HANSER

Der Systemtest

Harry M. Sneed, Manfred Baumgartner,
Richard Seidl

Anforderungsbasiertes Testen von Software-Systemen

ISBN 3-446-40793-6

Leseprobe

3

3 Systemtestplanung

3.1 Zweck der Testplanung

Der System- bzw. Abnahmetest ist ein Projekt für sich, ein Projekt, das neben dem eigentlichen Entwicklungs- bzw. Integrations-, Migrations- oder Installationsprojekt läuft. Als solches muss es wie andere Projekte definiert, kalkuliert, organisiert und geplant werden. Das Ziel des Testprojekts ist es, die Produktionsreife der von dem Hauptprojekt gelieferten Software festzustellen. In der Praxis hat sich ein anderes Ziel manifestiert, nämlich die Fehler für die Entwickler zu finden. Angesichts des Zeitdrucks unter denen Projekte abgewickelt werden, hat sich herausgestellt, dass es schneller geht, wenn die Entwickler von der Last des Testens befreit sind. Außerdem sind viele Entwickler durch die steigende Komplexität moderner IT-Systeme schlicht überfordert. Trotz aller Mahnungen, die eigene Qualität zu sichern, trotz aller neuen Entwicklungsmethoden und ausgeklügelten Modultestwerkzeuge bleibt die Zahl der Fehler erschreckend hoch. Die Erfahrung zeigt, dass Software, die vom Entwickler kommt, 3 bis 18 Fehler pro 1.000 Anweisungen aufweist [GRAV00]. Mit immer neuen Programmiertechnologien steigt die Fehlerrate. Demnach haben alle Bemühungen, Qualität in die Entwicklung zu bringen, nicht gefruchtet. Ob prozedural, objektorientiert oder agentenorientiert, die Entwicklung von Software bleibt ein fehlerhaftes Unterfangen. Und trotz aller Versuche, Entwickler dazu zu bringen, ihre eigene Komponente systematisch zu testen, testen Entwickler nach wie vor sehr ungern. Oft fehlt ihnen auch noch die Zeit dazu. Es existieren psychologische Barrieren, die verhindern, dass Entwickler ihre eigenen Fehler überhaupt finden wollen [WEIN72]. Ergo bleibt nichts anderes übrig, als andere Menschen einzusetzen, die stellvertretend für die Entwickler testen. Diese anderen Menschen sind die Tester, und ihr Projekt ist der System- bzw. der Abnahmetest.

Dieses Projekt, der unabhängige Test, benötigt ein eigenes Budget und einen eigenen Zeitplan. Um beides zu beantragen – das Geld wie auch die Zeit – muss die zu leistende Testarbeit kalkuliert werden. Aus der Testkalkulation geht hervor, wie viele Tester wie lange testen müssen, um alle erforderlichen Testfälle durchführen zu können und notfalls zu wiederholen. Oder, umgekehrt, kommt dabei heraus, wie viele Testfälle die Tester in der

vorgegebenen Zeit durchführen können. Ist einmal erkannt, wie viele Tester für wie lange benötigt werden, kann das Testmanagement dazu übergehen, die Testarbeit zu verteilen. Dies ist eine Frage der Testorganisation. Schließlich muss das Testmanagement gewisse Kontrollmechanismen installieren, um den Testfortschritt zu verfolgen, die Testkosten zu überwachen und die Qualität der Testarbeit zu sichern. Darin unterscheidet sich ein Testprojekt nicht von anderen Projekten. Die Voraussetzung dafür ist ein Testplan, in dem die Ziele abgesteckt sind.

Der Testplan regelt, was, warum, wann, wo, wie, womit und von wem getestet wird (siehe Abbildung 3.1). Für die Planung eines strukturierten Systemtests müssen diese Fragen beantwortet werden.

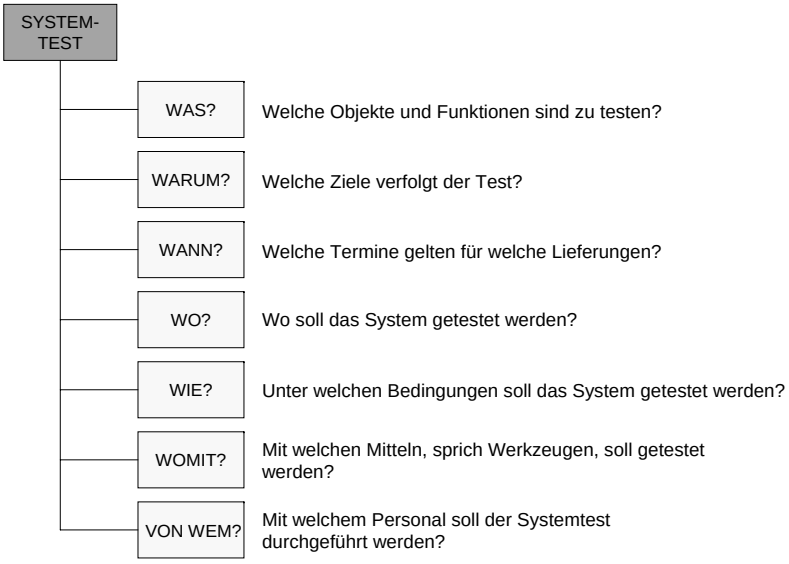

Abbildung 3.1 Die 7 Ws der Testplanung

Das *Was* bezieht sich auf die zu testenden Objekte und Funktionen des Systems. Ein System umfasst verschiedene Objekttypen die von außen erkennbar sind, darunter die Benutzeroberflächen, die Datenbanken, die Berichte und die Systemschnittstellen. Sie werden aus den Anforderungen hervorgehen, auch dann, wenn sie nicht explizit beschrieben sind. Sie sind die Gegenstände eines Black-Box-Testes. Die Funktionen sind die Anwendungsfälle, die explizit oder implizit in der Anforderungsspezifikation beschrieben sind. Aus der Anforderungsanalyse geht hervor, welche Objekte, Funktionen und Qualitätseigenschaften zu testen sind, also die Quantität und Qualität des Produktes. Die Quantität drückt sich in der Zahl der erforderlichen Testfälle aus. Die Qualität drückt sich in den Maßstäben der einzelnen Qualitätsziele aus. Der Testplan hält den Leistungsumfang des Testprojekts fest. Er bestimmt, wie viele Fälle zu testen und welche Qualitätseigenschaften zu messen sind. Mit dem Testplan verpflichtet sich das Testmanagement, den definierten und vereinbarten Umfang abzudecken (siehe Abbildung 3.2).

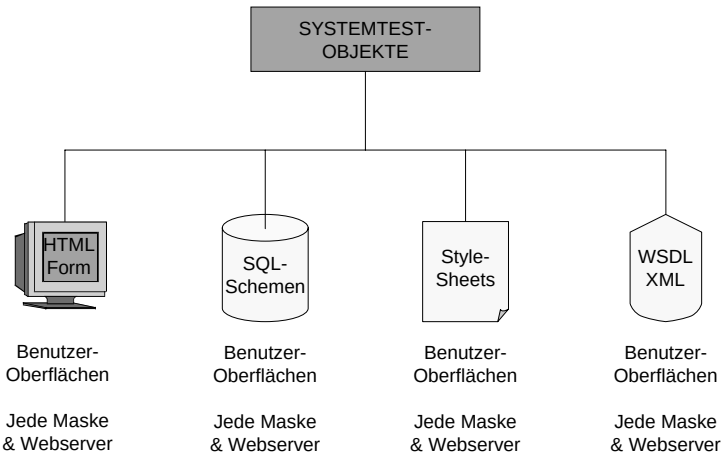

Abbildung 3.2 Systemtestobjekte

Das *Warum* ist eine Frage der Zielsetzung. Was sollte der Test erreichen? Welche Ziele werden angestrebt? Gerade beim Test, einer Arbeit, die sehr undurchsichtig ist, kommt es darauf an, die Leistungen präzise zu definieren. Nach der Grundlektüre „Basiswissen Softwaretest“ hat der Test zwei Hauptziele:

- Fehler zu finden
- Vertrauen in das System zu gewinnen [SPILL02]

Diese Hauptziele können jedoch nur über Nebenziele erreicht werden. Da man nie genau wissen kann, wie viele Fehler eine Software hat, kann man sie nur aufgrund der Erfahrung der Vergangenheit projektieren. Sie werden anhand der Fehlerrate der Vergangenheit hochgerechnet. Umso wichtiger ist es, die Fehlerrate aller vergangenen Projekte festzuhalten. Die Fehlerrate ist das Verhältnis der gefundenen Fehler zur Größe des Projekts, sei es in Anweisungen, Function-Points, Object-Points oder in beliebigen anderen Kategorien. Diese bisherige Fehlerrate wird herangezogen, um die Fehlerrate des geplanten Projekts zu schätzen. Das Vertrauen in der Software wird indirekt über diverse Überdeckungsmaße aufgebaut. Bei der Testplanung geht es darum, diese Überdeckungsgrade mit der Projektleitung abzustimmen und festzuschreiben. Typische Überdeckungsgrade beim Systemtest sind die Überdeckung aller Anforderungen aus dem Fachkonzept, die Überdeckung aller Qualitätsziele und die Überdeckung aller Schnittstellen. Es ist auch möglich, den Softwarecode zu instrumentieren und die Überdeckung aller Source-Bausteine, wie Methoden und Prozeduren zu messen. Jedenfalls ist es unerlässlich, eindeutig messbare Ziele – so genannte Testendekriterien – für den Systemtest zu setzen.

Das *Wann* ist eine Frage der Termine. Bis zu welchem Termin wird welcher Test abgeschlossen. Jede Testaktivität soll einen Starttermin und einen Endtermin haben. Welche Zeitpunkte das sind, hängt vom Zeitpunkt der Softwarelieferung und dem Status der anderen Projekt- und Testaktivitäten ab. Es gilt außerdem, gewisse Abhängigkeiten zwischen den Testaktivitäten zu betrachten. Es ist zum Beispiel nicht sinnvoll, mit dem Lasttest zu

beginnen, bevor der Funktionstest nicht in einer fortgeschrittenen Phase ist. Die Software wird meistens in Paketen geliefert. Der Test von einem Paket kann erst beginnen, wenn das Paket installiert ist. Oft lässt sich ein Paket nicht testen, bevor ein anderes nicht getestet wurde. In dem Testplan werden diese zeitlichen Abhängigkeiten geregelt und die Termine bestimmt, zum Beispiel in Form von Netzplänen oder in Form von Gantt-Diagrammen (siehe Abbildung 3.3).

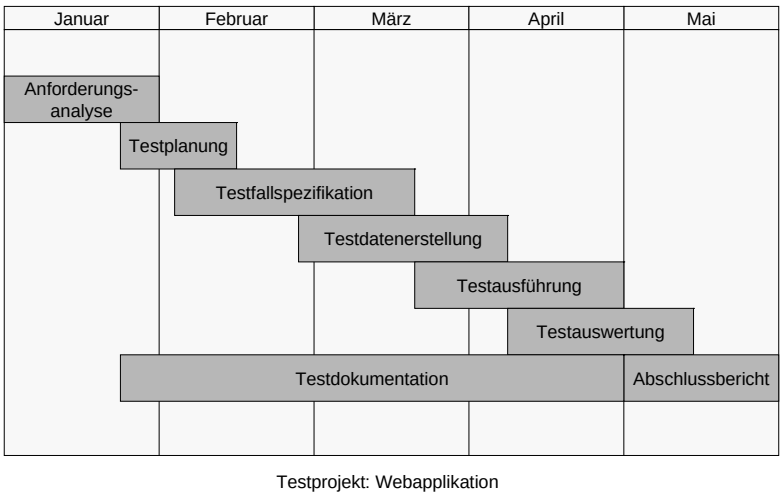

Abbildung 3.3 Testprojektplan

Das *Wo* ist eine Frage des Orts, an dem der Test ausgeführt wird. Dies muss nicht unbedingt in einem extra dafür bereitgestellten Testraum sein. Die Tester können bei den Entwicklern sitzen oder bei den Endanwendern. Sie können auch zu Hause sitzen oder gar irgendwo in einem fremden Land. Moderne Kommunikationsmittel haben es möglich gemacht, die Arbeit vom Ort zu trennen. Dies setzt jedoch voraus, dass die Arbeit genau spezifiziert ist und alle offenen Fragen per E-Mail geklärt werden. Ist dies nicht der Fall, sollten die Tester zumindest in der Nähe der Entwickler sitzen, damit sie jederzeit persönlich nachfragen können. Die Tester müssen auch nicht alle zusammen in einem Raum oder Gebäude sitzen. Im Prinzip können sie über das Weltall verteilt werden, wichtig ist nur, dass sie – wo immer sie auch sitzen – Zugriff auf das Testobjekt, die Testwerkzeuge und die Verfasser der Anforderungen haben. Die Frage der Örtlichkeit des Tests soll in dem Testplan geregelt werden.

Das *Wie* ist eine Frage der Testmethode. Wie werden die einzelnen Tests durchgeführt? Ein Funktionstest kann von den Testern vom Bildschirmarbeitsplatz aus durchgeführt werden, indem sie einfach das eingeben, was ihnen gerade einfällt. Das nennt man kreatives Testen. Deshalb wurde Testen ursprünglich als Kunst angesehen [MYER79]. Viele Entwicklungsbetriebe haben versucht, billiges, unqualifiziertes Personal als Tester einzusetzen. Sie hätten sich genauso gut Affen aus dem lokalen Tiergarten ausleihen und sie auf die Bildschirmarbeitsplätze los lassen können. Deshalb wird dies als Affentest bezeichnet

[BEIZ95]. Die Ergebnisse blieben unter dem Strich negativ – wenig Geld für wenig Fehler. Die Fehler, die auf diese Art aufgedeckt werden, sind auch eher trivial. Andererseits kann der Betrieb qualifizierte Tester anweisen, sich strikt an die spezifizierten Testfälle zu halten. Sie sollten die spezifizierten Testfälle vor sich haben und einen nach dem anderen bearbeiten. Dies ist der systematische Test. Er deckt mehr und anspruchsvollere Fehler auf, kostet aber entsprechend mehr. Schließlich könnten die spezifizierten Testfälle dazu benutzt werden, einen Automaten zu bedienen, der sie der Reihe nach durchführt. Das ist der automatisierte Test. Der automatisierte Test bringt die gleichen Vorteile wie der systematische Test, kostet aber nur einen Bruchteil davon. Der Last- und Performanztest wird ohnehin mit Automaten durchgeführt, weil es gar nicht anders geht. Kurzum wird hier entschieden und beschrieben, wie die einzelnen Tests durchzuführen sind. Am Ende muss ein Testprozess festgelegt werden, in dem die einzelnen Testaktivitäten auf die Reihe gebracht werden (siehe Abbildung 3.4).

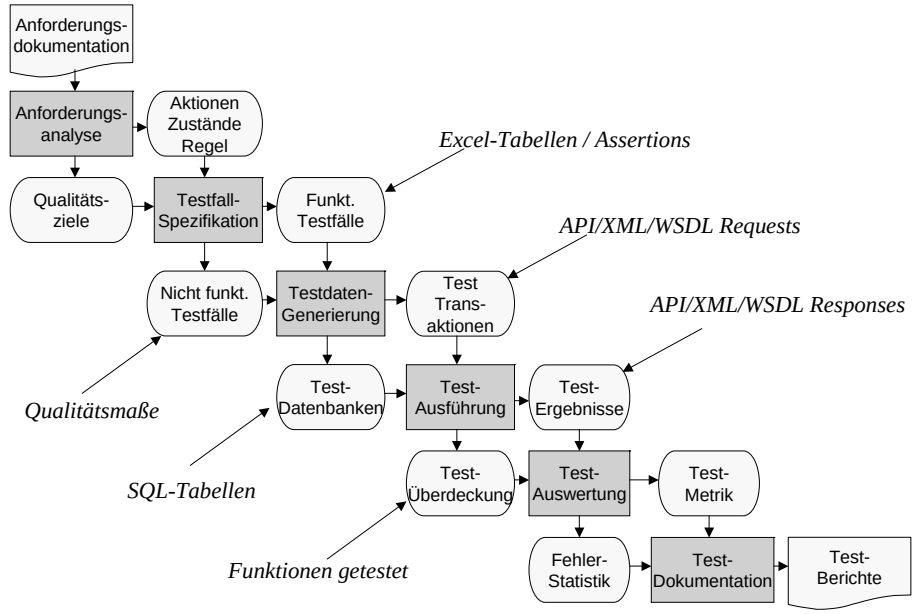

Abbildung 3.4 Festlegung des Testprozesses

Das *Womit* ist eine Frage der Testwerkzeuge. Um den Test zu beschleunigen und gleichzeitig die Qualität und Effektivität des Tests zu steigern, werden Werkzeuge eingesetzt. Es beginnt schon mit der Analyse der Anforderungen und der Ermittlung der Testfälle. Auch dafür können Werkzeuge eingesetzt werden. Für die Validierung der Ergebnisse bzw. den Abgleich der Soll- und Ist-Ergebnisse, bieten sich andere Werkzeuge an. Werkzeuge werden ebenfalls für das Aufzeichnen und Zurückspielen der Testfälle eingesetzt. Statt mit Affen oder Menschen zu testen, kann man mit Automaten bzw. Testrobotern testen. Mit Werkzeugen werden Testdaten generiert, Testergebnisse kontrolliert, Testabläufe verfolgt und Testüberdeckung gemessen. Ohne Werkzeuge ist es gar nicht möglich, Performanz

und Systemauslastung zu messen. Das Fehlermanagement geht auch nur mit Hilfe eines Werkzeuges. Dennoch wird vor einer übertriebenen Abhängigkeit von Werkzeugen gewarnt. Der Tester muss seine Werkzeuge beherrschen und notfalls für sie einspringen, denn er wird sich nicht für jede Testaufgabe das geeignete Werkzeug finden. Testautomaten befinden sich immer noch in einer frühen Phase. Deshalb wird das Testmanagement des Öfteren gezwungen, bei vielen Testaufgaben auf qualifizierte Menschen zurückzugreifen. Auf Affen kann es verzichten.

Das führt zu der letzten Frage, die der Testplan zu beantworten hat. Von *Wem* werden die Testaufgaben ausgeführt? Denn auch, wenn Testwerkzeuge eingesetzt werden, braucht man Menschen, um die Werkzeuge zu bedienen. Hier geht es also um die Arbeitsteilung. Wer macht was? Das Testmanagement muss die zu leistenden Testaufgaben auf das vorhandene Testpersonal verteilen. Dabei soll es auch auf die Neigungen und die Qualitäten der einzelnen Tester Rücksicht nehmen. Dies geschieht in Form einer Zuordnungsmatrix mit zwei Vektoren. In einem Vektor stehen die Aufgaben, im anderen die Namen der Mitarbeiter, die für sie verantwortlich sind [CRAI02]. So hat das Testmanagement für jede Aufgabe einen Verantwortlichen. Dies entspricht der üblichen Projektorganisation. Allerdings ist sie hier nur auf das Testprojekt beschränkt (siehe Abbildung 3.5).

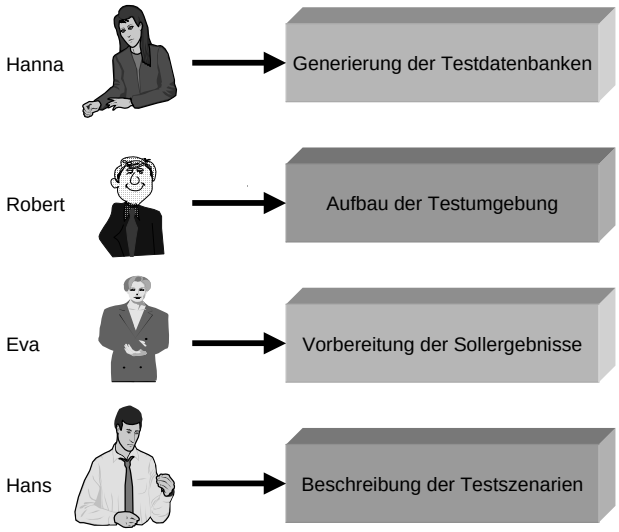

Abbildung 3.5 Zuteilung der Testaufgaben

Zusammenfassend ist der Zweck der Testplanung die oben angestellten Fragen – was, wann, wo, wie, womit und von wem – zu beantworten und verbindlich zu dokumentieren. Es soll für alle klar sein, wozu das Testen gut sein soll. Der Testplan kann als Basis für einen Testauftrag, z.B. für einen extern beauftragten, unabhängigen Test, dienen.

3.2 Voraussetzungen einer systematischen Testplanung

Eine systematische Testplanung, die die oben angestellten Fragen sinnvoll beantwortet, setzt gewisse Informationen voraus, ohne die eine genaue Planung nicht möglich ist. Wer planen will, was zu testen ist, muss wissen, welche und wie viele Fälle zu testen sind. Diese Informationen lassen sich bei neuen Systemen nur aus den Anforderungen und bei bestehenden Systemen nur aus der Systemdokumentation ableiten. Darum ist die Anforderungsanalyse bzw. die Systemanalyse eine unabdingbare Voraussetzung für die Testplanung. Sie muss vor der Testplanung stattfinden. Beim Komponententest ist es erforderlich, erst den Source-Code der Komponente zu analysieren, um die Ziele des Tests abzustecken. Beim Integrationstest ist es erforderlich, erst die Architektur des Systems zu analysieren. Beim Systemtest ist es das Konzept bzw. die Anforderungsspezifikation, welche vorher zu analysieren ist, um den Umfang des Systemtests abzustecken.

Zur Definition der Testziele – das Warum – werden zum einen die Größenmaße des Systems – zum Beispiel die Größe in Function-Points, Object-Points oder Anweisungen – und zum anderen die Fehlerrate vergangener oder ähnlicher Projekte benötigt. Für neue Systeme müssen die Größenmaße aus der Analyse der Anforderungen gewonnen werden. Für existierende Systeme, die ersetzt, saniert, migriert oder integriert werden sollten, ist es notwendig, das alte System erst durch Reverse Engineering nachzudokumentieren und anschließend die Größenmaße aus der Nachdokumentation zu gewinnen. Die Fehlerrate bisheriger Projekte lässt sich aus einer Analyse der Fehlerdatenbank ermitteln. Ohne diese Messwerte wird es kaum möglich sein, messbare Testziele zu setzen.

Die Bestimmung der Testtermine – das Wann – hängt von den Lieferterminen ab. Diese werden in dem allgemeinen Testplan festgelegt. Demzufolge setzt die Testplanung die allgemeine Projektplanung voraus. Es muss als erstes ein Projektplan vorliegen. Dann kann der Testplan darauf aufbauen.

Die Regelung der Testörtlichkeit – das Wo – setzt Kenntnisse der Verfügbarkeit voraus. Der Testmanager muss zum Zeitpunkt der Testplanung bereits wissen, mit welcher Hardware-Ausstattung getestet wird und ab wann diese Geräte verfügbar sein werden. Die Entscheidung, den Test zu verteilen oder zentralisiert durchzuführen, hängt u.a. auch von der Verfügbarkeit der Rechnerressourcen ab.

Die Wahl der richtigen Testmethode – das Wie – setzt voraus, dass der Testplaner die Art der Anwendung kennt. Er muss wissen, worauf es ankommt, wo die Risiken liegen und wo die meisten Fehler auftauchen könnten. Nur mit diesen Kenntnissen ist er in der Lage die geeignete Vorgehensweise auszuwählen und diese wiederum dem Projekt anzupassen. Hier werden also sowohl System- als auch Anwendungskenntnisse gefordert. Darüber hinaus muss der Testplaner mit den gängigen Systemtestmethoden vertraut sein. Sonst wird er nicht wissen, welche Ansätze überhaupt zur Auswahl stehen.

Zum Einsatz von Testwerkzeugen – das Womit – ist es erforderlich zu wissen, welche Werkzeuge zur Auswahl stehen und für welche Aufgaben die Werkzeuge geeignet sind. Denn im Gegensatz zum Entwurf eines neuen Systems bei dem meistens ein Werkzeug

ausreicht, braucht man beim Test eines Systems viele verschiedene Werkzeuge – eins für jede Art von Testaktivität. Der Testplaner braucht Information über die Funktionalität und die Qualität der in Frage kommenden Werkzeuge. Er muss auch wissen, wie lange man braucht, um die Werkzeuge zu beherrschen. Wenn der Termin zu knapp ist, werden die Tester nicht die Zeit haben, sich mit den Werkzeugen vertraut zu machen. Andererseits werden manche Aufgaben ohne Werkzeug nicht zu bewältigen sein. Der Testmanager steht hier vor einer sehr schweren Entscheidung.

Zur Verteilung der Testarbeit – das Wem – muss der Testmanager seine eigenen Mitarbeiter kennen und wissen, wozu sie fähig sind. Auf der einen Seite stehen die zu bewältigenden Testaufgaben, auf der anderen die verfügbaren Mitarbeiter. Man muss beide kennen um sie einander zuordnen zu können. Auch das Testen verlangt viele spezielle Kenntnisse. Der eine Tester wird sich mehr mit der Technik, der andere mehr mit der Fachlichkeit auskennen. Ein Dritter wird mehr mit den Methodiken vertraut sein. Es obliegt dem Testmanagement, jeden nach seinen Fähigkeiten einzusetzen.

Schließlich braucht der Testplaner einige Daten aus der Vergangenheit, damit er die Testaufwände und Testdauer abschätzen kann. Zum einen geht es um die bisherige Fehlerrate. Um die Anzahl der zu erwartenden Fehler hochrechnen zu können, braucht der Planer Angaben über die Anzahl gefundener Fehler in den bisherigen Projekten. Die Fehlerdichte, das heißt die Anzahl der Fehler pro 1.000 Größeneinheiten, ist ein Maß, das sich auf das neue System übertragen lässt. Dieses Maß muss durch den Grad der Testüberdeckung relativiert werden. Deshalb genügt es nicht, nur die absolute Anzahl der Fehler zu kennen, man muss auch die Größe der betroffenen Software und den Grad der Testüberdeckung kennen, wie wir bei der Schätzung der Testaufwände gleich sehen werden.

Zum anderen geht es um die Produktivität der Tester. Tester sollten Fehler finden und Vertrauen in das System aufbauen. Dazu müssen sie möglichst viele und möglichst effektive Testfälle ermitteln. Die Zeit, die sie dafür brauchen, hängt von ihrer Produktivität ab. In der Softwareentwicklung ist die Produktivität ein Maß für die Anzahl, Größeneinheiten, die ein Entwickler pro Zeiteinheit erstellen kann, zum Beispiel die Anzahl Anweisungen pro Tag oder Function-Points pro Monat. Beim Systemtest ist die Produktivität ein Maß für die Anzahl der Testfälle, die ein Tester pro Zeiteinheit ausführen kann. Natürlich wird das von Projekt zu Projekt variieren. Es hängt auch vom Grad der Testautomaten ab. Dennoch lassen sich künftige Aufwände nur aufgrund bisheriger Produktivitäten voraussagen. Der Testplaner benötigt eine Erfahrungsdatenbank, aus der das Verhältnis geleisteter Testtage zu ausgeführten Testfällen hervorgeht.

Diese beiden Metriken – die Fehlerrate und die Testproduktivität – sind unerlässliche Voraussetzungen für das, was jetzt ansteht – die Schätzung der Testaufwände.

3.3 Schätzung der Testaufwände

Ein Plan lässt sich nicht herstellen, ohne zu wissen, wie viele Kapazitäten und wie viel Zeit für eine Aufgabe benötigt werden. Denn auch wenn Zeit und Kapazität durch die Projektumstände diktiert werden, soll der Testmanager wissen, wo er sparen muss. Jede Aufgabe, auch der Systemtest, braucht ihre Zeit. Zeit ist wiederum eine Frage des Aufwands. Die Zeit lässt sich durch den Einsatz von mehr Personal kürzen, was aber nur bedingt möglich ist. Das Verhältnis zwischen Aufwand und Zeit ist nicht linear, das heißt, wir können ein Projekt von 12 Mannmonaten nicht auf 6 Mannmonate reduzieren, indem wir die Anzahl der Projektbeteiligten verdoppeln. Dies wird als der Softwaremythos bezeichnet [BROK75]. Je mehr Menschen an einem Projekt arbeiten, desto höher ist der Kommunikations-, Organisations- und Administrationsaufwand. Die Produktivität sinkt mit jedem zusätzlichen Mitarbeiter. Zeit und Aufwand sind daher nicht beliebig austauschbar, wie alle Studien zur Softwareaufwandskalkulation bewiesen haben. Sie beeinflussen sich gegenseitig. Der Schlüssel zur Ermittlung beider ist die Produktivität (siehe Abbildung 3.6).

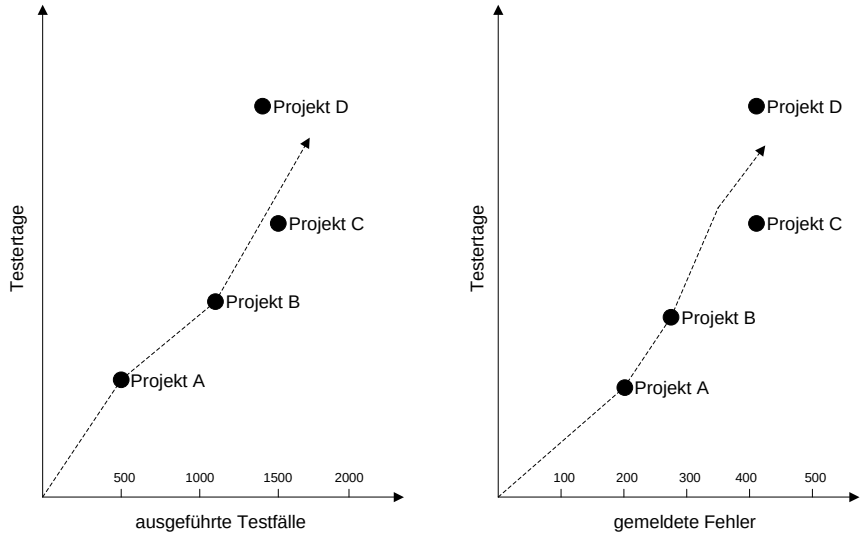

Abbildung 3.6 Ermittlung der Testproduktivität

Testproduktivität drückt sich in der Anzahl durchgeführter Testfälle pro Testertag aus. Sie hängt im Wesentlichen von dem Grad der Testautomatisierung und der Erfahrung und Motivation der Tester ab. Leider lässt sie sich nicht einfach aus der Luft holen. Produktivität will gemessen werden und zwar anhand der Statistik bisheriger Projekte. Also muss der Testplaner wissen, wie viele Testfälle es pro Projekt gegeben hat und wie viele Testertage dafür gearbeitet wurden. Dazu muss er die Testumstände bzw. die Einflussfaktoren berücksichtigen, um diese Beziehung bewerten zu können [ANSE03]. Demnach ist:

$$Testproduktivität = \frac{Testfälle}{Testertage} * Einflussfaktor$$

Die Testproduktivität zu kennen, ist leider nicht genug, um den Aufwand für ein neues Testprojekt zu kalkulieren. Es gehört mehr dazu, denn jedes Produkt hat Besonderheiten und jedes Projekt findet unter anderen Bedingungen statt. Daher empfiehlt sich die Cocomo-II Formel von Boehm für die Schätzung des Testaufwands [BOEH99]. Außer der Produktivität berücksichtigt Cocomo-II drei weitere Faktoren:

- Projektbedingungen
- Produktqualität
- Systemtyp

Die Projektbedingungen werden in einem Exponenten von 0,91 bis 1,23 zusammengefasst. Die fünf Bedingungen, die Boehm vorschlägt und die zu einem Testprojekt relativ gut passen, sind:

- Grad der Testwiederholbarkeit
- Stabilität der Testumgebung
- Kenntnis der Anwendung
- Zusammengehörigkeit des Testteams
- Reife des Testprozesses

Jede dieser Bedingungen wird auf der nominalen Skala

- Sehr niedrig = 1,23
- Niedrig = 1,10
- Mittel gut = 1,00
- Hoch = 0,96
- Sehr hoch = 0,91

eingestuft. Der Skalierungsexponent ist der arithmetische Mittelwert dieser fünf Bedingungen. Wenn also

- die Testwiederholbarkeit mittel gut ist
- die Stabilität der Testumgebung hoch ist
- die Kenntnisse der Anwendung niedrig sind
- die Zusammengehörigkeit des Testteams niedrig ist
- die Reife des Testprozesses mittel gut ist

wäre die Skalierungsexponente:

$$\frac{1,0 + 0,96 + 1,10 + 1,10 + 1}{5} = 1,03$$

Die Softwarequalität ist aus der Sicht des Tests, deren Testbarkeit, das heißt wie leicht bzw. wie schwer ist es, dieses System zu testen. Hier kommen mindestens vier Komplexitätsmaße in Frage:

- Komplexität der Benutzeroberflächen
- Komplexität der Systemschnittstellen
- Komplexität der Datenbanken
- Komplexität der Anwendungsfälle [KROP03]

Alle vier Komplexitätsmaße sind rationale Maße auf der Skala von 0,0 bis 1,0. Die Komplexität der Benutzeroberfläche ist das Verhältnis der Anzahl Steuerungseingaben zu der Summe der Oberflächeneingaben insgesamt. Je mehr Steuerungseingaben – Radio-Buttons, Menüeinträge, Checkboxen, usw. – desto komplexer der Test einer Benutzeroberfläche.

$$\frac{Steuerungseingaben}{Gesamteingaben}$$

Die Komplexität der Systemschnittstellen ist das Verhältnis der Anzahl verschiedener Datentypen zur Anzahl der Datenelemente pro Schnittstelle. Je mehr verschiedene Datentypen es gibt, desto aufwändiger ist es, eine Systemschnittstelle zu testen.

$$\frac{Datentypen}{Datenelemente}$$

Die Komplexität der Datenbanken ist das Verhältnis der Schlüssel (Primär- und Fremdschlüssel) und der Indizes zu der Gesamtanzahl aller Attribute. Je mehr Attribute einer Datenbank Teil von Primär- und Fremdschlüsseln sind und je mehr Attribute indiziert sind, umso schwieriger ist die Testdatenaufbereitung.

$$\frac{Schlüsselattribute + Indizes}{Attribute}$$

Die Komplexität der Anwendungsfälle ist schließlich das Verhältnis der Bedingungen zu den Aktionen. Je mehr Bedingungen einen Vorgang steuern, desto größer ist der Aufwand, den Vorgang bzw. den Anwendungsfall zu testen.

$$\frac{Bedingungen}{Aktionen}$$

Möglicherweise wird der Testplaner nicht genug detaillierte Information über das System haben, um diese Zahlen zu ermitteln. In dem Fall kann er die Komplexität mit einer nominalen Skala schätzen.

- Sehr hoch = 0,8
- Hoch = 0,6
- Durchschnittlich = 0,5
- Niedrig = 0,4
- Sehr niedrig = 0,2

Die Systemtestkomplexität ist der arithmetische Mittelwert der einzelnen Testkomplexitäten. Der Testbarkeitsfaktor ist die gemessene oder geschätzte Komplexität dividiert durch die mittlere Komplexität = 0,5 [SNEE06].

$$Testbarkeitsfaktor = \frac{gemessene\ Komplexität}{mittlere\ Komplexität}$$

Der Systemtyp hat nach Boehm eine große Auswirkung auf den Aufwand. Er teilt die Systeme in vier Kategorien ein:

- Standalone-Systeme-Single-User-Systeme
- Verteilte Multi-User-Systeme
- Integrierte, verteilte Systeme mit Multi-User
- Embedded-Systeme

Standalone Systeme wiegen 0, verteilte Systeme 1, integrierte Systeme 2 und Embedded Systeme 4. Wesentlich ist, dass man immer mit dem gleichen Systemtyp vergleicht. Der Systemtyp darf nur verwendet werden, wenn das geplante System von einem anderen Typ ist als das System von dem die Produktivität entnommen wird. Mit diesen Parametern lässt sich der Testaufwand mit folgender Gleichung schätzen.

$$Testaufwand = Systemtyp * \left(\frac{Testfälle}{Testproduktivität} \right)^{Skalierungsexponente} * Testbarkeitsfaktor$$

Angenommen, es werden 1.600 Testfälle anhand der Konzeptanalyse gezählt. Im letzten ähnlichen Projekt wurden 1.200 Testfälle in 125 Personentagen getestet. Das ergibt eine Testproduktivität von 1.200/125 = 10 Testfällen pro Personentag.

Die Skalierungsexponente für dieses Projekt wird wie folgt eingeschätzt.

- Testwiederholbarkeit = mittel = 1,00
- Stabilität der Testumgebung = hoch = 0,96
- Kenntnis der Anwendung = niedrig = 1,10
- Zusammengehörigkeit des Testteams = hoch = 0,96
- Testprozessreife = hoch = 0,96

Daraus folgt der arithmetische Mittelwert 0,99.

Die Analyse der Software deutet auf eine überdurchschnittliche Komplexität dieses Systems von 0,60 hin. Durch die Komplexität des Vergleichsystems von 0,5 ergibt dies eine gesteigerte Testkomplexität von 1,2. Schließlich handelt es sich um denselben Systemtyp wie bisher – ein verteiltes Client/Server-System, also Typ 1. Demzufolge ist der geschätzte Testaufwand:

$$Testaufwand = 1 * \left(\frac{1600}{10} \right)^{0,99} * 1,2 = 182 Personentage$$

Die besseren Projektbedingungen, die durch den Exponent 0,99 erfasst sind, werden durch die höhere Komplexität des neuen Systems ausgeglichen, so dass am Ende die Testproduktivität bei circa 10 Testfällen pro Testertag gleich bleibt. Dieser Aufwand umfasst genau jene Testaktivitäten, die die alte Produktivitätsmessung umfasst hatte. Wenn dort der Aufwand für die Testplanung nicht mit erfasst wurde, wird er hier auch nicht berücksichtigt.

Darum ist es erlässlich bei der Produktivitätsmessung konsistent zu bleiben und immer die gleichen Aufwände zu erfassen. Nur so können wir eine genaue Hochrechnung für künftige Projekte sichern [SNEE03].

3.4 Schätzung der Testdauer

Nach dem COCOMO-Modell von Boehm besteht ein algorithmisches Verhältnis zwischen dem Aufwand für ein Projekt und der Dauer des Projekts. Die Gleichung für die Dauer einer Softwareentwicklung war in COCOMO-I:

$$TDEV = 2{,}5 * (Aufwand)^{Exponent}$$

wobei der Exponent folgendermaßen gewählt wird:

- 0,38 für Standalone-Systeme
- 0,35 für verteilte Systeme
- 0,32 für Embedded-Realtime-Systeme [BOEH84]

In dem neuen COCOMO-II Modell ist die Gleichung:

$$TDEV = (C' * (PM)^{F}) * \left(1 - \frac{SCED\%}{100}\right)$$

wobei

$$F = (D + 0{,}2 * (E-B))$$

In dieser Gleichung sind die Parameter wie folgt definiert:

- B = Untergrenze der Skalierungsexponente = 0,91
- C = Multiplikator = 3,67 für Neuentwicklung
- D = Zeitbasiskoeffizient = 0,28 für Neuentwicklung
- E = Skalierungsexponent der Entwicklung = 0,91 : 1,23
- F = Skalierungsexponent für die Projektdauer
- PM = geschätzter Aufwand in Personenmonaten
- SCED % = % Kompression des Projekts
- TDEV = Zeit für die Entwicklung [BOEH00]

Nach der ursprünglichen COCOMO-I Formel wäre die Dauer eines Testprojekts mit dem geschätzten Aufwand von 182 Personentagen bzw. 9 Personenmonaten.

$$TDEV = 2{,}5 * (9)^{0{,}35} = 5{,}4 Monate$$

Das wäre die minimale Dauer des Projekts 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 beliebig viele Tester nebeneinander arbeiten. Damit lässt sich sie Dauer eines Testprojekts

zwar nicht ganz, aber doch stark komprimieren. Daher ist die COCOMO-II Zeitformel für den Test besser geeignet.

Danach ist die Dauer des Projekts wie folgt zu berechnen:

$$F = (0{,}28 + 0{,}2 * (0{,}96 - 0{,}91)) = 0{,}29$$

$$TDEV = 3{,}67 * (9)^{0{,}29} = 6{,}9 Monate$$

Jetzt kommt aber der Schedule-Compression-Faktor dazu, und dieser ist beim Testen hoch. Wir können die Zeit zum Testen durch die Verteilung der Arbeit auf mehrere Tester um mindestens 60% verkürzen. Das heißt:

$$TDEV = 6{,}9 * \left(1 - \frac{60}{100}\right) = 2{,}8 Monate$$

Demnach könnten wir z.B. fünf Tester einsetzen. Die Anforderungsanalyse und die Testplanung werden von zwei Testern bewerkstelligt. Danach kämen drei weitere Tester dazu, die zusammen mit den ersten zwei Testern die Testfälle spezifizieren, die Testdaten generieren, die Testumgebung aufbauen, den Test durchführen und die Testergebnisse auswerten. Das Ganze könnte sich innerhalb von drei Monaten abspielen. Es ist schließlich eine Frage der Projektorganisation.

3.5 Testprojektorganisation

Testen ist in einem starken Maße von der Umgebung abhängig. Spezifizieren, Entwerfen und Codieren sind Aktivitäten, die auch ohne Rechnernutzung ausgeführt werden könnten bzw. der Rechner wird nur zum Zeichnen und Texte editieren verwendet – Aufgaben, die im Prinzip per Hand zu erledigen wären. Erst zum Kompilieren wird wirklich ein Rechner benötigt. Am Anfang der Softwareentwicklung war dies auch der Fall. Zum Testen war man aber schon immer auf den Rechner angewiesen.

3.5.1 Organisation der Testressourcen

Es gab Zeiten, in der Rechenkapazität nicht im Überfluss vorhanden war. Da konnte es vorkommen, dass die Rechnernutzung für Tests zum Engpass für die Projekte wurde. Es durfte nur eine begrenzte Anzahl von Menschen eine begrenzte Zeit mit dem Rechner arbeiten. Falls der Test im Batch-Betrieb lief, waren die Tester gezwungen, ihre Tests als Batchjobs auf Band, Lochkarten oder Lochstreifen vorzubereiten und dann bei den Operatoren abzugeben. Auf diese Weise konnte man pro Tag maximal drei Tests fahren. Notfalls konnte ein Projekt den Rechner für sich bekommen, um mehrere Tests hintereinander zu fahren, aber dies war nur nachts möglich. Bei seinen ersten Testprojekten Ende der 70er Jahre durfte der Autor nur nachts arbeiten. Bei Siemens in München von 16.00-24.00 Uhr und bei der Spardat in Wien von 18.00-02.00 Uhr. Ergo musste das Testen sehr strikt organisiert werden, um die beschränkte Rechnerzeit optimal auszunutzen. Oft musste die

Rechnerzeit Monate im Voraus vorbestellt werden. Heute sieht das anders aus. Rechnerkapazität ist im Überfluss vorhanden, dafür gibt es andere Engpässe.

Der Hauptengpass beim heutigen Testen ist nicht die Rechnernutzung, sondern die Datennutzung. Zum Testen braucht man einen Bildschirmarbeitsplatz, eine Verbindung zum Server, die Software auf dem Server und die Daten auf dem Server. Die Serversoftware ist in der Regel multiserverfähig und wenn nicht, lässt sie sich für jeden Test duplizieren. Leider trifft dies für die Daten nicht immer zu. Für den Test braucht jeder Tester seine eigenen Daten, damit er den Vorzustand jederzeit wieder herstellen kann und damit er den Nachzustand validieren kann, ohne die Gefahr, dass jemand anderes seine Daten zwischendurch absichtlich oder unabsichtlich verändert. Es ist deshalb dringend notwendig, die Testdaten für jeden Tester getrennt zu halten. Dies klingt banal, ist aber für viele Projekte ein großes Problem, weil die Plattenspeicherkapazität nicht ausreicht oder weil das Datenbanksystem nur eine Kopie einer Datenbank zulässt [MOSL00].

Die Verfügbarkeit der Testdaten ist ein organisatorisches Problem, das bereits bei der Testplanung zu berücksichtigen ist. Der Testmanager muss dafür sorgen, dass jeder Tester einen Testarbeitsplatz mit Verbindung zum Server hat, dass der Server genug Hauptspeicherkapazität für die gleichzeitige Bedienung aller Tester hat, dass das Kommunikationsnetz die Testlast verträgt und vor allem, dass genug externe Speicherkapazität vorhanden ist, um für jeden Tester eine eigene Testdatenbank zu haben. Die Datenbankkommunikation muss auch gewährleisten, dass das Datenbanksystem mehrere Kopien der gleichen Datenbank nebeneinander verwalten kann. Die Ressourcenplanung ist also ein wichtiger Bestandteil der allgemeinen Testplanung und darf nicht vernachlässigt werden. Es kommt darauf an, Kapazitätsengpässe bei der Testdurchführung zu vermeiden. Dies ist am besten zu erreichen, wenn der Kapazitätsbedarf rechtzeitig geregelt wird (siehe Abbildung 3.7).

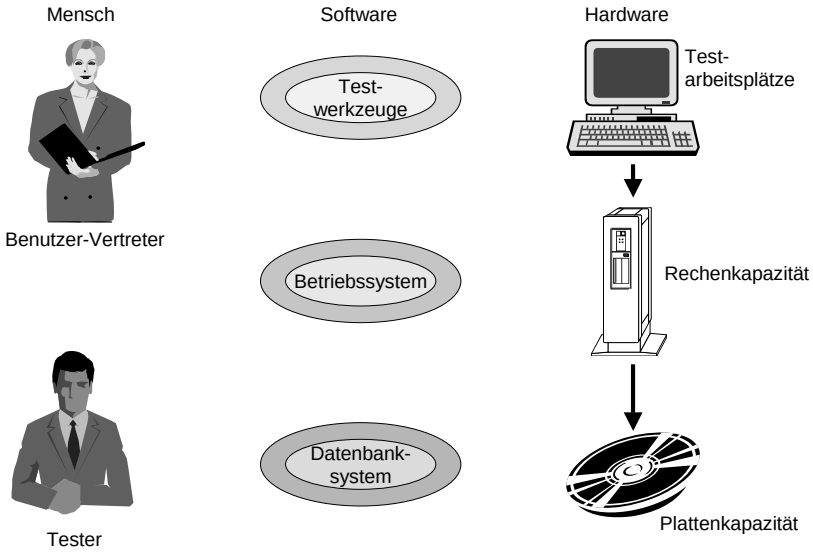

Abbildung 3.7 Einteilung der Testressourcen

3.5.2 Organisation des Testpersonals

Die Bereitstellung der nötigen Maschinenressourcen ist nur die eine Seite der Testorganisation. Die andere Seite ist die Bereitstellung der nötigen Humanressourcen. Testen ist noch immer eine menschliche Aktivität, es lässt sich nur bedingt automatisieren. Menschen sind erforderlich, um die Testfälle zu spezifizieren, die Testdaten zu ermitteln, die Testumgebung aufzubauen, den Test durchzuführen und die Testergebnisse zu kontrollieren. Auch wenn Testwerkzeuge zu diesem Zweck eingesetzt werden, braucht man Menschen, um die Werkzeuge zu bedienen. Hier stellt sich die Frage, was für Menschen gebraucht werden und wie viele. Ein Entwickler ist nicht immer als Tester geeignet. Viele Entwickler haben eine Abneigung gegen Testen und sind dafür nicht zu begeistern. Sie haben weder die nötige Geduld und Ausdauer, noch haben sie die nötige Aufmerksamkeit für Details. Außerdem fehlen ihnen die Fachkenntnisse. Dafür hätten sie die erforderlichen technischen Kenntnisse und Fähigkeiten. Sachbearbeiter von der Fachabteilung bringen die Fachkenntnisse der Anwendung mit. Sie haben auch meistens die Ausdauer und die Aufmerksamkeit für die Detailfragen. Was ihnen fehlt, sind die erforderlichen technischen Kenntnisse und Fähigkeiten, sowie das Vermögen, sich in die Feinheiten der Softwarekonstruktion hineinzudenken. Demzufolge können sie sich schwer vorstellen, welche Art von Fehler auftreten könnte. Systemanalytiker bzw. diejenigen, die zwischen den Anwendern und den Entwicklern vermitteln sollten, sind die am meisten geeigneten Kandidaten. Sie besitzen die technischen Fähigkeiten und haben die besten Kenntnisse über das System. Immerhin haben sie die Anforderungen spezifiziert, die jetzt getestet werden sollten. Was ihnen fehlt, ist die Geduld, die Aufmerksamkeit für Details und die Ausdauer, die ein Tester braucht. Hinzu kommt, dass sie sich mit Routinetestarbeiten nicht gerne abgeben. Analytiker sind kreative Menschen, die nur ungern wiederholte Tätigkeiten ausführen. Summa Summarum, ist es nicht einfach, geeignete Tester zu finden. Im Grunde genommen braucht man eine Mischung aus Entwickler, Analytiker und Anwender, die noch dazu in der Anwendung gängiger Testmethoden und Testwerkzeuge geschult ist. Da diese Mischung nicht in einem Menschen zu finden ist, ist der Testmanager gezwungen, ein gemischtes Testteam zu bilden, ein Team, das aus Entwicklern, Analytikern, Anwendern und professionellen Testern besteht.

Oft gibt es im Test auch Tätigkeiten mit monotonem wiederholendem Charakter, wie zum Beispiel das Erfassen hunderter Testdatensätze oder die fünfte Wiederholung ein und desselben simplen Testfalles. Dort, wo die Testautomatisierung noch nicht entsprechend fortgeschritten ist, wird auch dafür Personal benötigt. Dieses Personal kann von außen geholt werden, zum Beispiel von einer Personal-Leasing-Agentur oder Studenten von der lokalen Universität. Dies sollte jedoch nur als vorübergehende Lösung angesehen werden. Die angestrebte Lösung ist die Automatisierung des Tests [FEWS99].

3.6 Testrisikoanalyse

Der Systemtest ist, wie andere IT-Unterfangen, nicht ohne Risiken. Es kann immer etwas vorkommen, das den Testtermin gefährdet. Die häufigsten Risiken sind:

- Die vorliegenden Anforderungsdokumente sind mangelhafter als erwartet.
- Die zu testende Software wird nicht rechtzeitig geliefert.
- Die Testwerkzeuge funktionieren nicht wie erwartet.
- Die geforderten Rechnerressourcen werden nicht rechtzeitig bereitgestellt.
- Die Speicherkapazität reicht nicht aus.
- Performanzengpässe treten auf.
- Wichtiges Personal fällt aus.
- Die Software ist viel fehlerhafter als erwartet, so dass der Test mehrmals wiederholt werden muss [COHE04].

Jedes Risiko kann das Projekt verzögern, und mehrere Risiken zusammen können den Aufwand und die Dauer des Tests vervielfachen. Planung heißt auch vorbauen. Der Testmanager muss bei der Terminzusage einen Puffer vorsehen, um eventuelle Risiken zumindest teilweise auffangen zu können. Für die Schätzung der Pufferzeit ist zu empfehlen, einen Risikozuschlag auszurechnen. Dafür muss jedes einzelne Risiko mit der Risikowahrscheinlichkeit und der Risikoaussetzung multipliziert werden. Die Risikowahrscheinlichkeit entspricht der Wahrscheinlichkeit in Prozent, dass dieses Risiko auftritt. Die Risikoaussetzung ist die prozentuale Zunahme der Projektdauer, zum Beispiel, dass die Zeitdauer um 25% verlängert wird. Die Summe der Produkte dieser beiden Faktoren für alle Risiken ergibt den Gesamtrisikofaktor, zum Beispiel 0,64. Dieser wird mit eins addiert und mit der Testdauer in Wochen oder Monaten multipliziert um die potentielle Zeitverschiebung zu errechnen.

$$Zeitverschiebung = 1 + \Sigma\, Risiko\ (RA * RW)$$

Bei einem Webprojekt hat einer der Autoren als Testplaner eine Zeitverschiebung von 3 Wochen vorgesehen (siehe Abbildung 3.8, nächste Seite). Diese Verschiebung ist tatsächlich eingetreten und zwar genau um 3 Wochen. Es obliegt dem Testmanagement, risikomindernde Maßnahmen mit deren Hilfe die Zeitverschiebung reduziert werden könnte, vorzuschlagen. Beispiele dafür wären:

- Reservetester
- zusätzliche Rechenressourcen
- weitere Testwerkzeuge usw.

Der Testmanager hat jedoch auf viele Risiken, wie die Fehlerhäufigkeit, die Lieferpünktlichkeit und die Performanzengpässe keinen Einfluss. Er kann sie allenfalls zu Protokoll geben, um sich später zu rechtfertigen, wenn sie eintreten. Denn nach Murphys Law werden sie eintreten. Ein Softwaretester, und erst recht ein Softwaretestmanager, muss von Natur aus ein pessimistischer Mensch sein, der trotz aller Widrigkeiten des Testgeschäfts die Fassung behält.

Die offensichtlichen Risiken des Tests sind:
• Verschiebungen der Lieferungen mit einer Wahrscheinlichkeit von 40%
• Hohe Fehlerhaftigkeit der gelieferten Software mit einer Wahrscheinlichkeit von 30%
• Technische Probleme mit der Testumgebung mit einer Wahrscheinlichkeit von 25%
• Probleme mit den Testwerkzeugen mit einer Wahrscheinlichkeit von 20%
• Ausfall wichtiger Projektmitarbeiter mit einer Wahrscheinlichkeit von 15%

Die Auswirkung der Terminverschiebung auf die Testdauer könnte zwischen 2 bis 12 Wochen liegen.
Die Fehlerhaftigkeit der Software könnte 1 bis 6 Wochen kosten.
Technische Probleme mit der Testumgebung können 1 bis 4 Wochen kosten.
Probleme mit den Testwerkzeugen können ebenfalls 1 bis 4 Wochen kosten.
Der Ausfall vom Personal kann das Projekt nicht mehr als 2 Wochen zurückwerfen,
da genüg Reserven vorhanden sind und Tester sich mit geringem Aufwand ersetzen lassen.

Eine Analyse dieser Risiken führt zu folgendem Risikozuschlag:

Terminverschiebung	= [Max – Min Zeitverlust]/2 = 5 Wochen * 0,4 = 2 Wochen
Fehlerhaftigkeit	= [Max – Min Zeitverlust]/2 = 2,5 Wochen * 0,3 = 0,75 Wochen
Testumgebung	= [Max – Min Zeitverlust]/2 = 1,5 Wochen * 0,25 = 0,38 Wochen
Werkzeugprobleme	= [Max – Min Zeitverlust]2 = 1,5 Wochen * 0,20 = 0,30 Wochen
Mitarbeiterausfall	= [Max – Min Zeitverlust]/2 = 1 Woche * 0,15 = 0,15 Wochen

Der Maximum Zeitverlust wäre die Summe aller Höchstgrenzen = 28 Wochen, bzw. 7 Monate.
Der wahrscheinlichste Zeitverlust wäre die Summe der einzelnen Risikofaktoren = 3,58 Wochen

Abbildung 3.8 Beispiel einer Testrisikoanalyse

3.7 Festlegung der Testendekriterien

Eine unvermeidliche Aufgabe eines jeden Testprojektleiters ist es, die Testendekriterien festzulegen. Hier geht es um die lästige Frage, wann der Test zu Ende ist. Es gibt Projekte, die ewig dahin laufen, weil keiner entscheiden kann, ob der Test ausreichend ist. Diesen Zustand muss man von vornherein vermeiden. Dazu braucht das Testprojekt messbare Endkriterien. Vorneweg ist festzustellen, dass kein komplexes Softwaresystem je voll getestet werden kann. Auch, wenn es theoretisch möglich wäre, würde es zu lange dauern und zu viel kosten. Darüber ist schon genug geschrieben worden [MUSA89]. Software ist wie das Universum in dem wir leben, nach Einstein prinzipiell endlich, aber für alle praktischen Zwecke endlos. Ergo können wir einen vollständigen, 100%igen Test von Anfang an ausschließen. Umso mehr brauchen wir realistische Ziele für unseren Systemtest. Ein solches Ziel wäre die Fehlerüberdeckung. Aufgrund der Erfahrung mit ähnlichen Projekten aus der Vergangenheit, wird die Anzahl der Fehler hochgerechnet. Man nimmt die alte Anzahl der Fehler und justiert sie durch die Größe und Komplexität des neuen Systems. Hatte das letzte Projekt dieser Art zum Beispiel 400 gemeldete Fehler bei einer Größe von 5.000 Function-Points und einer Systemkomplexität von 0,54, wären beim neuen System, das auf 6.000 Function-Points mit einer Komplexität von 0,6 geschätzt wird, 533 Fehler zu erwarten. Erst justieren wir die Größe des letzten Systems durch seine Komplexität.

$$5000 * \frac{0,54}{0,50} = 5400$$

Diese Größe dividieren wir durch die Anzahl der Fehler, um die Fehlerdichte zu ermitteln.

$$\frac{400}{5400} = 0{,}074$$

Im nächsten Schritt justieren wir die geschätzte Größe des neuen Systems durch die geschätzte Komplexität.

$$6000 * \frac{0{,}6}{0{,}5} = 7200$$

Diese justierte Systemgröße wird jetzt mit der Fehlerdichte multipliziert, um die geschätzte Fehlerzahl zu errechnen.

$$7200 * 0{,}074 = 533 \text{ Fehler}$$

Das eigentliche Testendekriterium wäre dann so lange zu testen, bis ein gewisser Prozentsatz der geschätzten Fehler aufgedeckt wird. Diese geschätzte Fehlerzahl muss auch nicht starr bleiben. Sie sollte dynamisch angepasst werden. Wenn der Testmanager beim Test der ersten Teilsysteme oder der ersten Version feststellt, dass die Fehlerdichte höher ist, zum Beispiel 0,09 statt 0,074, kann er die Anzahl der erwarteten Fehler hochschrauben, zum Beispiel von 533 auf 648.

Fehlerüberdeckung ist nicht das einzige Endekriterium. Es gibt auch die Abdeckung der Funktionalität, die Abdeckung der Daten oder die Abdeckung des Codes. Daraus folgen die drei Überdeckungskriterien:

- Funktionsüberdeckung
- Datenüberdeckung
- Codeüberdeckung

Funktionsüberdeckung ist der Prozentsatz, zu dem die spezifizierten Funktionen getestet werden. Aus der Anforderungsanalyse geht hervor, welche Anwendungsfälle mit welchen Aktionen und Bedingungen zu testen wären. Ein messbares Testziel wäre es, einen vereinbarten Prozentsatz jener Funktionen auszuführen, zum Beispiel 90%.

Datenüberdeckung ist der Prozentsatz zu dem die spezifizierten Datenattribute in den Datenbanken, Systemschnittstellen und Benutzeroberflächen getestet werden. Für jedes zu testende Attribut sollte es eine Assertion geben, in der die Pre- und Postwerte des Attributs spezifiziert sind. Die Datenüberdeckung ist demnach der Prozentsatz der ausgeführten Assertions relativ zur Anzahl aller Attribute. Hier lässt sich auch ein gewisser Prozentsatz als Testendekriterium vereinbaren.

Schließlich gibt es die altbekannte Codeüberdeckungsmaße. Es sollte nicht das Ziel eines Systemtests sein, 90% Zweigüberdeckung zu erreichen. Bei einem System mit einem hohen Anteil wiederverwendetem Code, wird ein Großteil des Codes nie benutzt. Er ist nur deshalb da, weil es zu schwierig ist, den benutzen Code vom unbenutzten Code zu trennen. Die Codeüberdeckung ist eher ein Ziel des Unittests [BIND99]. Was jedoch gemessen werden kann, ist die Modul- bzw. die Methodenüberdeckung. Wie dies zu bewerkstelligen ist, wird erst später bei der Testauswertung behandelt. Es genügt hier einen gewissen Mo-

dulüberdeckungsgrad als Testziel zu vereinbaren und dies auch konsequent anzustreben. Wenn er erreicht ist, ist das Testendekriterium erfüllt (siehe Abbildung 3.9).

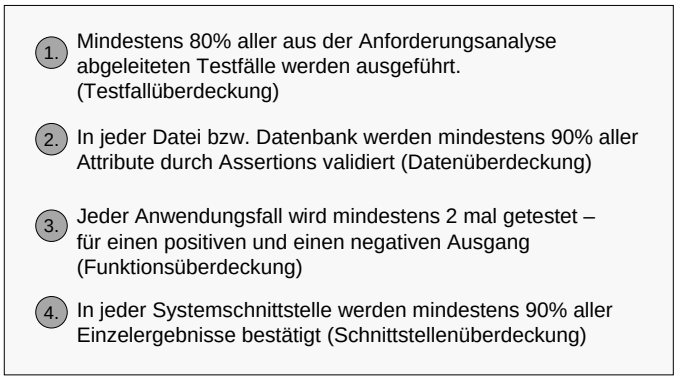

Abbildung 3.9 Mögliche Testendekriterien

3.8 Gestaltung des Testplans

Der letzte Akt der Testplanung ist die Verfassung des Testplandokuments. Bis dahin soll der Testprojektleiter, genauso wie jeder Autor einer wissenschaftlichen Arbeit, alle Fakten zusammengetragen haben.

- Er hat die Testziele gesetzt.
- Er hat die Testobjekte und Testfunktionen erkannt.
- Er hat die erforderlichen Soll-Testfälle gezählt.
- Er hat die Testressourcen identifiziert.
- Er hat den Testaufwand und die Testdauer geschätzt.
- Er hat die Testrisiken analysiert.
- Er hat einen Personenplan aufgestellt.
- Er hat die Testendekriterien aufgestellt.

Das Letztere ist eine unbedingte Voraussetzung für die Testplanung. Der Testprojektleiter muss noch mit dem Gesamtprojektleiter und dem zuständigen Produktmanager darüber einig werden, wann der Test ausreichend ist bzw. wann man damit aufhören darf. Einige mögliche Endkriterien sind im letzten Abschnitt vorgestellt worden. Sie reichen von einer gewissen Code-, Daten- oder Funktionsüberdeckung bis hin zur Findung einer bestimmten Anzahl Fehler. Hinzu kommen die Einschätzungskriterien, wie die Errechnung der Budgetgrenze und die Überschreitung des Endtermins. Wichtig ist, dass die Verantwortlichen unter sich einig sind, welche Kriterien für diesen Test gelten sollen. Wenn dies feststeht, kann mit der Verfassung des Testplans begonnen werden. Als Muster für den endgültigen Testplan sollte das Gliederungsschema aus dem ANSI/IEEE Standard 829 verwendet werden und zwar nicht nur, weil dieser die herrschende internationale Norm ist, sondern auch,

weil er sich in der Praxis bewährt hat und vielfach zitiert wird [IEEE829]. Dieser Standard sieht ein Dokument mit 16 Abschnitten vor (siehe Abbildung 3.10).

Abbildung 3.10 Aufbau des Testplans (nach ANSI/IEEE-Standard 829)

3.8.1 Testkonzept-ID

Hierbei handelt es sich um ein eindeutiges Kennzeichen des Dokuments, um es einerseits mit einer Suchfrage wieder zu finden und andererseits von anderen Dokumenten aus zu referenzieren ist unabdingbar. Im Falle eines XML-Dokuments wird dies Attribut „ID“ haben.

3.8.2 Testeinführung

Dies ist eine kurze Zusammenfassung der zu testenden Hard- und Softwareeigenschaften und eine Zielerklärung, was mit dem Test erreicht werden soll, zum Beispiel, welche Risiken zu vermeiden sind. Falls es mehrere Testpläne gibt, die hierarchisch geschachtelt sind, muss jeder untergeordnete Testplan den übergeordneten Testplan referenzieren. In der Einleitung soll auch etwas über den Umfang des zu testenden Systems stehen. Hier soll es auch Hinweise auf für den Test relevante Dokumente geben. Diese sind:

- Spezifikationsdokumente bzw. das Fachkonzept
- Entwurfsdokumente bzw. das technische Konzept
- Benutzerhandbücher bzw. die Bedienungsanleitung
- Betriebshandbücher
- Installationsunterlagen

3.8.3 Zu testende Objekte

Hier sind sämtliche zu testende Objekte – Oberflächen, Schnittstellen, Datenbanken und Programme einschließlich der Versionsnummer bzw. Revisionsnummer – aufzuführen. Objekte, die ausdrücklich nicht getestet werden sollen, sollten ebenfalls aufgeführt werden.

3.8.4 Zu testende Funktionen

Hier wird die Benennung sämtlicher zu testender Funktionen und Funktionskombinationen verlangt. Falls das Anforderungsdokument nach Anwendungsfällen gegliedert ist, sind die Anwendungsfälle die zu testenden Funktionen. Für jede zu testende Funktion bzw. Funktionskombination sollte es einen Querverweis auf das dazugehörige Testskript oder die Testfallbeschreibung geben. In einem XML-Dokument wird dies das Attribut „href“ sein. In der Praxis kann diese Liste sehr lang werden, vor allem bei großen Systemen.

3.8.5 Nicht zu testende Funktionen

Besonders zu erwähnen sind jene Funktionen, die – aus welchen Gründen auch immer – nicht zu testen sind. Sie sind von dem Systemtest ausgeklammert. Mit dieser Ausklammerung entzieht sich die Systemgruppe der Verantwortung für die Funktionsfähigkeit dieser Funktionen bzw. erzwingt eine Festlegung an anderer Stelle, wer für die Qualitätssicherung dieser Funktionen verantwortlich sein sollte.

3.8.6 Testvorgehensweise

An dieser Stelle erfolgt eine Beschreibung des allgemeinen Testvorgehens bzw. der Teststrategie. Die Testphasen und Testaktivitäten sind hier anzugeben. Auch die einzusetzenden Techniken und Testwerkzeuge sind zu spezifizieren. Das Vorgehen sollte so weit beschrieben werden, dass es möglich ist, die einzelnen Testaufgaben zu identifizieren, zu schätzen und zuzuteilen. Weiterhin sind alle bedeutenden und bekannten Einschränkungen des Testvorgangs aufzuführen, wie zum Beispiel Verfügbarkeit der Testobjekte, Ressourcen und Termine.

3.8.7 Testendekriterien

Hier geht es um Messkriterien für die Beendigung des Tests bzw. um die Frage, wann der Test fertig ist? Es müssen konkrete, messbare Ziele gesetzt werden, Ziele, wie das Erreichen eines bestimmten Testüberdeckungsgrades, die Aufdeckung einer bestimmten Anzahl Fehler oder die Ausführung einer bestimmten Anzahl der Testfälle. Die Erfüllung dieser Endekriterien muss mit einem einfachen Ja oder Nein zu beantworten sein. Entweder ist das Kriterium erfüllt oder nicht. Nur, wenn das Kriterium erfüllt ist, kann der Test als beendet betrachtet werden. Sonst gilt der Test als nicht abgeschlossen oder abgebrochen.

3.8.8 Testabbruchkriterien

An dieser Stelle sind jene Kriterien zu beschreiben, die eine Unterbrechung aller oder eines Teils der Testaktivitäten bedingen. Typische Abbruchkriterien sind:

- Schwerwiegende Fehler, die den Test behindern
- Zu viele Fehler
- Ungenügende Hardwarekapazitäten
- Die vorausgesetzten Softwareprodukte funktionieren nicht.
- Die Frist für den Test ist überschritten.

Falls ein oder mehrere dieser Kriterien auftreten, wird der Test zunächst unterbrochen. Wieder aufgenommen wird er erst, wenn die Probleme behoben sind, oder wenn eine Umgehungsmöglichkeit besteht. Hier kann auch festgelegt werden, unter welchen Bedingungen ein Test wieder auf zusetzen ist.

3.8.9 Testergebnisse

Die Testergebnisse sind jene Produkte, die das Systemtestteam im Laufe des Tests zu liefern hat. Daran wird der Fortschritt des Tests gemessen. Die Liste beginnt mit dem Testplan und endet mit dem Testabschlussbericht. Die von der Norm vorgeschlagenen Ergebnisse sind:

- Testplan
- Testentwurf bzw. das Testkonzept
- Testfallspezifikation
- Testskripte
- Testausführungsprotokolle
- Fehlerberichte
- Testabschlussbericht

Es können weitere Ergebnisse dazu kommen, aber diese Produkte sind das Minimum, was ein Testprojekt anstreben sollte.

3.8.10 Testaufgaben

Neben den Ergebnissen, die vom Test zu produzieren sind, folgen hier die Aufgaben, die der Test zu erledigen hat. Es handelt sich um alle Tätigkeiten, die zur Vorbereitung, zur Ausführung und zur Auswertung des Tests nötig sind. Typische Testaufgaben sind:

- Erstellung des Testplans
- Entwurf eines Testkonzepts
- Spezifikation der Testfälle
- Generierung der Testskripte
- Bereitstellung der Testdaten
- Aufbau der Testumgebung

- Durchführung diverser Tests
- Protokollierung der Tests
- Fehlerberichterstattung
- Fehlerverfolgung
- Verfassung der Testberichte

Dies sind nur die wichtigsten Testaufgaben. In der Praxis wird es etliche Ausprägungen dieser Aufgabentypen geben. Anzustreben ist es, Aufgaben zu definieren, die von einer oder höchstens zwei Personen in wenigen Tagen zu erledigen sind. Je feiner die Aufgaben definiert sind, umso leichter wird es, sie zu kontrollieren.

3.8.11 Testumgebung

Hier werden die erforderlichen Eigenschaften der Testumgebung festgelegt. Neben den gewünschten Hardwaregeräten wie Testarbeitsplätzen, Vermittlungsrechnern und Servern, Druckern und Speichergeräten werden die nötigen Softwareprodukte angegeben: Betriebssysteme, Datenbanksysteme und Middleware, sowie Office-Produkte und spezielle Frameworks. Sollten Testwerkzeuge benötigt werden, sind diese auch hier zu verlangen. Ebenso werden hier räumliche Anforderungen gestellt, so zum Beispiel zusätzliche Testräume mit Netzanschlüssen oder Funkverbindungen. Für alle Materialien, die im Moment noch nicht zur Verfügung stehen, ist eine Quelle zu nennen.

3.8.12 Testverantwortlichkeiten

Hier sind die verantwortlichen Gruppen bzw. Dienststellen für Management, Design, Vorbereitung, Ausführung, Beglaubigung, Überprüfung und Support zu definieren. Soweit bekannt, werden die einzelnen Testaufgaben verantwortlichen Personen oder Stellen zugewiesen. Es geht also auch um die Organisation des Testprojekts und die Zuteilung der Rollen.

3.8.13 Testpersonalbedarf

Ausgehend von den Rollen im Testverfahren wird hier ein Personalbedarfsplan aufgestellt. Zur Erfüllung der einzelnen Rollen werden gewisse Fähigkeiten verlangt, so zum Beispiel als Datenbankspezialist, als Netzspezialist, als Sicherheitsexperte oder als XML-Kenner. Zusätzlich zu diesen speziellen Kenntnissen, die vorausgesetzt werden, wird ein Ausbildungsplan für das Testpersonal vorgelegt, damit die Tester besser in der Lage sind ihre Rollen zu erfüllen. Dazu gehören die Schulung der Testmethodik und der Bedienung der Testwerkzeuge, sowie auch Schulungen bezüglich der Testobjekte, der zu testenden Applikationen selbst.

3.8.14 Testzeitplan

In dem Zeitplan werden die Meilensteine im Testprojekt, sowie die Lieferungstermine der Testobjekte festgehalten. Auch zusätzliche Meilensteine können aufgeführt werden. Für jede Testaufgabe ist der Zeitbedarf zu schätzen, so dass ein Netzplan oder PERT-Diagramm erstellt werden kann, aus denen der kritische Pfad hervorgeht. Zur Darstellung des Zeitplans können konventionelle Projektmanagementmittel, wie GANTT-Diagramme herangezogen werden. Aufgrund des Testplans ist schließlich für jede Testressource – Hardware, Software oder Personal – festzuhalten, wann sie zu welchem Zweck gebraucht wird.

3.8.15 Testrisiken und Risikomanagement

Jeder Systemtest ist mit Risiken behaftet. Es gibt vertragliche, technische, betriebliche und menschliche Risiken. Hier sind die einzelnen Risiken zu identifizieren und deren Auswirkung und Wahrscheinlichkeit abzuschätzen. Typische Testrisiken sind:

- Mängel in den Anforderungsdokumenten (Verfügbarkeit, Aktualität, Qualität)
- Fehlender Zugriff auf die Verfasser der Anforderungsdokumente
- Verzögerungen bei der Softwarelieferung
- Schwerwiegende Konstruktionsfehler in der Software
- Hardwareausfälle
- Performanzengpässe
- Instabile Grundsoftware
- Nichtfunktionierende Testwerkzeuge
- Personalprobleme

Für jedes Risiko, das auftreten kann, ist eine Gegenmaßnahme vorzusehen, zum Beispiel zusätzliche Hardwarekapazität beschaffen, den Softwarelieferanten Konventionalstrafen androhen, Personal austauschen, Überstunden anordnen und auf andere Werkzeuge ausweichen. Die Maßnahmen sind zu bewerten, in welchem Maße sie dazu dienen können, Risiken zu mindern. Anhand der Gegenmaßnahmen ist ein Netzplan zu erstellen.

3.8.16 Genehmigungen

Am das Ende des Testplans kommen die Namen und Titel aller Verantwortlichen, die diesen Testplan genehmigen müssen. Ihre Unterschrift bezeugt, dass sie mit dem Inhalt des Testplanes einverstanden sind. Aus Dokumentationszwecken sollte eine gedruckte Kopie des Testplans unterzeichnet und archiviert werden. Es empfiehlt sich, die elektronische Version des Testplans als XML-Dokument zu speichern. So kann es jederzeit wieder abgerufen und so mit den anderen Dokumenten des Projekts leichter verbunden werden.

Im Gegensatz zu vielen anderen Normen, die eher akademischer Natur sind, ist die IEEE Standard 829 sehr praxisbezogen. Sie wird nicht nur von der Firma ANECON, sondern auch von etlichen anderen Testfirmen, wie z.B. der SQS in Köln und der IFS Consulting in

Eschborn, praktiziert. Die IFS hat einen interessanten Erfahrungsbericht über die Umsetzung des Standards in dem GI Software Management Rundbrief veröffentlicht [SLAD05]. Anhang A dieses Buches enthält einen Testplan aus der ANECON-Testpraxis..