8 Testplanung

Für einen kosten-effizienten Test, vor allem bei größeren Projekten, ist die technische (was wird in welcher Reihenfolge mit welchen Methoden getestet?) und organisatorische (wer testet wann und mit welchen Hilfsmitteln) Planung des Tests enorm wichtig. Dieses Kapitel behandelt die technische und organisatorische Planung von Tests sowie die ökonomische Optimierung des Kosten/Nutzen-Verhältnisses durch iterative Planung.

8.1 Technische und organisatorische Planung

Die Testplanung erfolgt zu Beginn des Entwicklungsprojekts und wird, da sich ein Projekt an neu aufgetauchte Gegebenheiten anpassen muss, während der gesamten Laufzeit stetig aktualisiert. Die Testplanung kann in folgende Schritte weiter unterteilt werden[1]:

- organisatorische Planung (Ressourcenplanung)
- technische Planung (Teststrategie)

Da die organisatorische Planung stark von den Ergebnissen der technischen Planung abhängt, wird oft zuerst ein grober Rahmen festgelegt, meist in Form einer Aufwandsbeschränkung oder eines Endtermins, und innerhalb dieses Rahmens die technische Planung durchgeführt. Die Ergebnisse der technischen Planung fließen wiederum in die organisatorische Planung ein. Die Ergebnisse von organisatorischer und technischer Planung werden im Testplan festgehalten.

In vielen Fällen entspricht die Aufteilung von organisatorischer und technischer Planung auch der Verteilung der Aufgaben auf die unterschiedlichen Rollen im Testprozess: die organisatorische Planung obliegt dem Testmanager, die technische Planung dem Testingenieur (siehe dazu auch das Beispiel in Kapitel 6 und die Definition der Rollen in Kapitel 7).

8.1.1 Organisatorische Planung (Ressourcenplanung)

Bei der organisatorischen Planung steht die Ressourcenplanung im Mittelpunkt: Wer testet wann und wie, welcher Aufwand wird dafür veranschlagt, wie teuer kommt der Test? Ein wichtiger Punkt dabei ist, dass die kostenrelevanten Entscheidungspunkte im vorhinein identifiziert werden, sodass die benötigten Daten rechtzeitig zur Verfügung stehen. Dafür müssen die Ziele des Tests bekannt sein. Um den Test rechtzeitig an neue Gegebenheiten im zeitlichen oder budgetären Rahmen anpassen zu können, definiert die organisatorische Planung inkrementelle Testausbaustufen und Mechanismen zur Fortschrittsüberwachung.

> *„Die Ressourcenplanung hat zum Ziel, den Aufwand an Mitarbeitern samt der von ihnen voraussichtlich benötigten Arbeitszeit, verwendeten Werkzeuge und anderen Hilfsmittel, abzuschätzen und in einem sog. Testkonzept (engl. test plan) festzuhalten. Das Testkonzept*

[1] In [Spillner, 2002] wird die Testplanung in vier Schritte (Ressourcenplanung, Teststrategie, Testpriorisierung und Werkzeugunterstützung) unterteilt. Wir fassen die Testpriorisierung und die Planung der Werkzeugunterstützung jedoch als Teil der technischen Planung auf.

enthält auch einen Zeitplan für den Einsatz der Projektmitarbeiter. Weiterhin werden Aussagen zu Mitarbeiter-Schulungen, zur Organisation und zum Management der Testdurchführung gemacht und im Testkonzept schriftlich festgehalten." [Visek, 2003]

Die Ergebnisse der organisatorischen Planung umfassen also folgende Punkte:

- Festlegung der Testziele
- Kostenschätzung und Auswahl von inkrementellen Testausbaustufen
- Zeit- und Kostenplanung
- Schulungsplanung
- Festlegung der Organisation und des Managements der Testdurchführung
- Mechanismen zur Fortschrittsüberwachung

Im Rahmen der organisatorischen Testplanung stellen nur die Festlegung der Testziele und der inkrementellen Testausbaustufen neue Konzepte dar. Die anderen Aspekte verlaufen prinzipiell gleich wie im Projektmanagement (die organisatorische Testplanung ist ja oft auch Aufgabe des Projektmanagers) und werden daher hier nicht näher erläutert.

Festlegung der Testziele

Planung bedeutet die Vorbereitung der notwendigen Schritte um ein Ziel zu erreichen – das heißt also, dass zuerst klar gestellt werden muss, was denn überhaupt die Ziele sind. Im Idealfall, also wenn der Software-Entwickler sich an ein Vorgehensmodell wie das V-Modell (siehe z.B. [IESE, 2003]) hält, dann sind diese Ziele schon in schriftlicher Form in der Anforderungsanalyse oder im Pflichtenheft festgehalten. Ansonsten müssen die Qualitätsanforderungen an das System im Rahmen der technischen Planung ermittelt werden, zum Beispiel in Zusammenarbeit mit dem Kunden.

Kostenschätzung und Auswahl von Testausbaustufen

Vorgehensmodelle wie das V-Modell, bei denen das Testen und die Integration die letzten Stufen der Entwicklung darstellen, verursachen aber oft Probleme, wenn ein fix festgelegter Endtermin naht und die Budgets verbraucht sind. Terminüberschreitungen der vorhergehenden Phasen lassen sich nur noch auf Kosten des Testens ausgleichen, d.h. es stehen keine Ressourcen mehr für das Testen zur Verfügung.

Dies geschieht sehr oft und sollte in der Testplanung in Form von Testausbaustufen (Fallback-Varianten) berücksichtigt werden. Zur organisatorischen Planung gehört also auch die Forderung von inkrementellen Testvarianten mit unterschiedlichen Intensitäten von der technischen Planung. Somit kann ein bestimmtes Mindestmaß an Qualität im Fall von Terminproblemen oder Budgetkürzungen garantiert werden, sofern zumindest die Variante niedrigster Testintensität zur Gänze durchgeführt wird.

Solche inkrementellen Testausbaustufen können auch als Grundlage für die Angebotslegung verwendet werden, wenn Testen als Dienstleistung angeboten wird. Dem Auftraggeber können dann Angebote mit unterschiedlicher Testintensität unterbreitet werden, von denen er eines

auswählen kann. Zusätzlich kann er die gewählte Testausbaustufe nach eigenem Ermessen in Zusammenarbeit mit dem Dienstleistungsanbieter anpassen.

Die Aufgabe der organisatorischen Planung ist nun die Kostenschätzung der einzelnen Testausbaustufen, die Festlegung, welche davon überhaupt zur Anwendung kommen sollen, und die Zuweisung der verfügbaren Ressourcen.

Die Kostenschätzung ist im allgemeinen aber ein etwas schwierigerer Prozess, der auf mehreren nur vage eingeschränkten Parametern basiert. Wie der Planende zu möglichst soliden Aufwandsschätzungen für die Tests kommt, wird an späterer Stelle eingehender beschrieben.

8.1.2 Technische Planung (Teststrategie)

Die technische Planung befasst sich mit der Frage, was mit welchen Methoden, d.h. wie intensiv getestet wird. Dazu wird die Software in Komponenten und Qualitätsmerkmale zerlegt, und Testausbaustufen je Qualitätsmerkmal werden festgelegt. Ein wichtiger Teil der Teststrategie ist die Vergabe der Testprioritäten, d.h. die Festlegung, in welcher Reihenfolge die Tests durchgeführt werden sollen, damit auch bei Zeitmangel ein Test der kritischen Systemteile gewährleistet ist. Dazu gehört auch die Festlegung inkrementeller Testausbaustufen und die sorgfältige Planung des Ablaufs der Teststufen, insbesondere des Integrationstests. Ein weiterer Aspekt ist die Planung des Einsatzes von Testwerkzeugen. Hier wird geprüft, welche bereits vorhandenen Testwerkzeuge eingesetzt werden können und ob zusätzliche angeschafft werden sollen.

> *„Ziel der Teststrategie ist es, so früh wie möglich die wichtigsten Fehler zu den geringsten Kosten zu finden.“ [Pol et al, 2000]*

> *„Die Teststrategie legt fest, welche Teile des Systems mit welcher Intensität getestet werden müssen. Eine Teststrategie ist notwendig, da ein vollständiger Test, d.h. ein Test, der alle Teile des Systems mit allen möglichen Eingabewerten unter allen Vorbedingungen überprüft, in der Praxis nicht durchführbar ist. Deswegen muss in der Testplanung anhand einer Risikoabschätzung festgelegt werden, wie kritisch das Auftreten eines Fehlers in einem Systemteil einzuschätzen ist (z.B. nur finanzieller Verlust oder Gefahr für Menschenleben) und wie intensiv (siehe auch z.B. Überdeckungsmaße), unter Berücksichtigung der verfügbaren Ressourcen und des Budgets, ein Systemteil getestet werden muss oder kann.“ [Visek, 2003]*

Wichtige Aspekte der technischen Planung sind also:

- Zerlegung der Software in Komponenten und Qualitätsmerkmale
- Festlegung von Erfüllungsgraden je Qualitätsmerkmal und Systemkomponente
- Risikoeinschätzung
- Testpriorisierung (Festlegung der Teststrategie)
- Festlegung von inkrementellen Testausbaustufen
- Planung der Teststufen, insbesondere der Integration (Integrationsplan)

- Planung des Einsatzes von Testwerkzeugen

Die Bestimmung der Teststrategie wird weitgehend durch die Einschätzung der Relevanz von Systemkomponenten und Qualitätsmerkmalen durch den Kunden beeinflusst. Es ist daher in diesem Teil der Planung besonders wichtig, ein gutes Verhältnis zum Kunden zu haben.

Zerlegung der Software in Komponenten und Qualitätsmerkmale

Der erste Schritt zur Vorbereitung der Strategiefindung ist die Bestimmung der Qualitätskriterien, die für das Gesamtsystem relevant sind. Dabei ist es aus der Sicht des Anwenders, des Entwicklers und des Operators zu betrachten um *alle* relevanten Kriterien zu finden. Hierbei sind standardisierte Zerlegungen der Qualität aus Qualitätsmodellen wie z.B. ISO 9126 sehr hilfreich.

Der nächste vorbereitende Schritt zur Festlegung der Teststrategie besteht aus der Bestimmung aller Teilsysteme des Systems, z.B. basierend auf dem System-Entwurf. Das System wird in Untersysteme unterteilt, weil nicht für alle Teilsystem die gleichen Qualitätsansprüche zu gelten brauchen und die verschiedenen Teilsysteme andere Risiken bedingen können. Die Einteilung sollte im Prinzip die gleiche sein wie in der Entwurfsdokumentation. Falls sie davon abweichen sollte, so muss dies deutlich begründet werden.

Wenn eine Migration eines Datenbestandes nötig ist, ist diese als eigenes Teilsystem zu behandeln, da ja meist auch hier Software-Komponenten erzeugt werden müssen.

Zusätzlich wird häufig das Teilsystem „Gesamtsystem“ unterschieden, um angeben zu können, dass manche Qualitätsmerkmale erst anhand eines integralen Tests effektiv beurteilt werden können.

Erfüllungsgrade für Qualitätsmerkmale

Nachdem jetzt die Qualitätsmerkmale bekannt sind, welche für das Gesamtsystem Relevanz haben, ist noch zu definieren, welche Grade der Erfüllung dieser Qualitätsmerkmale es gibt, d.h. ein (bestenfalls quantitatives) Maß für die Qualität (eine Einführung in Maßzahlen bzw. Metriken ist z.B. unter Holzmann zu finden [Holzmann, 2002]). Dies hilft uns später bei der Beurteilung der Fähigkeit von Testmethoden, bestimmte Qualitätsmerkmale zu testen. Ein ideales Erfüllungsmaß ist ein Zahlenwert, z.B. ein wert zwischen 0 % und 100 %.

Im IEEE- Standard 1061 ist der Begriff Softwarequalitätsmetrik folgendermaßen definiert:

> *"Eine Softwarequalitätsmetrik ist eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet. Dieser berechnete Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit."*

Bei funktionalen Tests ist der Erfüllungsgrad des Merkmals in gewissem Maß durch die Testüberdeckung gegeben (siehe Kapitel 9). Bei vielen nicht-funktionalen Qualitätskriterien bestimmen schon die Maßzahlen, welche Testmethoden überhaupt angewendet werden können. Hier ist es am leichtesten jeweils diskrete, verbal beschriebene Stufen der Qualität festzulegen (siehe Kapitel 10).

Testmethoden für ein gegebenes Qualitätsmerkmal variieren in ihrer Eignung, einen bestimmten Erfüllungsgrad nachzuweisen. Je niedriger die Testintensität ist, desto niedriger wird auch der Erfüllungsgrad sein, der nach dem Test garantiert werden kann. Je nachdem, welcher

Erfüllungsgrad eines Qualitätsmerkmals gefordert ist, werden also unterschiedliche Testmethoden bzw. Intensitätsstufen davon Anwendung finden.

Risikoeinschätzung

Wenn ein System den Qualitätsanforderungen nicht oder nur unzureichend entspricht, entsteht dadurch ein Schaden für den Entwickler, da beispielsweise hohe Fehlerbehebungskosten anfallen oder Kunden unzufrieden sind. Dies stellt daher ein Risiko für den Entwickler dar.

Das Risiko lässt sich quantifizieren als Risiko = Fehlschlagwahrscheinlichkeit x Schaden. Die Fehlschlagwahrscheinlichkeit kann weiter zerlegt werden in Einsatzfrequenz und Fehlerwahrscheinlichkeit. Allerdings sind sowohl die Fehlerwahrscheinlichkeit als auch der Schaden nicht genau bekannt, können also nur geschätzt werden.

Die Fehlerwahrscheinlichkeit ist umso höher, je mehr der folgenden Faktoren im System und im Entwicklungsprozess vorhanden sind [Pol et al, 2000]:

- Komplexe Funktionen
- Gänzlich neue Funktionen
- (Insbesondere vielfach) angepasste Funktionen
- Funktionen, bei denen bestimmte Tools oder Techniken zum ersten Mal angewendet wurden
- Funktionen, deren Entwicklung zwischenzeitlich einem anderen übertragen wurden
- Funktionen, die unter extremem Zeitdruck entwickelt wurden
- Funktionen, die überdurchschnittlich optimiert werden mussten (z.B. zur Beschleunigung)
- Funktionen, in denen bereits zu einem früheren Zeitpunkte viele Fehler gefunden wurden
- Funktionen mit vielen Schnittstellen
- Unerfahrene Entwickler
- Unzureichende Einbeziehung oder Beteiligung der Endbenutzer
- Unzureichende Qualitätssicherung während der Entwicklung
- Neue Entwicklungswerkzeuge und -umgebungen
- Große Entwicklungsteams
- Fehlende Kommunikation im Entwicklungsteam (durch geographische oder persönliche Ursachen)

Um den Schaden möglichst gering zu halten, ist es wichtig, Fehler mit großem Schadenpotential so früh wie möglich zu finden. In jedem Fall muss der Testmanager aber darauf achten, dass die zur Verhinderung eines Risikos aufgewendeten Mittel nicht größer sind als der aus dem Risiko möglicherweise resultierende Schaden, vor allem in Anbetracht der Tatsache, dass Tests nur eine bestimmte Wahrscheinlichkeit haben, Fehler aufzudecken. Ein bestimmtes Risiko ist also selbst bei hohen Testaufwendungen immer noch vorhanden.

Eine Teststrategie zielt daher auf das Finden eines optimalen Gleichgewichts zwischen dem zu leistenden Testaufwand und der gewünschten Aufdeckung der Risiken ab. Systemkomponenten und Qualitätsmerkmale, deren Fehlschlag ein höheres Risiko in sich birgt, müssen intensiver getestet werden als solche mit geringem Risiko. Im Extremfall, wenn das Risiko nicht existiert, wird auch nicht darauf hin getestet.

Die in den folgenden Schritten durchzuführende Priorisierung der Tests nimmt implizit Rücksicht auf die vorhandenen Risiken je Systemkomponente und Qualitätsmerkmal. Dies reicht in den meisten Fällen der Softwareentwicklung ist dies auch ausreichend. Sind allerdings die zu erwartenden Risiken sehr hoch, empfiehlt sich auch eine quantitative Abschätzung des Risikos (siehe dazu auch Kapitel 15)

Testpriorisierung (Festlegung der Teststrategie)

> *„Das Ziel der Teststrategie ist es, so früh wie möglich die wichtigsten Fehler zu den geringsten Kosten zu finden [Pol et al, 2000].“*

Die wichtigsten Fehler sind üblicherweise in den wichtigsten Komponenten des Systems zu finden, und betreffen die wichtigsten Qualitätsmerkmale. Diese sind also als erstes zu testen. Daher wird bei der Testpriorisierung die Wichtigkeit der Teilsysteme sowie der Qualitätsmerkmale abgeschätzt. Die besten Ergebnisse werden dabei erzielt, wenn dies in Zusammenarbeit mit dem Kunden bzw. dem Endbenutzer geschieht, z.B. im Rahmen eines Workshops.

Der erste Teilschritt ist die Bestimmung der Bedeutung aller Teilsysteme. Um konkrete und im Sinne einer Priorisierung verwertbare Ergebnisse zu erzwingen (eine Aussage wie „jedes Teilsystem ist wichtig“ reicht nicht, um eine eindeutige Reihenfolge festzulegen), ist es ratsam, die Bedeutung der Teilsysteme relativ zueinander zu bestimmen. Dies geschieht am besten durch Vergabe von Werten zwischen 0 und 100 für jedes Teilsystem, wobei höhere Werte höhere Bedeutung bezeichnen. Die Summe aller zugeordneten Werte soll 100 ergeben. Es ist auch ratsam, eine Mindestgröße, beispielsweise 5, für die Relevanz festzulegen.

Dieselbe Abschätzung der relativen Bedeutung wird nun auch für die Qualitätsmerkmale durchgeführt.

Schließlich muss noch genauer bestimmt werden, welche Bedeutung die Kombination von Qualitätsmerkmal und Teilsystem hat. Qualitätsmerkmale können für manche Teilsysteme überaus wichtig sein und für andere überhaupt nicht. Diese Detaillierung kann ganz einfach durch Verwendung einer Matrix durchgeführt werden. In einer Matrix wird die Bedeutung jedes Qualitätsmerkmals je Teilsystem durch Zuweisung einer Wichtigkeitsstufe (z.B. „A“’ bis „F“) im Schnittpunkt zwischen Teilsystem und Merkmal festgesetzt. Die relative Bedeutung aus den vorherigen Schritten wird in der letzten Spalte und Zeile angeführt.

	Teilsystem 1	**Teilsystem 2**	**Teilsystem 3**	**Bedeutung**
Sicherheit	C	C	B	5
Einsetzbarkeit	C	B	C	5
Kontinuität	C	B	C	5
Funktionalität	B	A	A	25

Benutzerfreundlichkeit	A	A	A	30
Leistung	B	B	C	10
Integrierbarkeit	B	B	C	10
Sparsamkeit	B	C	B	10
Bedeutung	35	45	20	100

Tabelle 8.1: Matrix Qualitätsmerkmale und Komponenten von Software

Aus dieser Priorisierungsmatrix kann jetzt direkt eine Gewichtung jeder Kombination Qualitätsmerkmal/Teilsystem abgeleitet werden. Somit haben wir schon eine Reihenfolge für die Tests festgelegt, die garantiert, dass das Wichtigste zuerst getestet wird. Die Gewichtung lässt sich aber auch als eine Art Aufschlüsselung für die Verteilung der Testintensitäten verwenden.

Es fehlt also nur noch eine Zuweisung einer Testmethode zu jeder Kombination Qualitätsmerkmal/Teilsystem, welche diese Testintensitäten optimal erreicht. Hierzu muss bekannt sein bzw. geschätzt werden, welchen Grad der Erfüllung des Qualitätsmerkmals eine Testmethode gewährleisten kann, und was der damit verbundenen Aufwand ist. Es ist natürlich hilfreich, wenn ein großer Pool von Testmethoden zur Auswahl bereitsteht, d.h. wenn das Testteam viele Methoden kennt und auch anzuwenden weiß. Zuerst solle daher eine Liste der überhaupt in Frage kommenden Testmethoden und der damit erreichbaren Erfüllungsgrade erstellt werden.

Aus dieser Sammlung von zur Verfügung stehenden kann nun die je Kombination Qualitätsmerkmal/Teilsystem am besten geeignete Methode ausgesucht werden. Jedem Teilsystem und Qualitätskriterium wird je nach relativer Bedeutung eine Testmethode zugeordnet. Teure Methoden mit guter Überdeckung und Testintensität sollen nur für die wichtigen Merkmale und Teilsysteme angewendet werden. Als Rahmenbedingung ist hier meist die Beschränkung des Budgets und der Ressourcen zu berücksichtigen. Da manche Testmethoden nur mit Werkzeugunterstützung durchgeführt werden können, dürfen die damit unter Umständen verbundenen zusätzlichen Kosten (Einkauf des Werkzeugs, Einschulung der Mitarbeiter, etc.) nicht vergessen werden. Auf die Optimierung des Kosten/Nutzen-Verhältnisses wird später noch eingegangen.

Festlegung von inkrementellen Testausbaustufen

Um weiter zu gewährleisten, dass keine wichtigen Tests aufgrund von Zeit- oder Budget-Problemen unter den Tisch fallen, ist eine Definition von inkrementellen Testausbaustufen notwendig. Darunter ist eine Menge T von Tests zu verstehen, deren Teilmengen (die Testausbaustufen) jeweils wieder Teilmenge der nächst größeren Teilmenge (Testausbaustufe) sind: $T_1 \subset T_2 \subset ... \subset T_i \subset T$

Das heißt nichts anderes, als dass nach Erreichung einer Testausbaustufe (T_i) die nächste Testausbaustufe (T_{i+1}) durch Ausführen von zusätzlichen Tests ($T_{i+1} \backslash T_i$) erreicht werden kann. Wichtig ist die resultierende Eigenschaft $T_i \backslash T_{i+1} = \varnothing$. Diese besagt, dass beim Wechsel auf eine höhere Ausbaustufe keine Testfälle, die in der niedrigeren Ausbaustufe durchgeführt wurden, obsolet werden.

Somit können wir den Test bei der niedrigsten Ausbaustufe beginnen und uns Schritt für Schritt nach oben arbeiten. Falls Budget- oder Zeit-Probleme auftreten, kann trotz Abbruch des Tests jederzeit eine fundierte Aussage über die Qualität des Systems getroffen werden. Diese Aussage wird umso genauer, je höher die erreichte Testausbaustufe ist.

Eine einfache Möglichkeit der Festlegung von Testausbaustufen besteht darin, die Kombinationen von Qualitätsmerkmale und Systemkomponenten nacheinander zu testen. Die einfachste Form davon ist wiederum, nur die Qualitätsmerkmale oder die Komponenten zu berücksichtigen.

Beispiel. Bei der Definition der Tests für den Produktverkauf via Internet werden je Testausbaustufe jeweils alle Komponenten auf ein Qualitätsmerkmal getestet. Zusätzlich wird die Funktionalität in Basisfunktionen und Zusatzfunktionen unterteilt. Die sich ergebenden Testausbaustufen sind:

1. Basisfunktionen
2. Zusatzfunktionen
3. Performance
4. Usability

Dabei wird allerdings nicht auf das Prinzip „Breite vor Tiefe“ eingegangen: der Überblick über die Qualität des Systems ist umfassender, wenn zuerst alles ein bisschen getestet, und dann erst in die Tiefe gegangen wird. Ansonsten können immer noch Qualitätskriterien oder Systemkomponenten ganz unter den Tisch fallen.

Besser ist also eine Definition von Testausbaustufen, die dem Konzept „Breite vor Tiefe“ gehorcht. Dies wird durch Testmethoden ermöglicht, die in ihrer Intensität in der Art variiert werden können, dass die intensivere Variante die weniger intensive einschließt.

Planung der Teststufen

Bis jetzt haben wir den Test geplant, ohne Rücksicht darauf zu nehmen, wie weit die Entwicklung der Komponenten und deren Integration bereits abgeschlossen ist. Allerdings besteht der Test aus einer Abfolge von Teststufen, die beim Modultest beginnt und meist beim Abnahmetest endet. Es gibt nun zwei Möglichkeiten, die Teststrategie auf diese Teststufen umzusetzen:

1. Die Qualitätsmerkmale werden einer (Kombination von) Teststufe(n) zugeordnet.
2. Eine Teststrategie wird für jede Testausbaustufe definiert.

In der ersten Variante werden die Tests der unterschiedlichen Qualitätsmerkmale auf die Teststufen verteilt. Dabei werden sie in der frühest möglichen (Kombination von) Teststufe(n) angesiedelt. Dies geschieht unter Berücksichtigung der Tatsache, dass manche Testmethoden nur in bestimmten Teststufen anwendbar sind, und mit dem Ziel, den Gesamtaufwand so effizient wie möglich zu verteilen.

Die zweite Variante kann als mögliche Erweiterung der ersten angesehen werden; die erste Variante dient dann als übergeordneter Testplan. Variante 2 ist etwas aufwendiger in der Planung und testintensiver als die erste Variante.

In beiden Fällen kann somit ein relativ detailliertes Vorgehensmodell für die Teststufen geplant werden. Besonderes Augenmerk sollte allerdings auf den Integrationstest gelegt werden, bei dem es erfahrungsgemäß zu den meisten Problemen kommt. Es ist ratsam, die Vorgehensweise im Integrationstest gemeinsam mit der Teststrategie auszuarbeiten und im Testplan festzuhalten. Die Vorgehensweise im Integrationstest beschreibt die Reihenfolge der Modulintegration und die dabei verwendeten Hilfsmittel (siehe auch Kapitel 7).

Planung des Einsatzes von Testwerkzeugen

In diesem Schritt muss erhoben werden, welche Testwerkzeuge bereits vorhanden und im Einsatz sind. Sodann ist zu prüfen, welche Prozesse durch diese Werkzeuge unterstützt werden können, und welche Prozesse durch eventuell neu anzuschaffende Werkzeuge (dies ist streng genommen Teil der Prozessverbesserung, siehe dazu auch Kapitel 12). Hier lohnt sich ein genauerer Blick auf die geplanten Testmethoden, denn manche davon werden durch Werkzeugeinsatz effizienter oder gar erst möglich.

Typische Einsatzgebiete für Testwerkzeuge sind die Fehlerverfolgung, die Testdurchführung oder die Testauswertung. Im Testplan sollte genau beschrieben werden, wie diese Werkzeuge zur Anwendung gelangen. Dazu gehört die Beantwortung von Fragen wie: Für welche Prozesse sind sie zu verwenden? Wer ist für die Entwicklung und Wartung zuständig? Wo sind sie zu finden bzw. wer übernimmt die Installation? Welche Standards und Konventionen sind zu beachten?

Ein Überblick über die verfügbaren Werkzeugtypen sowie eine mögliche Vorgehensweise zur Auswahl und Anschaffung von Testwerkzeugen wird in Kapitel 11 geboten.

8.2 Kosten/Nutzen-Optimierung der Zuordnung von Ressourcen und Testmethoden

Um die Ressourcen (meist Personenstunden) optimieren zu können, müssen erst einmal die Aufwände geschätzt werden. Es empfiehlt sich, verschiedene Testausbaustufen zu schätzen, die einerseits im Falle von eingetretenen Risiken wirksam werden können, und andererseits bei Angebotslegung für Testen als Dienstleistung als Intensitätsvarianten zur Auswahl stehen können.

Ein heiklerer Punkt ist die Schätzung des Nutzens: was bekommt der Kunde beim Test in einer bestimmten Ausbaustufe für sein Geld? Dafür muss eine Qualitätsbeurteilung des Testobjekts, z.B. über die zu erwartende verbleibende Fehlerdichte gemacht werden.

In diesem Kapitel wollen wir näher betrachten, wie schon bei der Definition der Teststrategie das Verhältnis zwischen Kosten und Nutzen optimiert werden kann. Hierzu fließen bei der Zuordnung der Testmethoden Kosten-Parameter mit ein, die in den meisten Fällen nur geschätzt werden können. Die wichtigsten dieser geschätzten Parameter sind:

- Größe der Teilsysteme
- Aufwand je Testfall
- Anzahl Testfälle je Testmethode und Größeneinheit
- Nutzen

8.2.1 Größe der Teilsysteme

Der wichtigste Parameter für jegliche Art der Schätzung ist die Größe der Teilsysteme. Je größer eine Komponente des zu testenden Systems ist, desto umfangreicher werden sich natürlich deren Tests gestalten. Die Frage ist nun die: wie und in welchen Einheiten soll die Größe gemessen werden?

Die Problematik ist ähnlich der bei der Schätzung des Aufwands für das gesamte Projekt. Daher können im allgemeinen die Schätzungen bei Projektbeginn, sofern sie existieren, hier wieder verwendet werden. Allerdings geschieht die Schätzung der Komponentengröße im Rahmen der Testplanung meist zu einem Zeitpunkt, zu dem schon fundiertere Aussagen getroffen werden können als zu Beginn der Entwicklung. Der Testmanager wird daher meist danach trachten, die Schätzung der Komponentengröße auf eine möglichst aktuelle Basis zu stellen.

*Lines of Code (LOC)***.** Ein sehr häufig verwendetes Maß für Produktgrößen sind die Lines of Code (LOC). Dabei wird ganz einfach die Anzahl der Quellcode-Zeilen eines Programms als Maß für dessen Größe herangezogen. Leerzeilen und Kommentarzeilen werden im allgemeinen mitgerechnet, können aber auch abgezogen werden.

*Kilo Delivered Source Instructions (KDSI)***.** Oft auch NLOC (Netto Lines of Code) genannt. Dabei handelt es sich um eine Schätzung der Anzahl der im Programm enthaltenen Befehle. Kommentare und Leerzeilen werden nicht gezählt.

*Zyklomatische Komplexität (nach McCabe)***.** Die obigen beiden Größenmaße berücksichtigen nicht die Komplexität des Programms. Die zyklomatische Komplexität tut das, indem sie die Anzahl der Bedingungen in die Berechnung des Größenmaßes mit einbezieht. Die Formel für die zyklomatische Komplexität ist V(G) = e – n + 2p. e ist die Anzahl der Kanten des Kontrollflussgraphen G, n ist die Anzahl der Knoten, und p ist die Anzahl der Komponenten. Je größer V(G) desto komplexer und schwieriger zu Testen ist ein Programm bzw. ein Teilsystem.

*Halstead's Software-Maße***.** Halstead sah Programmieren als einen nicht festgelegten Prozess des Selektierens von Operatoren und Operanden aus einer vorher festgelegten Liste. Vorausgesetzt die Wahrscheinlichkeit der Auswahl eines bestimmten Operators ist für alle Operatoren gleich und ein binärer Entscheidungsbaum repräsentiert eindeutig die mentale Auswahl eines Menschen so ergibt sich das Volumen (V) einer Funktion zu V = (N1 + N2) x log2 (n1 + n2) mit

n1: Anzahl der verschiedenen Operatoren einer Funktion

n2: Anzahl der verschiedenen Operanden einer Funktion

N1: Gesamtanzahl der Operatoren einer Funktion

N2: Gesamtanzahl der verschiedenen Operanden einer Funktion

*Function Points***.** Dies (genauer: die unbewerteten Function Points) sind ein Ergebnis der Function-Point Methode zur Aufwandsschätzung. Dabei werden Benutzeranforderungen in Kategorien zerlegt und in ihrer Komplexität klassifiziert. Bei Auswertung dieser Klassifizierung nach der Function-Point Methode ergibt sich ein Punktewert, der ein objektives Maß für die Produktgröße darstellt.

Obige Beispiele sind klassische Maße für die Größe von (prozeduralen) Systemen. Im Fall von objektorientierten Systemen können auch andere Maße Anwendung finden, die ebenfalls nicht nur die Größe, sondern auch die Komplexität widerspiegeln. Typische reine Größenmaße sind die Anzahl der Klassen, Methoden, Attribute etc. Kombinationen sind auch möglich, beispielsweise Methoden pro Klasse. Die *Kopplung* bezeichnet die Interaktion zwischen Objekten, die *Bindung* den Grad an Beziehungen innerhalb des Objekts.

8.2.2 Durchschnittlicher Aufwand je Testfall

Ein weiterer wichtiger Parameter für die Optimierung der Ressourcen-Zuordnung ist der durchschnittliche Aufwand je Testfall. Darunter ist der durchschnittliche Aufwand für Testfalldefinition, Testdurchführung, Testprotokollierung sowie für eine eventuelle Fehlererfassung und Nachtests zu verstehen. Dieser Parameter wird in Stunden oder Minuten angegeben. Je größer der durchschnittliche Aufwand je Testfall ist, desto größer ist natürlich auch der zu erwartende Gesamtaufwand für die Tests.

Mehrere Faktoren können Einfluss auf den durchschnittlichen Aufwand je Testfall haben:

- Erfahrung der Tester
- Einsatz von Werkzeugen
- Komplexität der Testfälle
- Komplexität des Programms
- Qualität des Programms

Aufgrund der großen Anzahl von schwer einschätzbaren Einflussfaktoren auf diesen Parameter ist er sehr schwer zu schätzen. Die ideale Schätzmethode wäre eine Protokollierung des Testaufwandes je Testfall in allen Projekten, sodass in späteren Projekten mit ähnlichem Aufbau leicht der durchschnittliche Aufwand berechnet und als Schätzung für diesen Parameter verwendet werden kann. Es ist allerdings leicht nachvollziehbar, dass dies zu einem enormen Overhead im Test führen würde, der nicht durch die (nur möglicherweise!) bessere Qualität der Schätzung wettgemacht werden kann.

Üblicherweise erfolgt daher die Schätzung dieses Parameters durch einen Experten, der mehr oder weniger „aus dem Bauch heraus“ den Aufwand schätzt.

8.2.3 Anzahl Testfälle je Testmethode und Größeneinheit

Jede Testmethode führt zu Testfällen irgendeiner Art. Die Anzahl der dabei generierten Testfälle hängt von der Testmethode sowie von der Größe und Komplexität des getesteten (Teil-)Systems ab. Viele Testmethoden können selbst wieder in ihrer Intensität variiert werden.

Nehmen wir zum Beispiel eine funktionale Testmethode, die je Verzweigung im Programm einen Testfall generiert. Je mehr Verzweigungen dieses Programm hat, desto mehr Testfälle ergeben sich. Wird diese Testmethode in ihrer Intensität gesteigert, z.B. in der Art, dass sie für jede Bedingung in einer Verzweigung einen Testfall generiert, so werden es auch mehr Testfälle.

Die Anzahl der Testfälle, die sich aus einer Methode je Größeneinheit (beispielsweise je Function Point) ergeben, ist ein Maß, das direkt proportional zum zu erwartenden Testaufwand ist.

8.2.4 Nutzen

Um beurteilen zu können, ob eine Testmethode bzw. eine Auswahl von Testfällen „ihr Geld auch wert ist“, d.h. möglichst viele Fehler zu möglichst niederen Kosten aufdeckt, müssen wir den Nutzen abschätzen. Der Nutzen kann auf eine der folgenden Arten charakterisiert werden:

- Wahrscheinlichkeit, einen Fehler zu finden
- Prozentueller Anteil gefundener Fehler
- Anzahl der Restfehler

Alle diese Charakterisierungen wollen mit geringen Unterschieden dasselbe aussagen: Um wie viel kann die Testmethode das Risiko, dass ein Fehler im fertigen Produkt auftritt, verringern? Welchen möglichen Schaden kann ein Test nach dieser Methode abwenden?

Im ersten Ansatz wird versucht, die Wahrscheinlichkeit abzuschätzen, dass ein durch eine gegebene Testmethode definierter Testfall einen Fehler aufdeckt. Der Wert von 100% wäre für eine ideale, umfassende Testmethode, die alle Fehler finden kann, zu vergeben.

Der Prozentuelle Anteil gefundener Fehler bezieht einen weiteren Faktor mit ein: wie viele Fehler wurden nicht gefunden? Wenn x % aller Fehler gefunden wurden, sind immer noch 100-x % der Fehler im System vorhanden.

Mit einer Schätzung, wie viele Fehler in absoluten Zahlen im System zu erwarten sind, kann somit die Anzahl der Restfehler aus dem prozentuellen Anteil gefundener Fehler berechnet werden. Dieser Wert kann dann direkt in einen möglicherweise auftretenden Schaden umgerechnet werden, wenn ein mittlerer Wert für den pro Fehler verursachten Schaden angenommen wird.

Eine Möglichkeit zur Schätzung der Fehlerzahl in einem System nach [Thaller, 2002] ist folgende Formel:

$F_r = C_1 + C_2\ (SCHG / KLOC) - C_3\ ISKL - C_4\ (DOCC / KLOC)$, *mit*

F_r: *Fehlerrate pro 1000 Programmzeilen (LOC)*

C_1: *Konstante 67,98*

C_2: *Konstante 0,46*

C_3: *Konstante 9,69*

C_4: *Konstante 0,08*

SCHG: Anzahl der Änderungen am Lastenheft während der Entwicklung

KLOC: Codeumfang in 1000 Programmzeilen (LOC)

ISKL: durchschnittliche Erfahrung des Entwicklungsteams mit einer Programmiersprache in Jahren

DOCC: Anzahl der geänderten oder neu erstellten Seiten in Designdokumenten

Die so ermittelten Werte stellen allerdings nur eine Schätzung dar: die tatsächlichen Werte sind möglicherweise um vieles höher oder niedriger!

Weitere mögliche Methoden zur Schätzung der Fehlerzahl in einem System sind beispielsweise in [VISEK,2003] beschrieben.

8.2.5 Zuordnung der Ressourcen und Testmethoden

Ausgehend von der Matrix der Qualitätsmerkmale je Teilsystem können wir nun die jeweils zur Wichtigkeit passenden und vom Aufwand her auch tragbaren Testmethoden zuordnen. In den vorigen Schritten haben wir die zur Schätzung des Aufwandes notwendigen Parameter abgeschätzt. Der Aufwand je Teilsystem, Qualitätsmerkmal und Testmethode ergibt sich nun aus der Formel:

Testaufwand = Anzahl der Testfälle/Größeneinheit x Größe des Teilsystems x Aufwand/Testfall

Die Wahl der Testmethoden sollte so erfolgen, dass sich der Testaufwand je Qualitätsmerkmal und Teilsystem an der durch die Teststrategie festgelegten Priorisierung orientiert. Natürlich gilt es auch, gegebene Einschränkungen im Gesamtaufwand zu berücksichtigen. Sind zwei Testmethoden vom Aufwand her gleich, so sollte diejenige, die den größeren Nutzen verspricht, gewählt werden. Der Gesamtaufwand ist dann die Summe des Aufwandes über alle Teilsysteme und Qualitätsmerkmale.

Wie eingangs bereits erwähnt, ist eine Festlegung von mehreren inkrementellen Testausbaustufen ratsam. Je Testausbaustufe kann sodann der Testaufwand sowie der mögliche Nutzen abgeschätzt werden.

Im Sinne des Risikomanagements kann somit auch eine Art „Notfallplan“ festgelegt werden, der beinhaltet, bei Eintritt welcher Risiken der Umstieg von einer höheren auf eine niedrigere Testausbaustufe stattzufinden hat, wie dieser Umstieg durchgeführt wird und welcher zusätzliche Aufwand damit verbunden ist.

Die Aufwandsschätzung sowie die gewählten Testmethoden (Ausbildungsstand der Tester!) bilden die Grundlage für die Zuteilung der Ressourcen. In der Literatur sind auch andere Verfahren zur Testaufwandsschätzung zu finden, z.B. die auf der Function-Point Methode basierende Testpunktanalyse in [Pol et al, 2000].

8.3 Ein iterativer Ansatz zur Testplanung

Mehrere Faktoren führen dazu, dass die Testplanung und speziell die verwendete Teststrategie in den wenigsten Fällen so durchgezogen wird, wie sie anfangs festgelegt war. Die zwei wichtigsten davon sind:

- Die Qualität der zu testenden Software ist nicht von Anfang an bekannt, wird aber während des Tests offensichtlich. Dies kann dazu führen, dass eine andere Teststrategie ökonomischer wird. Wenn Tests im Vergleich zur Anzahl der gefundenen Fehler zu teuer werden, sollten

weniger intensive Tests durchgeführt werden, wenn Teilsysteme wichtiger werden, sollten dafür die Tests intensiver gemacht werden, etc.

- In den wenigsten Fällen stimmen die im Vorhinein geschätzten Einflussgrößen, welche in die Testplanung mit eingegangen sind, mit den tatsächlichen überein. Bei diesen geschätzten Einflussgrößen handelt es sich z.B. um den durchschnittlichen Aufwand je Testfall, die durchschnittliche Anzahl Testfälle je Größeneinheit der Applikation und je Testmethode, die Qualität der Testfälle etc.

Daher sollten Tests in kurzen Iterationszyklen oder kurzen inkrementellen Stufen geplant werden. Iterationszyklen für Tests können auf vielerlei verschiedene Arten definiert werden. Die in der Praxis am häufigsten verwendete Art ist der Beginn eines neuen Testzyklus bei Freigabe einer neuen Version durch die Entwicklung. Bei einer solchen Einteilung ist der Test allerdings von der Geschwindigkeit der Entwicklung abhängig: sind die Entwicklungszyklen zu kurz, so können nicht alle geplanten Tests durchgeführt werden; sind sie zu lang, so entstehen Stehzeiten beim Test. In der Testphase des Entwicklungsprozess sollte aber der Test die Länge der Iterationszyklen vorgeben.

Besser ist also eine Einteilung in kurze Testzyklen, beispielsweise über die Dauer des Zyklus oder über die aufgewendeten Arbeitsstunden. Noch besser ist es, der Testmanager verwendet gleich die Ergebnisse der Teststrategie: ein Zyklus besteht aus dem Test einer Kombination aus Teilsystem und Qualitätsmerkmal. Sind die alle geplanten Tests einer Ausbaustufe beendet, so beginnt die nächste Ausbaustufe, sofern der geplante Aufwand noch nicht aufgebraucht ist. Der Test einer Kombination aus Qualitätsmerkmal und Teilsystem kann somit in mehreren, zeitlich voneinander getrennten Schritten bestehen, die jeweils einer anderen Testausbaustufe zugeordnet sind.

Ein Iterationszyklus besteht typischerweise aus folgenden Phasen (siehe dazu auch Kapitel 7):

- Planung der Tests im Iterationszyklus
- Vorbereitung und Durchführung der Tests
- Korrektur der gefundenen Fehler
- Überprüfung des Testfortschritts

Planung der Tests im Iterationszyklus. In der ersten Iteration bzw. in der ersten Stufe werden zunächst die Einflussgrößen in Intervallen (minimal bis maximal) geschätzt werden. In der nächsten Iterationsstufe können neue und aufgrund der in vorigen Stufen gesammelten Erfahrung genauere Werte für die Einflussgrößen bestimmt werden. Die neu ermittelten Ressourcen-Aufwände fließen zusammen mit den neuen Erkenntnissen über die Qualität des Produkts in die weitere Planung ein.

Die Teststrategie muss unter Umständen an den bisherigen Testfortschritt angepasst werden: finden Tests wenige Fehler, so kann die Intensität verringert werden; kommen Tests zu langsam voran, wird im Extremfall auf eine niedrigere Testausbaustufe umgestiegen.

Die Sammlung von Daten über die Einflussgrößen und deren genauere Bestimmung stellt einen ersten Schritt in Richtung Testprozessverbesserung dar. Dadurch wird in späteren Iterationsstufen oder Projekten die Auswahl der (Intensität der) Testmethoden sowie die Ressourcenzuteilung in methodischen Tests erleichtert.

Vorbereitung und Durchführung der Tests. Die geplanten Tests müssen zuerst vorbereitet werden. Typische Vorbereitungsschritte sind die Einrichtung von Testdatenbanken, Entwicklung und/oder Installation von Testwerkzeugen, Einrichtung einer Testversion des (Teil-)Systems etc. Sobald die Vorbereitung abgeschlossen ist, können die Tester die Tests durchführen, protokollieren und gegebenenfalls Fehler melden.

Korrektur der gefundenen Fehler. Die Entwickler können mit der Korrektur von Fehlern beginnen, sobald ein Fehler gefunden wurde. Wenn die Fehlerkorrektur parallel zu den Tests ablaufen soll, ist eine strenge Trennung von Test- und Entwicklungsumgebung notwendig. Manche Fehler können testverhindernd sein. Ihre Korrektur ist dann Voraussetzung für den Abschluss des Testzyklus. Ansonsten müssen Fehler nicht unbedingt im selben Zyklus korrigiert werden.

Die Korrektur von Fehlern, deren Korrekturaufwand im Vergleich zum möglichen Schaden zu hoch ist, kann nach hinten verschoben werden oder gar ganz entfallen. Die Priorisierung einer Fehlerkorrektur obliegt dem Test- oder Projektmanager, die natürlich die Kundenanforderungen berücksichtigen müssen, wobei wieder die Matrix mit der Relevanz der Qualitätsmerkmale je Teilsystem hilfreich ist.

Überprüfung des Testfortschritts. Die Überprüfung des Testforschritts geschieht laufend. Sind alle für den Testzyklus geplanten Tests abgeschlossen, erstellt der Testmanager Testberichte, deren Inhalte er mit den Testendekriterien vergleicht. Sind die Testendekriterien erfüllt, kann der Testmanager die Tests für beendet erklären, ansonsten beginnt ein neuer Testzyklus.

8.4 Zusammenfassung

8.5 Literaturreferenzen

[Balzert, 1998] Balzert, Helmut: *Lehrbuch der Software-Technik 1/2*, Spektrum Akademischer Verlag, Band 1 und 2, 1998/2000, ISBN: 3827403014.

[Henderson, 1995] Henderson-Sellers, Brian: "Object-Oriented Metrics: Measures of Complexity", Prentice Hall, ISBN 0132398729, 1995.

[Holzmann, 2002] Holzmann, Clemens: „Seminar Programmierstil: Metriken“, http://www.ssw.uni-linz.ac.at/Teaching/Lectures/Sem/2002/reports/Holzmann/.

[IESE, 2003] Fraunhofer Institut für Experimentelles Software-Engineering: „Der V-Modell-Guide“, http://www.iese.fhg.de/VModell/.

[Pol et al., 2000] Pol, Martin; Koomen, Tim ; Spillner, Andreas: “*Management und Optimierung des Testprozesses: ein praktischer Leitfaden für Testen von Software, mit TPI und TMap*”, dPunkt, 2000, ISBN 3-932588-65-7.

[Rätzmann, 2002] Rätzmann, Manfred: „Software-Testing - Rapid Application Testing, Softwaretest, Agiles Qualitätsmanagement“, Galileo, 2002, ISBN 3898422712.

[Schaefer, 1996] Schaefer, H.: „Surviving under time and budget pressure“, Proceedings Euro-STAR Conference, Amsterdam, 1996.

[Spillner, 2002] Spillner, Andreas; Linz, Tilo: „Basiswissen Software-Test“, dPunkt, 2002, ISBN 3898641783.

[Thaller, 2002] Thaller, Georg Erwin: *„Software-Test – Verifikation und Validation“,* Heinz Heise Verlag, 2002, ISBN 3882291982.

[VISEK, 2003] Bundesministerium für Bildung und Forschung (Deutschland): „*Virtuelles Software Engineering Kompetenz-Zentrum“*, http://www.visek.de.

[Zuse, 1998] Zuse, Horst: "A Framework of Software Measurement", Walter de Gruyter, 1998, ISBN 3110155877.

8.6 Übungen und Fragen