversprechende Projekte des Prozessmanagements können häufig ihre Wirkung durch die bestehende Funktionsstruktur nicht voll entfalten, weil eine zusammenhängende Prozessverantwortung nicht existiert. Das langfristige Strukturelement in Unternehmen sind wenige aber wesentliche Unternehmensprozesse, die in jedem Unternehmen für ein erfolgreiches Geschäftsmodell erforderlich sind. Mit einer gelebten Prozessstruktur wird ein Unternehmen unabhängiger von Personen und Entscheidungsträger, die häufig ihre Arbeitsplätze und Unternehmen wechseln. Nachhaltige erfolgreiche Prozess Exzellenz in Unternehmen erfordert eine Neuorientierung der Führungs- und Unternehmenskultur, ein integriertes Denken und Handeln in Prozessen. Prozessverantwortung muss als alleiniges Führungsprinzip etabliert werden [ZANG2009].

Mithilfe einer gelebten Prozessstruktur hat man es leichter, eine geeignete Teststrategie zu entwickeln. Das hilft auch während des Projektes bei den Testaktivitäten von Erstellung der Spezifikation über Testdurchführung und Testautomatisierung bis hin zum Reporting der Ergebnisse.

Jedes Projekt, jede Branche, jede Anforderung ist letztlich individuell. Immer wieder tauchen in einem neuen Projekt völlig neuartige Herausforderungen auf. Es gibt auch immer wieder Dinge, die in der betrachteten Organisationseinheit derart speziell ablaufen, dass man standardmäßige Vorschläge darauf gar nicht mehr anwenden kann, und mit dem

Defizit leben muss. Wichtig ist immer, dies vorab bereits als Risiko zu benennen und an geeigneter Stelle zu adressieren.

Von der Konzeptphase geht man fließend in die Umsetzungsphase über. Aber ein Testkonzept muss leben, es darf nicht in Stein gemeißelt sein, sondern muss an die betrieblichen Belange angepasst werden. Je mehr man immer wieder auf das Testkonzept verweist, desto eher wird es auch zu einem zentralen Dokument im Entwicklungs- und Testprozess und nicht nur als Alibidokument verwaltet.

Das Testmanagement entwickelt sich weiter. Die agile Transformation hat viele Unternehmen erreicht. Oft ist aber die Umsetzung der Vorgaben nur ansatzweise durchgeführt. Durch Änderungen im Umfeld agiler Prozesse wird das Testmanagement vor neue Herausforderungen gestellt, die mit einem höheren Testautomatisierungsgrad und neuen Strukturen einhergehen und deshalb auch eine konzeptionelle Neuausrichtung erfordern. Neue technologische Trends, von künstlicher Intelligenz über Blockchains bis hin zu Quantencomputern, werden weiterhin zu einer hohen Dynamik in der Softwareentwicklung führen und durchdachtes Testmanagement in Zukunft noch wichtiger werden lassen. Das größte Wachstum haben Unternehmen, bei denen Software kein Tool ist, das Workflows, Produktion oder Kommunikation unterstützt und dass man später als Add-On dazu kauft, sondern immer mehr zu einem Mindset wird, das zum Kernprodukt gehört und dadurch einen der gewinnbringendsten Vorteile darstellt. Mit Einführung einer reinen IT-Implementierung sowie der Einführung von ein paar agilen Methoden und Tools im Arbeitsalltag ist es nicht mehr getan, Software muss ein zentrales Selbstverständnis in allen Unternehmensbereichen sein. Dadurch wird der technologische Veränderungsdruck weiter ansteigen. Die Softwareentwicklung und dadurch auch das Testmanagement werden dadurch in ein Stadium einer höheren Produktreife kommen, in denen sich klassische Industrien wie Maschinenbau oder Fahrzeugbau bereits seit langem befinden. Die bereits seit mehreren Jahren beobachteten Prozesse zur Industrialisierung und Standardisierung von Testprozessen werden dadurch zunehmen. Indem sich der Softwaretest wandelt und in der Bedeutung weiter steigen wird, wird die Notwendigkeit zur strategischen Planung und Organisation von Testprojekten mithilfe klar strukturierter und vollständiger Testkonzepte steigen.

In meinem Buch habe ich an mehreren Stellen darauf hingewiesen, wie wichtig die Kommunikation in Projekten ist und wie häufig Projekte vor allem an mangelhafter Kommunikation und nicht an technischen oder mathematischen Herausforderungen scheitern. Dazu gehört auch, sich über die verwendeten Begriffe in der Organisationseinheit einig zu sein. Unter dem Namen einer Testumgebung, einer Software, eines Dokuments oder einer Abkürzung müssen alle dasselbe verstehen, sonst ist eine babylonische Sprachverwirrung vorprogrammiert. Ich habe die Begriffe im Text erklärt und am Ende des Buches ein Stichwortverzeichnis ergänzt. Dieses Vorgehen sollte auch in Unternehmen selbstverständlich sein. Ein eigenes Glossar oder ein Abkürzungsverzeichnis und eindeutige Erklärungen der verwendeten Verfahren sind wichtig, um nicht aneinander vorbeizureden. Aufgrund der unterschiedlichen organisatorischen Verortung, unterschiedlicher Ausbildung und beruflicher Erfahrung, unterschiedlicher Kulturen aber auch unterschiedlicher Charaktere sind

die Herausforderungen der Kommunikation ohnehin schon kompliziert genug. Das alles sind Probleme, die eine Testorganisation oder auch ein noch so gutes Testkonzept nur in einem engen Rahmen überhaupt beeinflussen kann. In den meisten Fällen muss man mit den vorhandenen Rahmenbedingungen umgehen und trotzdem versuchen, das Projekt noch zu retten.

Literatur

[ZANG2009] : <http://dodo.fb06.fh-muenchen.de/zangl/downloads/Prozessphilosophie.pdf> Denken und Handeln in Prozessen. Prof. Dr. Hans Zangl, Hochschule München, 2009, zugegriffen am 09.05.2020

Stichwortverzeichnis

A

Ablaufstruktur 152
Abnahmekriterien 107
 -kriterienkatalog 108
 -prozess 110
 -test 64, 75
 -testumgebung 110
A/B-Test 86
Adaptive Methode 223
Agiles
 Projekt 33, 235
 Testen 240
 Vorgehensmodell 43
Akzeptanz
 -kriterien 111
 -test 113
Amazon Web Service (AWS) 161
Amendment I 150
Anonymisierung 130
Application Programming Interface 133
Arbeitsplan 183
Architekturskizze 66
Argumentenbilanz 195
Aufbaustruktur 152
Auftraggeber 55
Ausbalancierte Matrixorganisation 213
Auswahl des Informationsempfängers 170
Automatisierte Datenbereitstellung 131
Autonome Projektorganisation 211
Autonomiegrad 244

B

Balance Scorecard 16
Balkenplan 185
BDTM (Business Driven Test Management Approach) 221
Berichtsplan 31
Bewertung des Projektfortschritts 214
Beziehungs
 -management 179
 -problem 179
Black-Box
 -Test 74
 -Verfahren 72
Blinder Ansatz 132
Business Driven Test Management Approach (BDTM) 221

C

Change 60
 -management 159
Chat 51
CMMI 12
COCOMO
 -II-Methode 190
 -Modell 181
Code-Inspektion 90
Continuous Integration 236
Cost
 -Performance-Index (CPI) 215

Variance (CV) 215
 CPM-Plan 184
 Critique the product 236
 Crowd
 -sourced Testing 238
 -testing 238

D

Daten
 -bereitstellung 54
 -maskierung 130
 -virtualisierung 130
 Dead-Lock-Situation 85
 Defect Burndown Rate 141
 Defectmanager 149
 Definition of Done 241
 Delegation 169
 Dokument 42

E

Effizient 228
 Einflussfaktoren Testaufwand 70
 EN 5012812
 Enabler 227
 Entscheidungs
 -baum 198
 -problem 178, 202
 -technik 194
 Error Guessing 76
 Erwartete Restkosten 215
 Estimate to Complete (ETC) 215
 Exploratives Testen 76
 Externe Testumgebung 160
 Extreme Programming 80

F

FCC-CUTS-VIDS-Touring-Heuristik 76
 Featureliste 65
 Fehler
 -findungskurve 40
 -klasse 46
 -quelle 176
 Fehlerklasse 46
 Finanzperspektive 17
 Freigabe 205
 -empfehlung 205

-prozess 206
 Function Point 33
 Funktionaler Test 83

G

Gezielter Ansatz 132

H

Handlungsalternative 194
 Hardware 54
 HICCUPPS 76

I

IEC 6150812
 IEEE 829 Standard for Software Test
 Documentation 1
 Impact Domain 67
 Impaktanalyse 153
 Informations
 -austausch 179
 -inhalt 172
 -problem 178
 Initial 228
 Inkrementelle Integration 74
 Integrations
 -strategie 13
 -test 22, 74
 Integritätslevel 7, 136
 ISO 9126 7
 ISO 12207 6
 ISO 15288 6
 ISO 25010 237
 ISO 26262 12
 ISO 91266 136
 ISO/IEC/IEEE 29119 2
 IT-Governance 222

K

Kanban 236
 Kick-Off-Meeting 61
 Kombiniertes Ansatz 132
 Kommunikations
 -matrix 168
 -planung 170
 -weg 169

- ziel 172
- Kompetenzproblem 178
- Komplexität
 - der Kommunikation 166
 - smaß nach Rechenberg 81
- Komponententest 22, 73
- Konfigurationsmanagement 159
- Konfliktmanagement 178
- Konsequenztabelle 197
- Kontrolliert 228
- Kontrollpunkt 227
- Kosten
 - abweichung 215
 - entwicklungsindex 215
 - schätzung 184
- Kosten- und Finanzplan 30
- Kritischer Pfad 52, 184
- Künstliche Intelligenz 243
- Kundenperspektive 17

L

- Lasttest 85
- Leistungsmerkmale 83
- Lenkungsausschuss 55

M

- Managed-Service-Modell 49
- Mastertestplan 222
- Master Test Plan Metrics Report 7
- Matrixorganisation 212
- Meeting 60
- Methode der Investitionsrechnung 195
- Metrik 141
- Mock-Objekte 96
- Modellbasiertes Testen 133
- Multivariater Test 87
- Mutationsansatz 132

N

- Netzplan 184
- Nichtinkrementelle Integration 74
- Norm DIN 69901 29
- Normen 1
- Norm IEC 62034 150

- Norm IEC 62304 151
- Nutzwertanalyse 199

O

- Object Point 33
- Objektmodell 67
- Office Environment 233
- On Premise 162
- Optimierend 228
- Orakel 76
- Organisations
 - plan 31
 - problem 178
 - struktur 55

P

- Peer Review 220
- Permanente Testorganisation 101
- Persönlichkeitsmerkmal 178
- Personalauslastungsplan 187
- Planung und Steuerung 60
- Potenzialperspektive 17
- PRINCE 2 56
- Priorisierung 94
- Private Cloud 162
- Problemlösungsprozess 151, 152
- Process Excellence 249
- Product Backlog Item 241
 - Owner 10, 79, 238
- Produktrisiken 127
- Projekt
 - ablaufplan 30
 - ampel 24
 - bedingung 191
 - controlling 142
 - dokumentation 5
 - kommunikation 173
 - leiter 55
 - lenkungsausschuss 55
 - managementkultur 56
 - mitarbeiter 55
 - organisation 54, 209
 - planung 31
 - produktokumentation 5

- risiken 127
- situation 17
- Projektstrukturplan (PSP) 30
- Projekt- und Sprint-Teststrategie 78
- Prozessperspektive 17

Q

Qualitäts

- sicherungsmaßnahmen 68
- sicherungsplan 30
- ziel 26

R

RACI-Methode 174

RACIO 175

RASCI 175

Rechenzeit 53

Refactoring 79

Reifegrad 226, 228

- prozess für die Testorganisation 228

Reine Projektorganisation 211

Release

- management 159
- Notes 6, 65
- Planning 43

Reporting 233

Requirement 42, 65

Requirementsmanagement 84

Ressourcenplan 30

Review 39, 90

Risiko

- analyse 201
- basiertes Testen 98
- orientiertes Testen 127
- plan 30

Rolle 55

Rollen

- beschreibung 148
- konflikt 179

S

Schedule

- Compression-Faktor 182
- Performance Index (SPI) 215

Variance (SV) 215

Scrum 45, 236

- Prozess 79

Softwarelebenszyklus 136

SPICE 12

Sprint 236

Stablinien-Projektorganisation 209

Stakeholder 55

- analyse 170

Start-up-Workshop 61

Startveranstaltung 61

Statusbericht 5

Strategieprozess 43

Stresstest 85

Supporting the team 236

SWOT-Analyse 196

System

- lebenszyklus 136
- test 22, 75

T

Termin

- entwicklungsindex 215
- plan 30

Test

- abbruch 115
- abdeckung 90
- abdeckungskenngroße 23
- ablaufspezifikation 4
- abschlussbericht 4, 141, 205
- abweichungsbericht 4
- analyst 149
- architektur 96
- aufgabe 147
- aufgaben im Regressionstest 152
- aufwand 13
- aufwandsplanung 188
- aufwandsschätzung 34
- automatisierung 34
- bericht 4
- bewertungsbericht 5
- charta 241
- Cluster 83
- datenbereitstellung 131
- datengeneratoren 133
- dauer 181

- dienstleistung 101
- dokumentation 135
- Driven Development 79, 236
- entwurfsspezifikation 3
- ereinsatzplanung 52
- ergebniskenngröße 23
- Factory-Modelle 49
- Factory-Services 49
- fall 34, 138
- fallnachzustand 68
- fallspezifikation 3
- fallvorzustand 68
- fortschritt 25
- fortschrittsbericht 6, 50
- kenngröße 23
- konzept 2
- manager 148
- methode 33, 72, 217
- objekt 40, 67
- objektübergabebericht 4
- organisation 49
- personal 52, 177
- phase 46, 68, 184
- plan 81
- planung 29
- point 190
- Process Improvement (TPI) 221
- produktivität 190, 191
- protokoll 4, 138
- prozess 22, 59, 227
- punkt 40
- reifematrix 226
- Reporting 24
- ressource 52
- richtlinie 9, 104
- risiken 120
- skript 137
- spezifikation 3, 13
- strategie 9, 218
- stufe 73
- stufensteckbrief 6
- szenario 74
- tools 153
- umfang 93
- umgebung 157
- umgebung in der Cloud 161
- umgebungsmanagement 158

- umgebungsmanager 149
- unterbrechung 41
- voraussetzung 32, 46, 65
- werkzeug 34
- wiederanlauf 41
- ziel 21, 74
- zielorientierte Strategie 74
- Testaufwand 13
- Testing Maturity Model (TMMI) 219
- Test Point 33
- Testware Management 233
- TMap 221
- TPI-Modell 225
- TPI Next 225
- TPI (Test Process Improvement) 221
- Traceability Matrix 65

U

- User Story 112

V

- VARISC 175
- Verantwortlichkeit 165
- Verfügbarkeitsmanagement 159
- Versionierung 39
- Vertragsstrafe 46
- Vorgehensorientierte Strategie 74

W

- White-Box
 - Test 74
 - Verfahren 72
- Wirkungsbereich 67
 - faktor 202

Z

- Zeitabweichung 215
- Zieleverantwortung 213
- Zugriffs
 - management 159
 - rechte 160
- Zuständigkeit 165
- Zyklomatische Komplexität 81