Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Änderung- und Versionsverfolgung		
Ursprung: Diplomarbeit24_6.rtf 24.06.1999 14:49		
Datum	Action	Bemerkung
07.07.1999	Erweitern	Kap. 3.5 .... 3.7 Stichpunkte *ver2
07.07.1999	Ändern	Trennen von 3.4 Testfallentwurfsmethoden in 3.4 Testfallentwurf 3.4.1 Testfallentwurfmethoden *ver2
07.07.1999	Ändern	*ver3 diverse lit
08.07.1999	Backup	08_7vers1
12.07.1999	Backup	12_7vers1
12.07.1999	Erweitern	Kap. 6.1 PerformanceTest im ITB diverse S. *12_7vers2
12.07.1999	Erweitern	Kap. 9.1 Evaluierung Testwerkzeuge *12_7vers3
17.07.1999	Backup	*17_7vers1.....ver3
18.07.1999	Backup	*18_7ver1
18.07.1999	Ändern	*18_7vers Kap. 3.4.8; 3.5; 3.6; 3.7; ausformuliert und ergänzt
19.07.1999	Backup	*19_7vers1
19.07.1999	Ändern	*19_7vers2 Rechtschreibkontrolle bis S.36
19.07.1999	Erweitern	*19_7vers3 Evaluierung Testwerkzeuge tabellenüberschrften
19.07.1999	Ändern	*19_7vers3
20.07.1999	Backup	*20_7vers1 RS bis 37
20.07.1999	Ändern	*20_7vers1 RS bis 41
21.07.1999	Backup	*21_7vers1
21.07.1999	Ändern	*21_7vers2 RS und div Aenderungen bis 48
21.07.1999	Ändern	*21_7vers3 RS und div Aenderungen bis 50
21.07.1999	Ändern	*21_7vers4 RS und div Aenderungen bis 55
22.07.1999	Backup/Ände	*22_7vers1 RS und Bilder 8.1-1; 8.0-1
22.07.1999	Ändern	*22_7vers2 bis Kap 8.3

1 Die Motivation zum Testen / Grundlagen des Software-Tests

In den unterschiedlichsten Anwendungsbereichen kommt heutzutage Software zum Einsatz. Da die Qualität der Software für den Kunden einen wachsenden Stellenwert erlangt, kommt speziell auf den Softwareentwicklungsprozeß eine verstärkte Orientierung auf Qualitätsmaßnahmen zu. Eine Schlüsselrolle spielt dabei das Testen, dass Probleme erkennbar macht, objektiviert und daher den Anfang für den notwendigen Kulturwandel darstellen kann.

Messungen, Testen und Datenanalysen stellen die wesentliche Grundlage zur Verbesserung der Vorgehensweise und Methoden dar.

Damit wird im Software-Engineering wiederentdeckt, was der messenden Naturwissenschaft schon lange selbstverständlich ist: Nicht Messungen liefern Hypothesen, sondern Hypothesen verlangen geeignete Maße und bestimmen die zu messenden Daten. Nur sorgfältig auf ein klares Ziel ausgerichtete Meßprogramme liefern relevante Daten, und schaffen damit eine solide Grundlage für eine konkurrenzfähige Teilnahme in Neuentwicklung, Wartung und Erneuerung von Software. [Taube]

1.1 Ziel und Bedeutung des Software-Tests

In dieser Arbeit werden die statischen Qualitätssicherungs(QS)-Maßnahmen wie Dokumenten-Reviews oder Code-Inspektionen unter dem Begriff "Prüfen" zusammengefaßt. Der Begriff "Testen" steht hingegen für alle dynamischen QS-Maßnahmen im Rahmen der Softwareentwicklung. Dem Testen kommt damit die Aufgabe zu, mittels Methoden zur Fehlererkennung, ein Nachweis der Übereinstimmung mit dem Sollobjekt zu führen.
Laut [SQS-QT] liegen Fehler zu 63% im Fachkonzept (Design) und zu 37% in der Realisierung (Programmierung). Daraus folgt die Notwendigkeit der Begleitung des Softwareprodukts durch das Testteam schon während der konzeptionellen Phase.

Die analytischen Methoden des Softwaretests unterteilen sich in statisches Prüfen (Walktrough; Codeinspektion) und in dynamisches Prüfen, dem Testen, bei dem die realen Einsatzbedingungen des Systems ausreichend berücksichtigt werden.

Ziel des Software-Tests ist es, durch Ausführung des Testobjekts auf einem Rechner mit ausgewählten Testdaten in einer definierten Testumgebung festzustellen, ob sich das Testobjekt in diesen Fällen so verhält, wie es eine definierte Testreferenz vorschreibt.
[Grimm]

2 Teststufen und Strategien

Es bestehen 2 Möglichkeiten Programme zu testen.

1. Beim Dokumententest (Codeinspection, Walktroughs) werden Code und Spezifikation von mehreren Leuten gelesen und gemeinsam ausgewertet. Dazu werden Testfälle direkt auf dem Papier unter Anwendung der Eingabedaten ausgeführt. Eine Fehlerprüfliste für Programminspektionen bezüglich Datenreferenz, Datenvereinbarung, Berechnungsfehler, Vergleich, Steuerfluß, Schnittstellenfehler, Ein-/Ausgabe-Fehler und sonstige Prüfungen befindet sich in [Myers].

2. Der Computertest ist der eigentliche Test des Programmes am Rechner nach allen Methoden der Testfallermittlung und –ausführung.

Im folgenden werden die Teststufen des Computertests differenziert und erläutert.

2.1 Entwicklertest

Der Entwicklertest umfaßt das Testen der Unterprogramme/Programme hinsichtlich der korrekten DV-technischen Realisierung sowie der formalen Aspekte der Benutzerschnittstelle. Dies entspricht einer Überprüfung auf Einhaltung der Programmier- und Dokumentationsrichtlinien. Der Entwicklertest ist die Voraussetzung der Übernahme der Programme von der Entwicklungsumgebung in die Testumgebung. Der Entwicklertest unterteilt sich in Modultests (Komponententest) und technische Integrationstests (Zusammenspiel der Komponenten / Verbundtest, Schnittstellentest).

Das erste und wichtigste Testtool des Entwicklers ist der verwendete Compiler selbst. Während des Compiliervorgangs prüft der Compiler vor allen Dingen, ob die ihm angebotene Information "richtig" ist, ob sie also formal den Regeln der Quellsprache genügt. Von besonderer Bedeutung ist dabei eine möglichst vollständige und leicht verständliche Analyse der etwaigen Fehler, damit eine einfache und schnelle Behebung derselben möglich wird.
Nach der Art der auftretenden Fehlerfälle unterscheidet man 3 Teilbereiche einer solchen Analyse:

- Die "lexikalische Analyse" stellt fest, ob die zu prüfende Information nur Symbole enthält, die dem ALPHABET der Quellsprache entstammen.
- Die "syntaktische Analyse" prüft, ob die formalen Regeln der GRAMMATIK (der "Syntax") erfüllt sind.
- Die "semantische Analyse" stellt fest, ob die BEDEUTUNG (die "Semantik") der gesamten Information in ihrem ZUSAMMENHANG ("Kontext") sinnvoll und korrekt ist.

[FHBiel]
Beim technischen Integrationstest (E/A-Test) beginnt zwangsläufig der Modultest, da ein integratives Testen von mehreren Modulen durchgeführt wird. Es wird das Modulverhalten bei Interaktion mit anderen Programmmodulen getestet. Dadurch entsteht der Effekt, daß Kombinationen von unterschiedlichen Programmteilen getestet werden.
Daraus entsteht die Grundfrage des Modultests.
Soll jedes Modul einzeln getestet werden und dann die Module zusammengesetzt werden oder soll man das nächste zu testende Modul mit dem schon getesteten Modulen kombinieren, bevor es getestet wird?
Der erste Fall wird nichtinkrementelles Testen genannt. Das Zweite wäre inkrementelles Testen.

Gesichtspunkt	**Nichtinkrementelles**	**Inkrementelles**
Arbeitsmenge	hoch- es sind Treiber und Stubs zu programmieren	niedriger- es sind entweder Treiber oder Stubs zu programmieren
Programmfehler zwischen Modulen	werden erst am Ende entdeckt	werden sofort entdeckt
Fehlerbehebung	schwer lokalisierbar	leicht lokalisierbar
Testgenauigkeit	niedriger, da nur Treiber und Stubs beteiligt	höher, da auch echte Module beteiligt
Maschinenzeit	geringer	höher
Paralleles Testen	kaum möglich	gut möglich

Tabelle 2.1-1: Vergleich inkrementelles und nichtinkrementelles Testen [Leitf]

Bottom-Up und Top-Down sind 2 Methoden des inkrementellen Testens.
Bei der Top-Down-Strategie untersucht man zuerst die Module der obersten Hierarchiestufe in einem Testobjekt (TO), beginnend beim Hauptmodul, und fügt anschließend einzelne Module der nächsttieferen Stufen hinzu, die von einem bereits getesteten aufgerufen werden. Für diejenigen Module der niedrigeren Stufen, die noch nicht in die Tests einbezogen sind, aber von hierarchisch höherliegenden Modulen benutzt werden, müssen Stubs (Platzhalter) implementiert werden, die die Aktionen des betroffenden Moduls simulieren und dessen Ausgaben generieren.

Nach der Bottom-Up-Strategie sind zunächst alle Programme auf der untersten Ebene zu testen, bevor die Programme der nächsthöheren Ebene sukzessive hinzugefügt werden.

Der Vorteil dieser beiden strukturierten Vorgehensweisen liegt darin, daß Abweichungen, die während der Testausführung auftreten, ihren Ursprung in der Regel in der zuletzt hinzugefügten Komponente haben. Eine andere Möglichkeit des Tests besteht darin, alle Module auf einmal zu testen (Big-Bang-Methode). Die Fehlerlokalisierung ist dementsprechend aufwendig.

Die Sandwich-Integration als Kombination der Vorteile des Bottom-Up und Top-Down-Tests wird in der Praxis häufig als Mischform eingesetzt. Dabei beginnen beide Vorgehen, von oben wie von unten, gleichzeitig. Die getesteten Module werden schichtenweise miteinander integriert und kommen in der „Mitte“ zusammen.

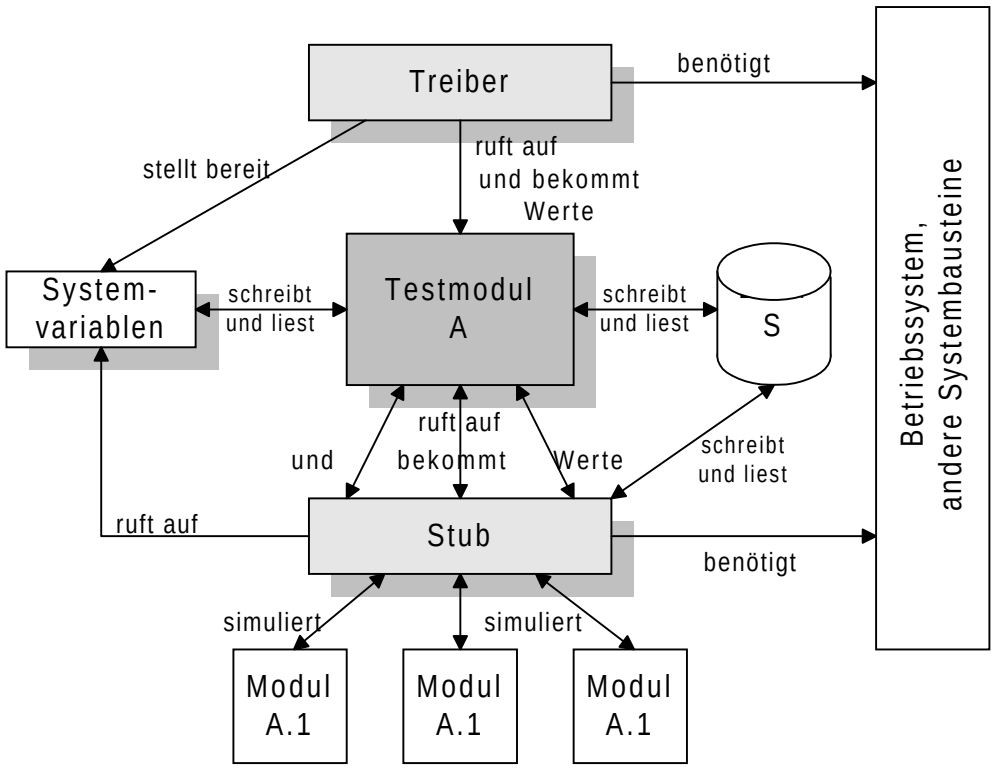


Abbildung 2.1-1: Treiber und Stubs [SQS-QT]

2.2 Funktionstest

Beim Funktionstest (Vorgangstest) wird die zu testende Anwendung im Hinblick auf die richtige Bearbeitung der Details der einzelnen Funktionen/Aufgaben ausgeführt. Bei der Betrachtung des TO (Testobjekt) von außen wird die Reaktion auf Testeingangsdaten nur an den Ausgängen beobachtet (Black-Box-Testen). Außer den Schnittstellen sind keine Informationen über die Struktur des TO vorhanden. Der Funktionstest kann das Fehlen von Funktionen bzw. Qualitätseigenschaften des TO erkennen. Das Vorhandensein unerwünschter Funktionen und Eigenschaften kann der Funktionstest nicht mit Sicherheit nachweisen.

2.3 Weitere Testarten

Beim Funktionskettentest (Vorgangskettentest, Anwendungstest) werden aus Benutzersicht Folgen von Funktionen (Aufgaben) des zu testenden DV-Systems überprüft. Im darin anschließenden fachlichen Integrationstest (Verbundtest) werden Folgen von Aufgaben / Funktionen unter besonderer Berücksichtigung der Schnittstellen mit bestehenden DV-Systemen berücksichtigt. Im Systemtest erfolgt der Test der Anwendung gegen die

Anforderungen Performance und Zuverlässigkeit mit Bezug auf die Verarbeitung von Massendaten.
Beim Installationskontrolltest erfolgt die Überführung in die Wirkumgebung (richtige Einbindung des neuen Anwendungssystems in eine Echtumgebung) sowie der Test der Installation der Anwendung in einer neuen Umgebung.

3 Testaktivitäten

Ein systematischer Test untergliedert sich in die Kernaktivitäten Testfallermittlung inklusive Sollergebnisbestimmung, Testdatengenerierung, Testdurchführung, Monitoring und Testauswertung sowie die den Test begleitenden Aktivitäten der Testorganisation und Testdokumentation.

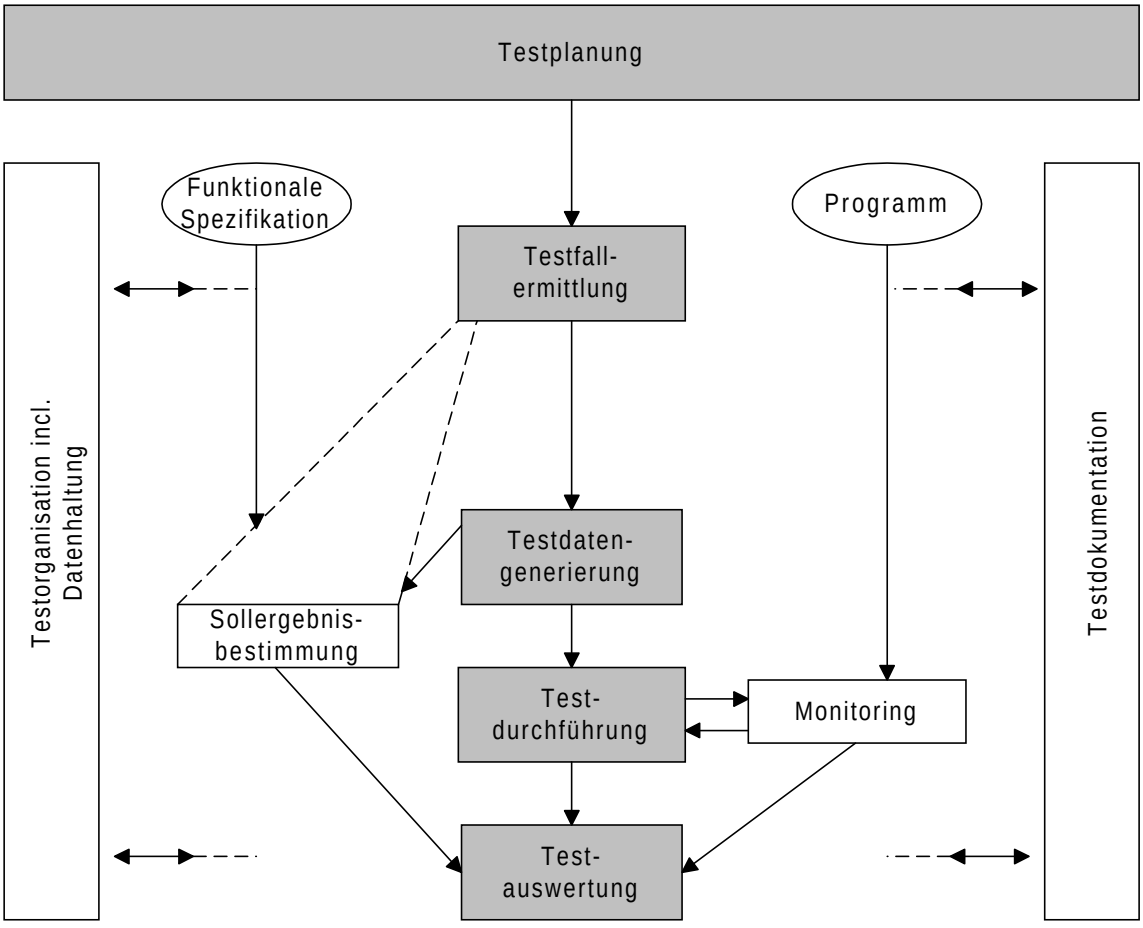


Abbildung 2.3-1: Die Testaktivitäten und ihr Zusammenwirken [SQS-QT]

1. Die erste jeweils auf ein TO bezogene Aktivität ist die Testfallermittlung. Ein Testfall legt eine bestimmte Eingabekonstellation fest, mit der das TO getestet werden soll und bestimmt ein erwartetes Ergebnis. Er umfaßt eine Menge von konkreten Eingabewerten

aus dem Eingabedatenraum des Testobjekts, die zur Ausführung derselben Programmfunktion, zum Durchlaufen desselben Programmzweiges oder zum Erreichen desselben Programmzustandes führen. Die Testmethoden werden nach der Vorgehensweise bei der Testfallermittlung unterschieden. Diese ist damit entscheidend für die Qualität des Tests, da hier Art und Umfang der Prüfung und damit die Güte des Tests festgelegt werden. Zu jedem Testfall und dem entsprechenden Testdatensatz sind mittels der Testreferenz, im Allgemeinen der Anforderungsspezifikation oder des Software-Entwurfs, Sollergebnisse für die Ausgangsgrößen des Testobjekts zu definieren. Lassen sich Sollverhalten oder Sollwerte nicht eindeutig angeben, so müssen die vom Testobjekt errechneten Ausgangswerte anhand anderer Referenzdaten oder Plausibilitätskriterien ausgewertet werden.

2. Im Zuge der Testdatengenerierung sind für die festgelegten Testfälle konkrete Testdaten zu definieren, indem aus den testfallspezifischen Mengen von Eingabewerten einzelne Werte als Repräsentanten ausgewählt werden (siehe Error Guessing, Grenzwertmethode etc.). Mit den Testdaten wird das Testobjekt während der Testdurchführung aufgerufen.
3. Bei der Testdurchführung ist das Testobjekt mit den ausgewählten Testdaten auszuführen, und die Testergebnisse, das heißt die vom TO erzeugten Ausgabewerte, sind in geeigneter Form, beispielsweise in entsprechenden Ausgabedateien, festzuhalten. Beim Test eines unvollständigen TO ist ein Testrahmen zu definieren, wobei bei fehlender Testobjektumgebung ein Treiber, bei fehlenden Teilfunktionen Stubs bereitzustellen sind. Über diesen Testrahmen wird das unvollständige Testobjekt, gegebenenfalls interaktiv, mit den erforderlichen Eingaben versorgt.
4. Das Verhalten des Testobjekts kann während der Testdurchführung über Monitoring-Funktionen beobachtet und aufgezeichnet werden. Um den Testablauf überwachen und die Erfüllung bestimmter Strukturtestkriterien prüfen zu können, sind durch Hard- und Software realisierte Funktionen / Schnittstellen bereitzustellen, die während der Testdurchführung eine Aufzeichnung des Programm- bzw. Prozeßgeschehens ermöglichen (beispielsweise Zähler in Programmschleifen).
5. Im Rahmen der Testauswertung sind die Testergebnisse mit den festgelegten Sollergebnissen zu vergleichen. Anhand der Sollwerte oder Sollbedingungen ist zu entscheiden, ob das Verhalten des TO im Test und die Testergebnisse als korrekt zu akzeptieren bzw. als falsch zurückzuweisen sind. Im einfachsten Fall resultiert diese Entscheidung aus dem Soll-/ Istwertvergleich von Variablenwerten. Erst der Vergleich von Ist- und Sollwerten oder Verhalten führt in der Testauswertung zu den Testergebnissen.

Die folgenden Aktivitäten sind TO-übergreifend:

1. Im Rahmen der Testplanung erfolgt die Festlegung der allgemeinen Vorgehensweise beim Test. Dabei wird entschieden, ob beim Testen bottom-up, top-down oder in anderer Weise

vorgegangen werden soll. Anschließend sind die TO und ihre Reihenfolge bei der Testdurchführung festzulegen. Weiterhin werden für jedes TO die Testkriterien definiert, nach denen die Auswahl von Testfällen erfolgen soll. Darüber hinaus sind jeweils Testziele festzulegen, die angeben, in welchem Umfang die festgelegten Testkriterien beim Test zu erfüllen sind (Testabbruchkriterium).

Im Rahmen der Testplanung ist weiterhin für jedes TO die Testumgebung zu definieren. Dazu gehören die Festlegung der beim Test einzusetzenden Hard- und Software-Konfiguration, einschließlich Betriebssystem etc. In der Prozeßdatenverarbeitung stellt auch der technische Prozeß mit seinen Schnittstellen zum Software-System einen wesentlichen Bestandteil der Testumgebung dar. Weiterhin sind Zeiten, Testpersonal und die entsprechenden Verantwortlichkeiten zu bestimmen.

2. Im Rahmen der Testorganisation ist eine Verwaltung für alle im Laufe des Tests relevanten Daten, wie TO, TF (Testfall), Testdaten und Testergebnisse, einzurichten. Für jedes TO ist die Testumgebung bereitzustellen. Bei der Testorganisation ist auf die Regressionsfähigkeit der Tests zu achten. Die Tests müssen jederzeit reproduzierbar und die Ergebnisse vorangegangener Tests für Folgende wiederverwendbar sein.
3. Zur Testdokumentation gehört die Identifikation aller Testobjekte, der entsprechenden Testumgebung und der festgelegten Testreihenfolge. Darüber hinaus ist für jedes TO eine ausführliche Dokumentation erforderlich über die festgelegten Kriterien zur Testfallermittlung, die ausgewählten Testfälle und Testdaten, die entsprechenden Sollergebnisse bzw. die zur Testauswertung definierten Akzeptanzkriterien, die Testergebnisse, die Auswertung der Ergebnisse mit entsprechenden Bewertungen einschließlich des erreichten Überdeckungsgrads sowie die aufgedeckten Fehler mit einer Fehlerstatistik.

[Grimm] [Müllerb] [Wilke]

3.1 Testkonzeption

Grundlage der Softwareentwicklung sind die von den Anwendern der Software (Anforderer) definierten fachlichen Vorgaben.

Im Rahmen der Testfallermittlung werden die fachlichen Anforderungen an die Software in Testfälle umgesetzt. Dazu sind zuerst die Eingabestrukturen der Anwendung zu identifizieren und anschließend sind die Elementklassen zu bilden. Dabei führen alle Werte innerhalb einer Klasse zur selben Wirkung. Elementklassen eines Elements sind disjunkt und bilden den Definitionsbereich, sie haben damit eine eigenständige Wirkung. Eine fachliche Wirkung soll von einem Eingabeelement abhängig sein.

Testfallerzeugung: -TF anlegen;
-Kombinationen zuordnen
-Elementklassen zuordnen

Jede Wirkung muß einmal erzeugt werden, wobei jede Elementklasse mindestens einmal einbezogen wird.

Die Testfälle sind vollständig dokumentiert und abgestimmt, wenn die folgenden Voraussetzungen erfüllt werden:

- Wertebereiche von Datenfeldern,
- Wirkungen und Meldungen des Systems,
- Masken und Maskenbäume,
- Dialogablauf für Dialogfunktionen,
- Zuordnung von Dateien und Datenbanktabellen zu Funktionen,
- Zugriffsarten auf Dateien und Datenbanktabellen.

Die Aufgabe der Testdatendefinition besteht in der Definition der Schnittstellen sowohl für die Dialogeingaben als auch für die Abbildung der Bestandsdaten.

Ziele der Testdatendefinition sind:

- Überprüfung der Funktion auf Korrektheit und Vollständigkeit,
- vollständige Abbildung aller Eingabevarianten und der dazugehörigen Ausgaben in möglichst wenigen Testdatenkombinationen,
- vollständige und eindeutige Dokumentation der Testdaten als Nachweis,
- erleichterte Wartbarkeit von Testdatendokumenten,
- Wiederverwendbarkeit der Testdaten.

Die Testdatendefinition ist abgeschlossen, wenn

- zu jedem Testfall mindestens eine Testdatenkombination definiert ist,
- zu jedem Wertebereich einer Elementklasse jeweils der untere und der obere Grenzwert in je einer Testdatenkombination definiert ist,
- zu jedem in einer Elementklasse einzeln aufgeführten Wert mindestens eine Testdatenkombination definiert ist,
- jede Testdatenkombination eine vollständige Dokumentation der relevanten Eingaben und der im Test zu überprüfenden Ausgaben (Ergebnisse) enthält,
- alle Testdatenkombinationen in mindestens einem Testlauf referenziert sind.

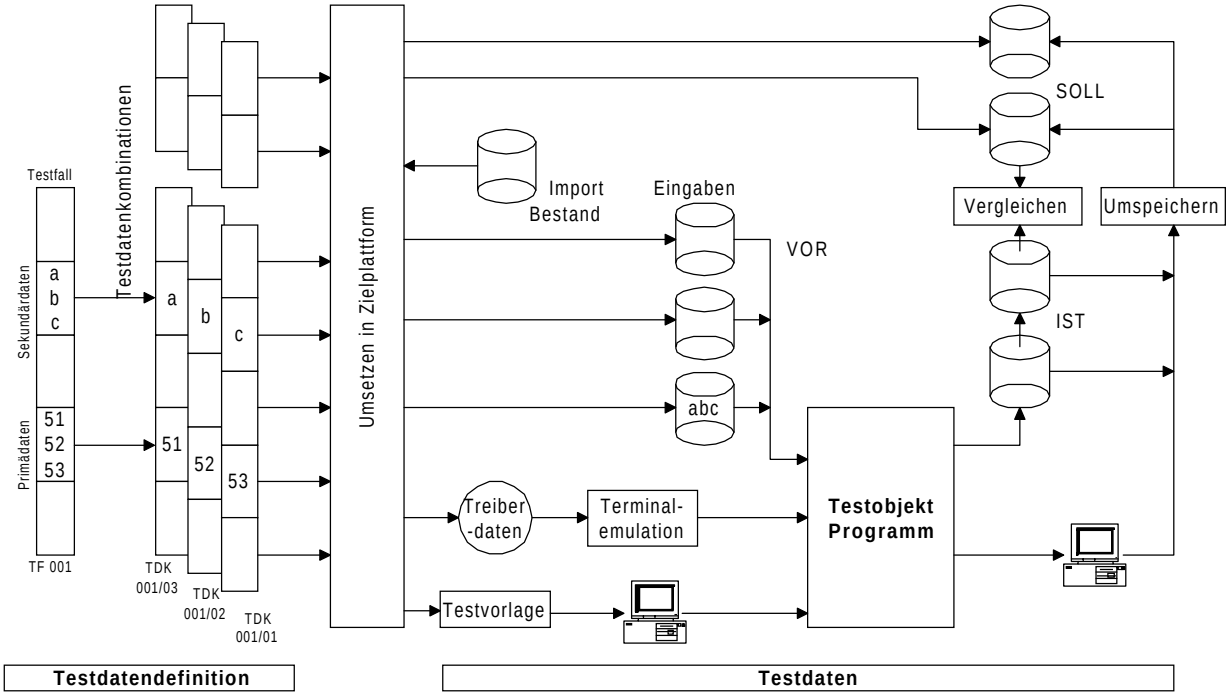


Abbildung 3.1-1: Ergebnisse im Testprozeß [TestDat]

Erläuterungen zu [Abbildung 3.1-1].

Vorgabe für die Testdatendefinition sind Testfälle, die den logischen Zusammenhang zwischen einer für einen Testfall erforderlichen Bestandsdatensituation und den zugehörigen Bewegungsdaten (Windoweingaben oder Eingaben aus einer Steuerdatei) beschreiben.
Im Rahmen der Testdatendefinition werden auf Basis des Testfalls Datenvarianten in Form von Testdatenkombinationen (TDK) abgebildet. Die Testdatendefinitionen führen zunächst die ihnen gemeinsame Testfallbezeichnung „001“ und eine individuelle testdatenkobinations-spezifische Variantennummer. Jede Testdatenkombination erhält aus dem Vorschlag des vorgegebenen Testfalls genau einen Wert der Elementklasse.

Sind alle Testfälle in ausreichend viele Testdatenkombinationen umgesetzt und ist die Testumgebung vorbereitet und das zu testende Programm darin verfügbar, dann kann mit der Testausführung begonnen werden.

[SCHLADWEILR Erweiterung: Anhang – Sonstiges Ablaufplan/diagramm zur Testfallermittlung]

3.2 Abgrenzung der Testobjekte

Durch die Abgrenzung der Testobjekte wird eine Anwendung logisch in überschaubare, leicht zu verwaltende Einheiten zerlegt. Diese sollten sich untereinander möglichst wenig beeinflussen und gleiche Anforderungen an die Sekundär- und Primärdatenbestände stellen, um eine Parallelisierung der Testdurchführung zu erleichtern. Die Testobjektabgrenzung bildet einen Kompromiss zwischen großen Objekten und damit verbundenen Redundanzen und Überschneidungen und kleinen Objekten mit ihrem erhöhten Aufwand bei der Testdurchführung.

3.3 Planung der Teststufen, Testdatendefinition

Innerhalb der Testdatendefinition werden die Wertebereiche der Testdaten, die zur Ausführung eines Tests an einem definierten Testobjekt benötigt werden, festgelegt. Mit den gegebenen Beschränkungen der Zeit, Kosten, Computerzeit etc. wird die Frage, welche Untermenge aller denkbaren Testfälle die größte Wahrscheinlichkeit bietet, möglichst viele Fehler zu finden, zum Hauptproblem der Testdatendefinition.

3.4 Testfallentwurf

Die Testfallbestimmung (Testfallermittlung / Testfallentwurf) kann bei den Funktionstests in 2 Klassen eingeteilt werden.

- Aufgabenorientierte Testfallbestimmung
- Funktionsorientierte Testfallbestimmung

Da erschöpfendes Blackboxtesten unmöglich ist und weil daher jeder Test eines Programms notwendigerweise unvollständig bleibt (d.h Testen bietet keine Garantie für Fehlerfreiheit), ist eine wohlüberlegte Strategie erforderlich, um diese Unvollständigkeit weitestgehend zu reduzieren.

Daraus folgt die Beschäftigung mit der Methode der Testfallentwürfe.

1. schwächste Methode --> Zufallsdaten oder ad-hoc-Testfallentwurf
2. White-box-test
3. Black-box-test
4. stärkste Methode --> Kombination von Black- und White-box-test, da jede Testmethode andere Arten von Fehlern findet !

Zur systematischen Erstellung von Testfällen existieren eine Vielzahl von Testfallentwurfsmethoden, die im Folgenden vorgestellt werden.

3.4.1 Logic-coverage Testing

Beim Logic-coverage Testing wird der Testdeckungsgrad gemessen.

1. Jede Anweisung im Programm muß mindestens einmal ausgeführt werden.
2. besser: In jeder Entscheidung muß der Ausgang für die erfüllte Bedingung und der Ausgang für die nichterfüllte Bedingung benutzt und jeder Eingang mindestens einmal angesprochen worden sein. In Programmen mit Entscheidungen, die Mehrfach-bedingungen verwenden, ist das Minimalkriterium damit eine hinreichende Anzahl von Testfällen, die alle möglichen Kombinationen von Bedingungen in einer Entscheidung berücksichtigen und alle Eingangspunkte in das Programm ansprechen.
3. Bei Mehrfachbedingungen ist das Minimalkriterium eine hinreichende Zahl von TF, die alle möglichen Kombinationen von Bedingungen in einer Entscheidung berücksichtigen und jeden Eingang ansprechen.
   - 3.1. Ausführung aller Programmpfade (nicht machbar aufgrund von Schleifen),
   - 3.2. Ausführung aller Anweisungen je einmal (nicht 100% aussagefähig, da z.b. falsch implementierte Funktionen nicht erkannt werden),
   - 3.3. Erfassung / Ausführung aller Entscheidungen oder Sprünge (decision oder branch coverage) – jeder *then* und jeder *else*-Zweig muß mindestens einmal durchlaufen werden. Damit wird auch jede Anweisung des Programms durchlaufen, vorausgesetzt es enthält Entscheidungen. Diese Methode wird als die Effektivste angesehen.
   - 3.4. Erfassung / Ausführung aller Bedingungen (condition coverage), d.h. jeder Eingang einer Entscheidung wird berücksichtigt,
   - 3.5. Erfassung / Ausführung aller Mehrfachbedingungen, d.h.alle Kombinationen der Bedingungen einer Entscheidung werden berücksichtigt,

Problem von 3.3 und 3.4: Es werden keine Fehler in logischen Ausdrücken berücksichtigt.

3.4.2 Äquivalenzklassenbildung

Bei der Äquivalenzklassenbildung als Vertreter der Blackboxstrategie, geht es um die Bildung von Untermengen der Eingabedaten, die die höchste Wahrscheinlichkeit bieten, Fehler zu finden. Vorteilhaft daran ist die Reduzierung der Anzahl an notwendigen Testfälle, die man entwickeln muß. Weiterhin überdeckt die Äquivalenzklassenbildung eine große Menge anderer Testfälle. D.h., sie zeigt die Abwesenheit oder die Anwesenheit von Fehlern in einem speziellen Satz von Eingabedaten. Man erwartet, dass alle anderen Testfälle in der Äquivalenzklasse das gleiche Testresultat produzieren.

Entwurf von Testfällen mit Hilfe der Äquivalenzklassenbildung:

1. Bestimmung der Äquivalenzklasse (ÄK)
2. Definition der Testfälle

Bei der Bestimmung der ÄK erfolgt nach dem Erfassen der Bedingungen eine Zuordnung in gültige und ungültige Äquivalenzklassen. *Gültige (erlaubte) ÄK* sind die entsprechend der Leistungsbeschreibung geforderten Eingabedaten, die das Programm verarbeiten muß. *Ungültige ÄK* sind die Eingabedaten, die das Programm als fehlerhaft (unerlaubt) bewerten soll. Nachteilhaft an der Äquivalenzklassenmethode ist, daß keine Kombinationen von Eingabebedingungen berücksichtigt werden.

Prinzipien zur Ermittlung der ÄK an einem Beispiel:

1. Einteilung bei vorgegebenen Wertebereich in gültige ÄK (z.B.: 1<=Wert<=999) und in ungültige ÄK (Wert<1 && 999<Wert),
2. bei vorgegebener Anzahl von Werten (beispielsweise bis zu 6 Usern) eine gültige ÄK und zwei ungültige (kein User und mehr als 6 User),
3. bei verschiedenen Eingabewerten jeweils einen gültigen und einen ungültigen Wert festlegen,
4. bei Eingabewerten mit bestimmtem Zustand (z.b. alphanumerischer Wert) muß ein gültiger und ein ungültiger Wert in ÄK gebildet werden,
5. bei der Vermutung, daß Elemente einer ÄK unterschiedlich vom Programm behandelt werden, muß die ÄK in zwei oder mehrere unterteilt werden,

Prinzipien zur Definition der TF aus ÄK:

1. eindeutige Nummerierung der ÄK,
2. Schreiben jeweils eines neuen TFs für möglichst viele der bisher unbehandelten gültigen ÄK, bis alle gültigen ÄK geprüft sind,
3. Schreiben jeweils eines TFs für einen und nur einen der bisher unbehandelten ungültigen ÄK bis alle ungültigen ÄK geprüft sind. Die ungültigen ÄK werden individueller behandelt, da bestimmte Fehlerüberprüfungen andere Prüfungen verdrängen oder maskieren.

3.4.3 Grenzwertanalyse

Bei dem Testfallentwurf mittels der Grenzwertanalyse werden Situationen, in denen die Testfälle Werte an den Grenzen oder den Rändern der Ein/ Ausgabeäquivalenzklassen (den Grenzwert selbst, direkt unter oder über dem Grenzwert) berücksichtigen. In zwei Beziehungen ergänzt die Grenzwertanalyse die Äquivalenzklassenbildung.

1. In der Grenzwertanalyse muß jeder Rand einer Äquivalenzklasse in einem Testfall auftreten, im Gegensatz zur Auswahl eines beliebigen Elements der ÄK, welches für diese als repräsentativ angesehen wird.
2. Die Aufmerksamkeit gilt nicht nur den Eingabebedingungen (Eingaberaum), sondern es werden auch Testfälle entworfen, die den Ergebnisraum berücksichtigen (d.h. Ausgabeäquivalenzklassen - Testfälle).

[Myers]

3.4.4 Ursache-Wirkungs-Graph

Die Grenzwertanalyse und Äquivalenzklassenbildung haben eine Schwäche gemeinsam. Sie können keine Kombination von Eingabebedingungen untersuchen. Die Technik des Ursache-Wirkungs-Graph dient der effizienten, systematischen Auswahl einer Menge erfolgreicher Testfälle. Weiterhin ergibt sich dabei ein Hinweis auf Unvollständigkeiten und Zweideutigkeiten in der Spezifikation. Ein Ursache-Wirkungs-Graph ist eine formale Sprache, in die eine Spezifikation aus der natürlichen Sprache übersetzt wird. Der Graph entspricht einer Schaltung der Digitallogik (ein kombinatorisches logisches Netz) mittels logischer Operatoren der Booleschen Algebra (AND;OR;NOT).

Folgendes Verfahren wird zur Testfallermittlung angewendet:

1. Die Spezifikation wird in Testobjekte unterteilt.
2. Ursachen und Wirkungen werden formuliert (d.h. Ursache ist [Äquivalenzklasse von -] Eingangsbedingung(en); Wirkung ist eine Ausgabebedingung oder eine Systemtransformation) D.h. es werden alle Wörter einer Spezifikation unterstrichen und durch eine eindeutige Zahl gekennzeichnet, die Ursache und Wirkung beschreiben.
3. Der semantische Inhalt der Spezifikation wird analysiert und in einen booleschen Graphen transformiert, der die Ursachen und Wirkungen verbindet. Das ist der Ursache-Wirkungs-Graph.
4. Der Graph wird mit Kommentaren versehen, die Kombinationen von Ursachen und/oder Wirkungen angegeben, die aufgrund syntaktischer oder kontextabhängiger Beschränkungen nicht möglich sind.
5. Nachdem man die Zustandsbedingungen in dem Graphen methodisch verfolgt, wird der Graph in eine Entscheidungstabelle mit beschränkten Eingängen umgesetzt. Jede Spalte oder Tabelle stellt einen Testfall dar.
6. Die Spalten der Entscheidungstabelle werden in die Testfälle konvertiert.

[Myers]

Am schwierigsten stellt sich die Umsetzung des Ursache-Wirkungs-Graphen in die Entscheidungstabelle dar. Dafür gibt es eine Vielzahl von technischen Hilfmitteln, diesen Vorgang der Transformation von einer logischen Darstellung in die andere zu vollziehen.

3.4.5 Error guessing

Error guessing (Fehlererwartung) ist eine Methode zur TF-Ermittlung, die die vorher beschriebenen Methoden um weitere Testfälle ergänzt. Es handelt sich dabei um eine TF-Entwurfsmethode, die nach intuitiven Gesichtspunkten erfolgt. Testfälle werden aufgrund der Erfahrung und der Intuition des Testers gebildet (ad hoc). Entsprechend der Programmspezifikation ist zu überlegen, an welchen Stellen gewisse Besonderheiten zu testen wären, die mit hoher Wahrscheinlichkeit Fehler enthalten.

3.4.6 Die Klassifikationsbaum-Methode

3.4.6.1 Grundidee

Die grundsätzliche Idee der Klassifikationsbaum-Methode ist es, zuerst die Menge der möglichen Eingaben für das TO getrennt auf verschiedene Weisen unter jeweils einem geeigneten Gesichtspunkt zu zerlegen, um dann durch Kombination dieser Zerlegungen zu Testfällen zu kommen.

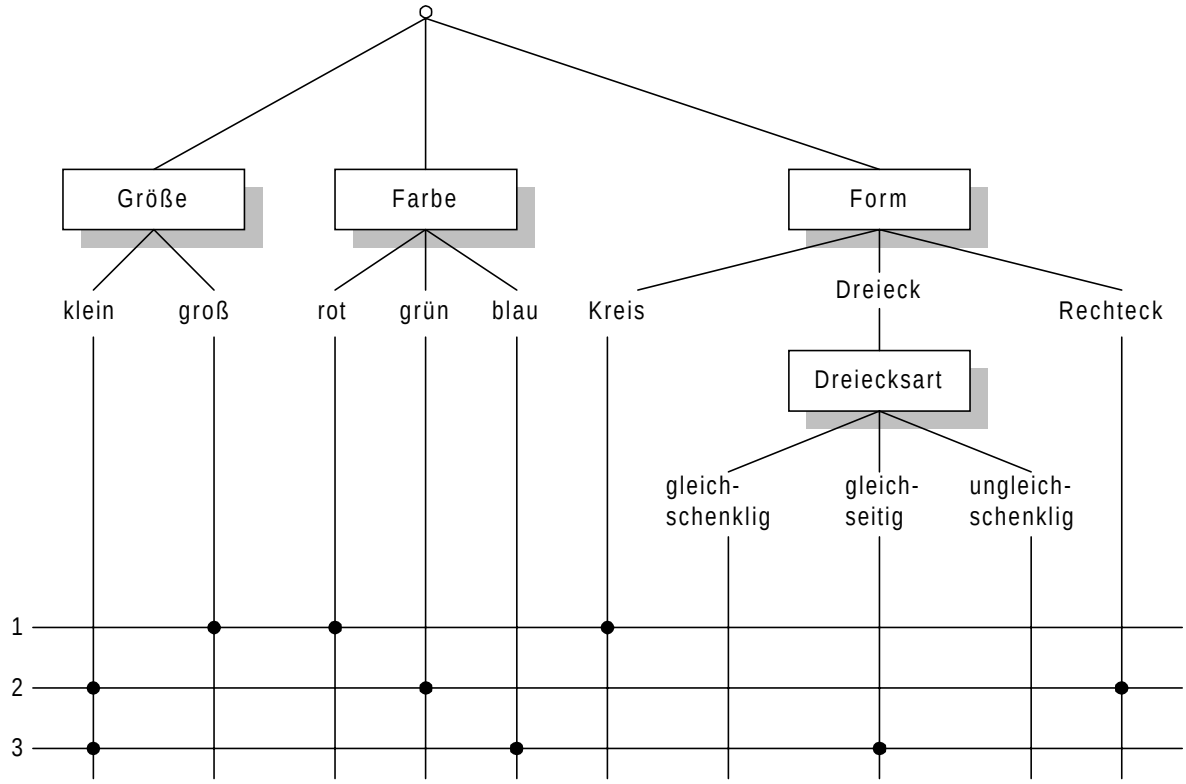


Abbildung 3.4-1: Klassifikationsbaum für ein einfaches Beispiel [Myers]

Anhand eines einfachen Beispiels [Abbildung 3.4-1] soll die Grundidee der Klassifikationsbaum-Methode erläutert werden. Das Testobjekt ist ein Erkennungssystem, dessen Aufgabe es ist, die Größe unterschiedlicher geometrischer Objekte zu bestimmen, die auf einem Transportband an einer Kamera vorbeilaufen. Die verschiedenen möglichen Eingaben sind hier unterschiedliche Bausteine. Die Aufgabe der Klassifikationsbaum-Methode besteht darin, diesen Eingabedatenraum zu strukturieren und mittels der Spezifikation eine endliche Anzahl relevanter Testfälle herauszufinden.

3.4.6.2 Einsatz der Klassifikationsbaum-Methode in der Praxis

Die Klassifikationsbaum-Methode hat sich im praktischen Einsatz als leistungsfähige Unterstützung für die systematische Testfallermittlung erwiesen. Die wesentlichen Stärken der Methode sind:

- Die Methode ermöglicht sowohl den Test an Normalfällen, Grenzwerten und "ungültigen" Eingabefällen, die jeweils als Klassen in die entsprechenden Klassifikationen aufgenommen werden können, als auch die systematische Auswahl testrelevanter Kombinationen von verschiedenen Eingabekonstellationen durch die explizite Markierung von Klassenkombinationen in der Tabelle.

- Durch die Anwendung der Klassifikationsbaum-Methode wird der Testfallermittlungsprozeß in kleine, einzelne und leicht handhabbare Teilschritte zerlegt. Jeder Schritt bleibt überschaubar und nachvollziehbar. Insbesondere die vollständige, disjunkte Fallunterscheidung wird dadurch extrem vereinfacht, sodaß sie jeweils unter nur einem Gesichtspunkt vorgenommen werden muß.
- Der Tester wird bei der Auswahl der testrelevanten Teilmengen des Eingabedatenraums systematisch geleitet, ohne daß seine Kreativität, beispielweise bei der Definition testrelevanter Gesichtspunkte, eingeschränkt wird.
- Der Einsatz der Klassifikationsbaum-Methode stellt keinerlei Anforderungen an die Form der funktionalen Spezifikation. Sie ist sowohl auf informelle als auch halbformale und formale Spezifikationen anwendbar.
- Mit Hilfe des Klassifikationsbaums und der baumbezogenen Maße kann die Komplexität des Testproblems sowohl graphisch als auch quantitativ verdeutlicht werden. Die entsprechenden Überdeckungsmaße können als Basis für die Beurteilung des Umfangs, in dem die Funktionalität des TO getestet wurde, verwendet werden. Zusammen mit der im Rahmen der Teststrategie durchgeführten Messung der beim Test erreichten Überdeckung des Testobjekts ergibt sich in der Regel eine gute Basis für die Beurteilung des gesamten Testumfangs.
- Der Testaufwand kann bewußt, das heißt in Kenntnis der Komplexität des Testproblems, durch gezielte Beschränkung auf bestimmte testrelevante Gesichtspunkte, Fallunterscheidungen und Klassenkombinationen jederzeit auf ein vertretbares Maß eingestellt werden. Durch die systematische Vorgehensweise und die explizite, graphisch visualisierte Definition der für den Test ausgewählten unterschiedlichen Klassen und Klassenkombinationen kann Redundanz bei der Testfallermittlung vollständig vermieden werden.
- Bei der Kombination verschiedener Eingabekonstellationen zu Testfällen wird der Testaufwand bei der Klassifikationsbaum-Methode bereits automatisch dadurch beschränkt, daß die Klassen nur klassifikationsübergreifend kombiniert werden können und dadurch die Kombination logisch nicht vereinbarer Eingangsbedingungen zum großen Teil von vorherein konstruktiv ausgeschlossen wird.
- Die Klassifikationsbaum-Methode erweist sich im praktischen Einsatz als leicht erlernbar und auch als gut handhabbar. Wesentliche Bedeutung kommt dabei der graphische Repräsentation mittels Klassifikationsbaum und Kombinationstabelle zu, die eine sehr anschauliche Form der Testfallermittlung und –spezifikation gewährleistet. Die Handhabbarkeit bei komplexeren Testproblemen wird durch die zur Verfügung stehenden Verfeinerungsmechanismen gefördert.
- Die Klassifikationsbaum-Methode kann in verschiedenen Software-Entwicklungsphasen zur Testfallermittlung herangezogen werden und damit sehr gut in den gesamten

Entwicklungsprozeß eingebettet werden. Die praktischen Erprobungen haben gezeigt, daß dabei die Betrachtung des Systems oder seiner Teile unter verschiedenen Gesichtspunkten und die Kombination verschiedener, unter diesen Gesichtspunkten definierter Eingabebedingungen, die Aufdeckung von Unvollständigkeiten und Widersprüchen in der Spezifikation unterstützen und dadurch sehr frühzeitig zu einer Verbesserung der Spezifikation beitragen.

Die Testfallentwurfsmethode bietet die Möglichkeit der Rechnerunterstützung für die methodische Vorgehensweise sowohl für den Funktions- als auch den Strukturtest.

3.4.7 Strategie zur Testfallermittlung

Als Strategie zur Testfallermittlung sollten entsprechend den Erfordernissen, die an die Tests gestellt werden (Systemtest, Funktionstest, Modultest o.a.), die Methoden zur TF-Ermittlung zweckmäßig kombiniert werden. Grundsätzlich sollte folgende Vorgehensweise sequentiell bevorzugt werden:

1. Aufstellen des Ursache-Wirkungs-Graphen
2. Grenzwertanalyse
3. Anlegen von gültigen und ungültigen Äquivalenzklassen
4. Ermittlung ergänzender Testfälle mittels error-guessing
5. Untersuchung der Programmlogik – logic coverage zur Erfüllung bisher nicht berücksichtigter Bedingungen

Die Reihenfolge der TF-Ermittlung läßt sich zwischen Punkt 1) und 3) vertauschen oder kann zeitgleich erfolgen, da für die Ursache-Wirkungs-Graphen auch Äquivalenzklassen ausreichende Grundlage als Bedingungen bilden können.

3.5 Testprozedurerstellung

Eine Testprozedur enthält alle Arbeitsanweisungen zum Ablauf des Testprozesses eines Testobjektes. Die Erstellung der Testprozeduren beschreibt die Aktivitäten des Einrichten der Testdateien, Sichern/Laden von Testbeständen, Starten des Testlaufs, Starten der Mitschnitte, Sichern der Ergebnisse für den Soll/Ist-Vergleich, Selektieren von Daten für die Testauswertung, Durchführen von Vergleichsläufen, Übernahme von Datenbeständen, Restart der Testdurchführung [siehe Abbildung 8.1-2: Testumgebung].

3.6 Testausführung

Innerhalb der Testausführung werden die bei der Testprozedurerstellung definierten Arbeitsabläufe zur Ausführung in der Testumgebung gebracht. Dazu gehören beispielsweise:

- Zusammenstellen des Testablaufs,
- Laden der DB/Dateien aus den Bestandsdaten (Primärdaten),
- Zwischenergebnisse definieren,
- Testausführung eines Testlaufs,
- Sichern und Aufbereiten der Ist-Ergebnisse,
- Messungen an repräsentativen Punkten / Breakpoints sowie Ausführungszeiten,
- Prüfen der Ergebnisse,
- Sichern von Eingabe-/Ausgabedaten.

Gegebenenfalls sind die Schritte bei der erstmaligen und wiederholten Testausführung unterschiedlich. Ein Testlauf beinhaltet im Intranet-Testbed-Kontext die Testausführung der Testfälle einer begrenzten Anzahl von Testobjekten, die auf die selben Bestandsdaten zugreifen.

3.7 Testauswertung und Bewertung

Die Ausführung der Tests liefert u.a. die folgenden Ergebnisse/Veränderungen, welche ausgewertet und bewertet werden:

- Vergleich von DB / Datei-Inhalten (manuell / maschinell),
- Ausführungsstatistik,
- Fehlerlokalisierung,
- Soll-Ist-Wert-Vergleich,
- Übernahme / Ändern des Sollwertbestandes.

4 Testkriterien

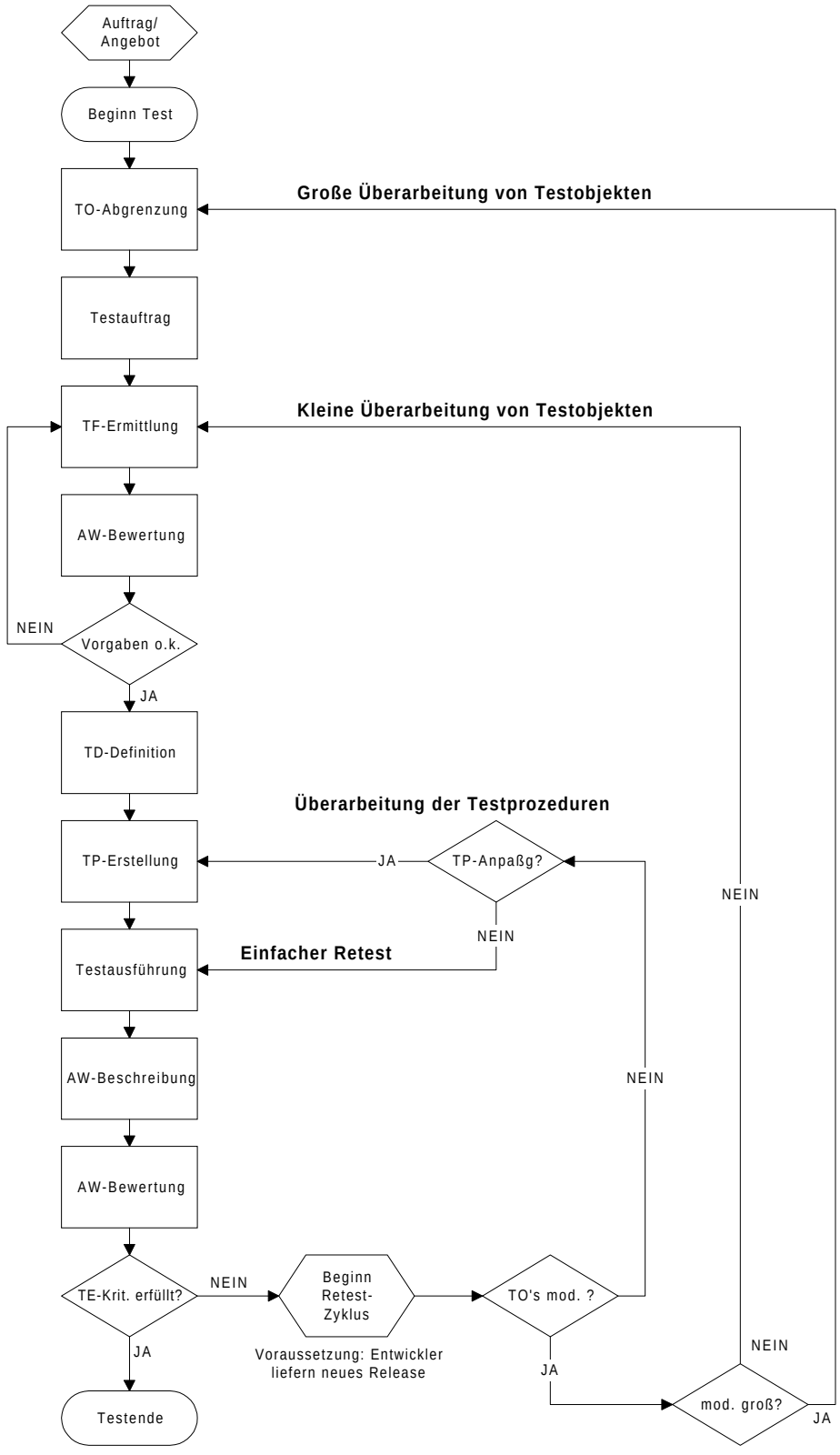
Nach [Myers] sollen in alle Phasen des Testens die folgenden Prinzipien Beachtung finden.

- ein notwendiger Bestandteil eines Testfalls ist die Definition der erwarteten Werte und des Resultats,
- Testen ist ein Prozeß, ein Programm mit der Absicht auszuführen, Fehler zu finden,
- ein positiver Test ist ein Test, bei dem ein Fehler gefunden wird,
- Testen garantiert nicht die Fehlerfreiheit von Programmen,
- Black- und Whiteboxtests sollten zweckmäßig kombiniert werden,
- ein Programmierer sollte nicht versuchen, sein eigenes Programm zu testen,
- eine Programmierorganisation sollte nicht ihre eigenen Programme testen,

- die Ergebnisse eines jeden Tests sind auf Plausibilität zu überprüfen,
- Testfälle müssen für ungültige und unerwartete ebenso wie für gültige und erwartete Eingabedaten definiert werden,
- es sind auch Testfälle zu entwickeln, die untersuchen, ob ein Programm etwas tut, was es nicht tun soll,
- Planung der Testfälle auf Wiederverwendbarkeit,
- Testfälle sind unter der Annahme zu planen, daß Fehler gefunden werden. Die Wahrscheinlichkeit für die Existenz weiterer Fehler in einem Programmabschnitt ist proportional zur bereits entdeckten Anzahl der Fehler.
- Ein Softwarefehler ist vorhanden, wenn ein Programm nicht das tut, was der Nutzer vernünftigerweise von ihm erwartet.
- Testen ist eine extrem kreative und intellektuell herausfordernde Aufgabe,
- Ein guter Testfall ist dadurch gekennzeichnet, daß er mit hoher Wahrscheinlichkeit einen bisher unbekannten Fehler zu entdecken imstande ist.
- Ein erfolgreicher Testfall ist dadurch gekennzeichnet, daß er einen bisher unbekannten Fehler entdeckt.
- wirtschaftliche Betrachtungen beim Programmtesten,
- Verwendung von Testfällen, die die größte Wahrscheinlichkeit bieten, möglichst viele Fehler zu finden,
- Es gibt zwei Möglichkeiten, warum die erhaltenen Testergebnisse nicht mit den erwarteten Ergebnissen übereinstimmen:
  - A) das Testobjekt enthält einen Fehler
  - B) der Testfall ist falsch.

  [Myers]

Die folgende [Abbildung 3.7-1: Testablaufplan] verdeutlicht die funktionalen Zusammenhänge der Testausführung.



Abkürzungen:

TD-	Testdaten
TE-	Testendekriterien
TO-	Testobjekte
mod. groß? -	Müssen nur TF einzelner TO's modifiziert werden, oder zerfallen, entstehen, entfallen, verschmelzen TO's?
TP-	Testprozeduren
AW-	Abweichung-

Abbildung 3.7-1: Testablaufplan

Der dargestellte Testablaufplan in [Abbildung 3.7-1] unterteilt die Modifikation der Testobjekte im Kleinen und im Großen. Modfikation im Kleinen setzt voraus, daß das Testobjekt schrumpft oder wächst. Damit entfallen oder entstehen Testfälle oder bisherige werden angepaßt. Bei der Modifikation der Testobjekte im Großen entstehen neue Testobjekte, entfallen, verschmelzen oder zerfallen in mehrere Testobjekte

4.1 Client-Server Anwendungen

Ausgehend vom Server-Prinzip der lokalen Netze und in Verbindung mit dem "Downsizing-Konzept" entstand das Client/Server-Modell (CS-Modell). Als allgemeinstes Konzept der verteilten Verarbeitung hat CS etwas mit Software-Architektur zu tun. Entsprechend bezeichnet man sie auch allgemein als Client/Server-Architektur.
Im traditionellen Softwaredesign wird das Konzept der modularisierten Anwendung über eine Trennung in Haupt- und Unterprogramm vorgenommen. Das Hauptprogramm ruft das Unterprogramm auf, im Unterprogramm erfolgt eine/die Informationsverarbeitung/-darstellung etc. und anschließend gibt das Unterprogramm die Steuerung wieder an das Hauptprogramm ab. Beim Client/Server-Konzept befindet sich die Rechnergrenze zwischen Haupt und Unterprogramm. Client/Server bietet also durch die Middleware (Betriebssystem, Kommunikationssoftware) die Möglichkeit, daß ein Programm die Arbeit oder Leistung auf unterschiedliche Rechner verteilt.
Client/Server Systeme lassen sich durch folgende Unterscheidungsmerkmale charakterisieren:

Dienste: Client/Server bezeichnet vor allem eine Beziehung zwischen Prozessen, die auf verschiedenen Rechnern laufen. Der Serverprozeß ist ein Dienstanbieter; der Client nimmt Dienste in Anspruch. Im Wesentlichen bietet die Client/Server-Verarbeitung eine saubere Trennung von Funktionen, basierend auf dem Dienstekonzept.
Gemeinsame Ressourcen: Ein Server kann mehrere Clients gleichzeitig bedienen und ihren Zugriff auf gemeinsam benutzte Ressourcen regeln.
Asymetrische Protokolle: Es besteht eine n:1-Relation zwischen Clients und Server. Clients initiieren den Dialog, indem sie einen Dienst anfordern.
Standort-Transparenz: Der Server ist ein Prozeß, der auf demselben Rechner wie der Client oder auf einem beleibigen anderen Computer im Netzwerk läuft. In der Regel verbirgt die Client/Server-Software den Standort des Servers. Ein Programm kann ein Client, ein Server oder auch beides sein.
Austauschbarkeit: Die ideale Client/Server-Software ist unabhängig von Hardware- und Betriebssystemplattform. Man sollte die Plattformen für Client und Server beliebig wechseln können.

Nachrichtenbasierter Datenaustausch: Client und Server sind lose gekoppelte Systeme, die über einen Mechanismus zum Austausch von Nachrichten interagieren. Die Nachrichten bilden den Transportmechanismus für Dienstanforderungen und Antworten. Als Merkmal des nachrichtenbasierten Datenaustauschs verwenden CS-Systeme die Verkapselung von Diensten durch Verwendung offener Nachrichtenschnittstellen. Die Skalierbarkeit von CS-Systemen (horizontal oder vertikal) durch Veränderung der Client- oder Serverzahl ermöglicht eine flexible Umgebung.
Integrität: Code und Datenbasis des Servers werden zentral gepflegt; Clients sind dezentral und unabhängig;

4.1.1 Die Architektur von CS-Anwendungen

TYP	IMPLEMENTIERUNGSBEISPIEL
Verteilte Präsentation	X-Windows-Anwendungen
Remote Präsentation	MS-Windows-CS-Anwendungen
Verteilte Anwendungen	"echtes" Client-Server
Remote Datenhaltung	Datenbankserver / Fileserver, Datenserver
Verteilte Datenhaltung	redundante Datenhaltung, verteilte Datenbanken über DBMS, lokaler Datenserver
Verteilte Objekte	CORBA-Anwendungen

Tabelle 4.1-1: Typen von CS-Anwendungen [TestDat]

- Distributed Presentation: die Art der Kommunikation zwischen jeder X-Windows-Anwendung und dem X-Window
- Remote Presentation: eine Designentscheidung des Softwareentwicklers, Trennung von Darstellung und "Berechnung"
- Distributed Function: konsequentes CS-Prinzip, geht in „Verteilte Objekte“ über
- Remote Data Management: remote Datenhaltung, Fileserver, Datenbankserver, Datenserver
- Distributed Data Management: verteilte Datenhaltung, das DBMS verwaltet die Daten über mehrere Rechner
- CORBA: verteilte Objekte, Funktionen oder Leistungen werden im Netz transparent als verteilte Objekte von einem Objektmanagementsystem zur Verfügung gestellt (auch unabhängig der Plattform / Programmiersprache), der Client greift nicht mehr auf einen konkreten Rechner (Server) zu, sondern wird vom Objektmanagementsystem zu der Ressource geführt. [CORBA-Common Object Request Broker Architecture]

[SQS-QT]

4.1.2 Besonderheiten bei Tests an CS-Anwendungen

Client/Server Architekturen ermöglichen die Entwicklung komplexer Systeme, welche aus Bestandteilen/Komponenten zusammengesetzt sind. Häufig kommt es auch zum Einsatz von Komponenten aus der Entwicklung Dritter, auf welche nicht im Source zugegriffen werden kann. Diese kommunizieren über Schnittstellen, welche nicht in Kombination mit anderen verwendeten Komponenten getestet wurden. Damit kommt dem Funktionstest, dem Konfigurationsmanagement und der System-Integration eine höhere Bedeutung zu. Das Hauptproblem bei solchen Systemen ist damit die Definition und die Verwendung der definierten Schnittstellen.

Weitere Begriffe zur Beschreibung von CS-Systemen:

Rightsizing: globales Designprinzip von Informationssystemen mittels CS (down sizing / up sizing), Weg hin zu dezentraler Datenhaltung und über kooperative Mechanismen die Daten und die mit der speziellen Anwendung verknüpften Funktionen/Methoden allgemein zur Verfügung zu stellen (Objektorientierung; Interoperabilität von Anwendungen),

offene Systeme: unabhänig von Hard-/Software-kombinationen; hohe Austauschbarkeit von Komponenten und Skalierbarkeit von Rechnerleistung im Netz, Bsp.: POSIX.1 ; Anwendungen nach POSIX.1 laufen nach Compilierung / Binden auf beliebigen Hardware / Betriebssystemplattformen; (z.b. Anwendungen für ODBC-Schnittstelle),
[ODBC-Open Data Base Connectivity]

4.1.3 Spezielle Probleme beim Testen an CS-Anwendungen

Mit den angegebenen Eigenschaften von CS-Systemen ergeben sich u.a. die folgenden Probleme bei der Durchführung von Tests:

- Verteilung von Daten ggf. redundant auf allen Rechnern (Client(s) / Server(n)),
- Konfigurationsmanagement auf Ebene "Gesamtsystem",
- Prozesse laufen ggf. asynchron, parallel,
- Transaktionsverarbeitung komplexer (Authentifizierung etc.),
- Netzlastproblematik (Performance),
- Testwerkzeuge /- automatisierung auf Netzwerkprotokollebene,
- Testumgebung ist in der Regel nur mit einer typischen Konfiguration ausgestattet (IP oder IPX),
- gemeinsame Komponente NETZ,

daraus folgt: Zuverlässigkeit und Performance der Anwendungen selbst wird durch die Zuverlässigkeit und Performance des gemeinsamen Mediums NETZ beeinflusst und umgekehrt,

- Socketproblematik unterhalb der virtuellen Schicht.

4.1.4 Spezielle Testaufgaben an CS-Anwendungen

Als spezielle Testaufgaben an CS-Anwendungen ergeben sich:

- Retesten der Anwendung für alternative virtuelle Rechner (technischer Systemintegrationsdienst),
- Testen der Standard-Dienste (HW/SW),
- Kompatibilität, techn. Systemintegration.

[SQS-QT]

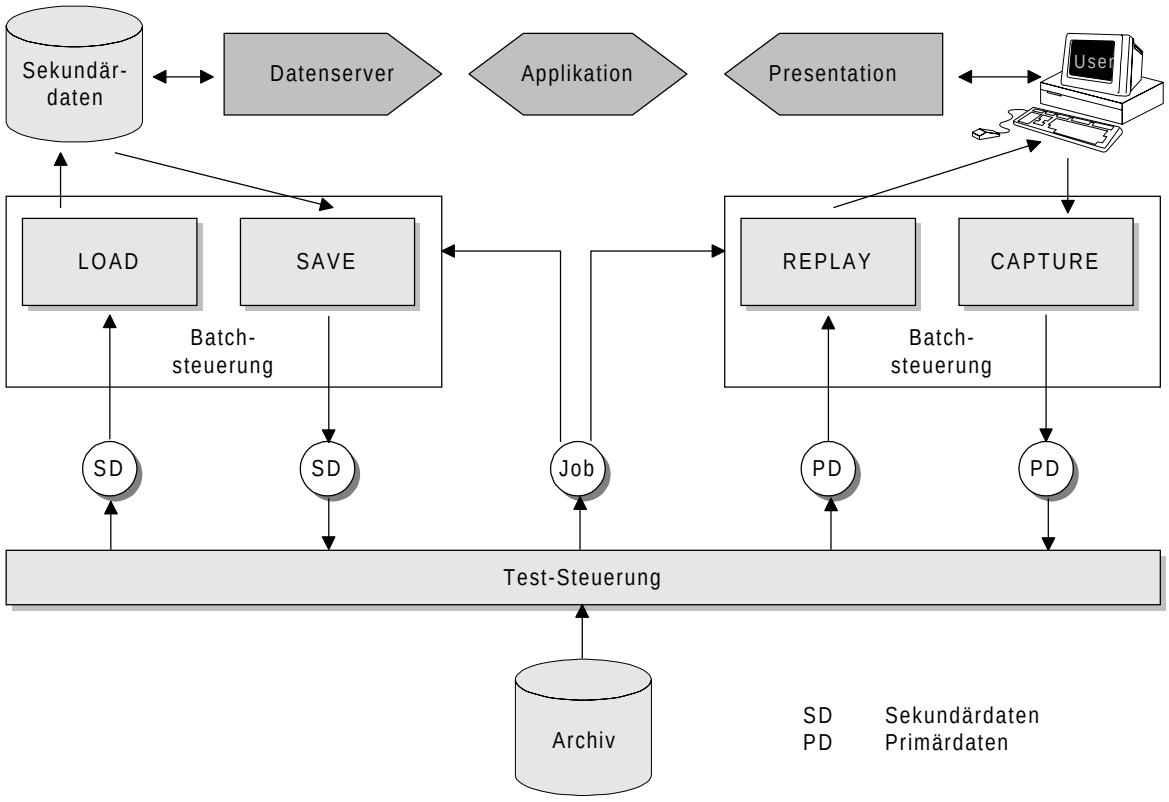


Abbildung 4.1-1: Technische Organisation im Client-/Server- Umfeld [SQS-QT]

4.2 Automatisierung der Anwendungstests

Die Automatisierung des Anwendungstests kann grundsätzlich auf zwei unterschiedlichen Wegen, auf GUI und auf Protokoll-Ebene, erfolgen.

4.2.1 Test auf GUI / CUI-Ebene

Zur Automatisierung von Testabläufen auf GUI-Ebene stehen verschiedene Lösungsansätze, welche unterschiedlich tief in das System eingreifen, zur Auswahl. Programme, welche die Mausbewegung aufzeichnen und abspielen können, greifen wenig in das System ein, haben allerdings Probleme mit unterschiedlichen Bildschirmauflösungen, da sie mit absoluten Mauskoordinaten auf dem GUI arbeiten, weiterhin kann bei höherer Rechnerbelastung das GUI teilweise noch nicht aufgebaut / aktualisiert sein, wenn eine Aktion ausgelöst wird. Vorteilhaft ist der Ort des Capture / Replay, direkt an der Mensch-Maschine-Schnittstelle.

Die professionelleren Tools der Rubrik Capture / Replay arbeiten direkt auf der Windows-API / X-Server oder starten die zu untersuchende Applikation im Windows-Debugg-Modus. Vorteilhaft an dieser Vorgehensweise ist die Flexibilität der aufgezeichneten Scripte, die Möglichkeit Programme ohne GUI im Hintergrund zu steuern (Performance). Nachteilig wirkt sich die Abhängigkeit von der Windows-API (nicht jedes GUI-Object basiert auf Objecten der API z.B. Access; WinAmp) und die hohe Eingreiftiefe in das System aus.

4.2.1.1 Expect

Expect ist eine Erweiterung der Programmiersprache Tcl zur Steuerung interaktiver „command-line“ und „character-graphic“ (z.B. Norton Commander) Applicationen. Expect kann auch mit Tk zur Entwicklung von GUI-Applikationen eingesetzt werden. Bekannte Vertreter von CUI-Programmen sind ftp, telnet und passwd. Im Anhang ist ein GUI-Programm vorgestellt, welches das Unix-Kommando passwd zur Änderung des Paßworts eines Users verwendet. Dabei übernimmt Expect die Wiederholung des Paßworts. Die wichtigsten Kommandos (spawn, send, expect, interact) werden im Folgenden an einem kurzen Beispiel zur Steuerung einer ftp-Session vorgestellt.

```
spawn ftp $host                ;#start der interactiven ftp-session zu host
expect "Name"                  ;#erwateter String "Name"
send "anonymous\r"             ;#sende Name mit anschließendem Carrige-Return (CR)
expect "Password:"             ;#erwarteter String "Password:"
send "email@host\r"            ;#senden einer eMail-Adresse mit CR
```

```
expect "ftp>"                ;#erwarteter String "ftp>"
send "cd pub\r"              ;#Verzeichnisebene wechseln
interact                     ;#Abgabe der gesteuerten Session an interactive Benutzersteuerung
```

Alle gesendeten Variablen und Kommandos können in Schleifen frei parametrisiert werden und eignen sich damit zur Automatisierung interactiver Programme, welche auf dem Character-User-Interface (CUI) beruhen.
[Explor]

4.2.1.2 Windows-Messages

Fensterorientierte, graphische Benutzeroberflächen wie Windows können mehrere Programme quasi parallel (multitasking) laufen lassen. Das Betriebssystem enthält einen Event-Handler, der alle Ereignisse (Tastatur, Maus, Pheripherie) zentral erfaßt und sie über Messages an die entsprechenden Anwendungen beziehungsweise deren Fenster verteilt. Jedes Programm richtet innerhalb seines Hauptprogramms *WinMain* eine sogenannte Nachrichtenschleife (Message Loop) ein, welche die eintreffenden Nachrichten (Messages) entgegennimmt *(GetMessage)*, verarbeitet *(TranslateMessage)* und weiterleitet *(DispatchMessage)*. Mit der eintreffenden Messages wird das entsprechende Fenster adressiert und erhält die eigentliche Nachricht *(PostMessage)*. Mittels den Windows eigenen Hooks-Funktionen *(z.B. WH_KEYBOARD für die meisten Tastatureingaben)* können andere Programme in diesen Nachrichtenaustausch eingreifen und die Art der Nachrichten verändern oder auch ohne Benutzeraktionen Nachrichten an Anwendungen senden. Nach diesem Prinzip funktionieren Macrorecorder und kommerzielle Tools zur Steuerung von GUI-Programmen.
[ct05/99]
Im Anhang wird ein Programm vorgestellt, welches die Windows-Hooks-Funktionen benutzt, um die Tastatureingaben auf den Tastencode der Funktionstaste F7 zu überprüfen und im Erfolgsfall an die aktive Anwendung (Eingabefeld mit dem Focus) einen definierten String zu senden.

4.2.2 Test auf Netzwerkprotokoll-Ebene

Bei der Testautomatisierung auf Protokoll-Ebene zeichnet ein Tool alle Aktionen (Senden und Empfangen) an der jeweiligen Schnittstelle auf. Dazu werden die Dateien, durch welche die aufzuzeichnenden Protokolle implementiert sind, durch Andere ausgetauscht, die dem Tool die Möglichkeit des Aufzeichnens und Wiedergebens von Aktionen geben (winsock; capi etc.). [siehe 6.1 Abschnitt LoadRunner]

5 Der Status Quo des Testens in Deutschland

In [Taube] wird der derzeitige Stand der Prüf- und Testprozesse in der Softwareentwicklung dargestellt. Die Ergebnisse sind einer empirischen Studie der Universität zu Köln entnommen. Die Befragung von Softwarehäusern, Entwicklungsabteilungen von Banken und Versicherungen wurde 1997 durchgeführt.

Im Durchschnitt standen in den befragten Unternehmen 21 Entwicklern rund ein Qualitätssicherer gegenüber. Zum Vergleich: 1993 lag das Verhältnis bei 35:1 oder bei Microsoft ist das Verhältnis im Jahr 1997 1:1. [ct19/98]

Annähernd die Hälfte der Unternehmen sind nach ISO 9000 zertifiziert und regeln ihre Prüf- und Testprozesse durch Vorschriften und Richtlinien.

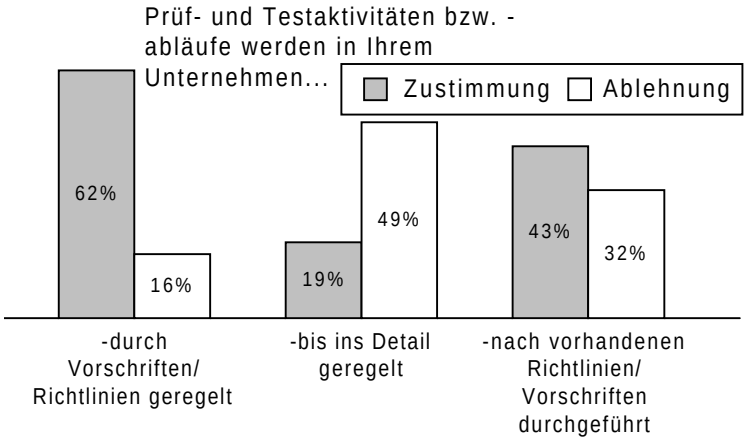


Abbildung 4.2-1: Prüf- und Testaktivitäten [Taube]

Ein Drittel der Unternehmen, die Standards einsetzen, gehen nicht nach den in [Kapitel 7] dargestellten Vorschriften vor, wobei 21 % der zertifizierten Unternehmen diese Standards nicht anwenden. Der Studie nach liegt dies an den sich unterscheidenden Softwareprojekten und der damit verbundenen hohen Heterogenität.

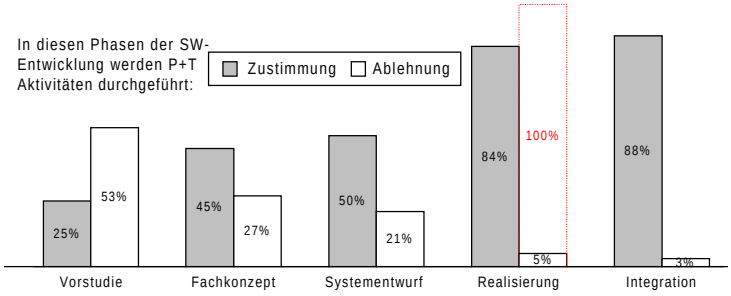


Abbildung 4.2-2: Entwicklungsphasen und P+T Aktivitäten [Taube]

Frühzeitige QS-Maßnahmen können die Fehlerquote in Programmen deutlich reduzieren. Ungefähr ein Viertel aller Unternehmen führen trotzdem in den Phasen der Fachkonzept-

bzw. Systementwurferstellung keine P+T-Aktivitäten durch. 16% der Unternehmen gaben an, in der Realisierungsphase gar nicht oder nur teilweise zu testen. Fast jedes dritte Unternehmen verwendet weder die Anforderungsdokumentation noch die Funktional-, Architektur- oder Design-Spezifikation als Prüfobjekt.

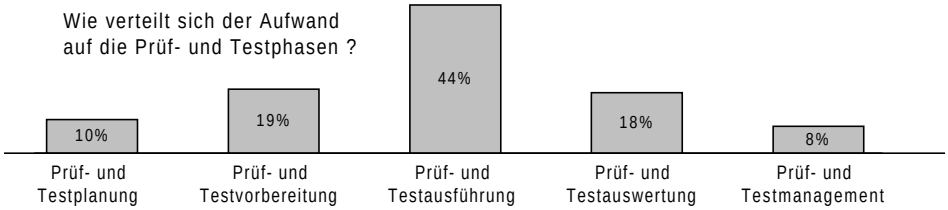


Abbildung 4.2-3: Aufwandsverteilung [Taube]

Nur zirka die Hälfte der Unternehmen führt eine explizite Aufwandsplanung für die P+T-Aktivitäten durch und mißt die entstandenen Aufwände. Die tatsächlichen Aufwände für QS-Maßnahmen liegen im Mittel bei 25,5% des Gesamtaufwandes für die Softwareerstellung (Forderung in der Fachliteratur: 40-50%) [Myers].

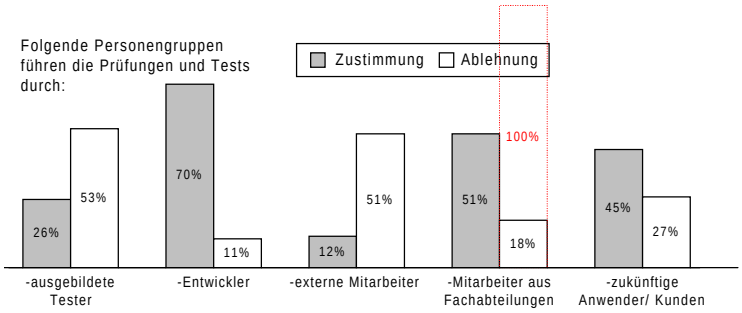


Abbildung 4.2-4: Personengruppen für die Testdurchführung [Taube]

Nur in einem Drittel der Unternehmen existiert eine selbständige Testabteilung. In den wenigsten Unternehmen werden ausgebildete Tester eingesetzt. In 35% der befragten Unternehmen testen Softwareentwickler die von ihnen selbst entwickelten Programme. Die weiteren Personengruppen der Testdurchführung sind Abbildung 4.2-4 zu entnehmen.

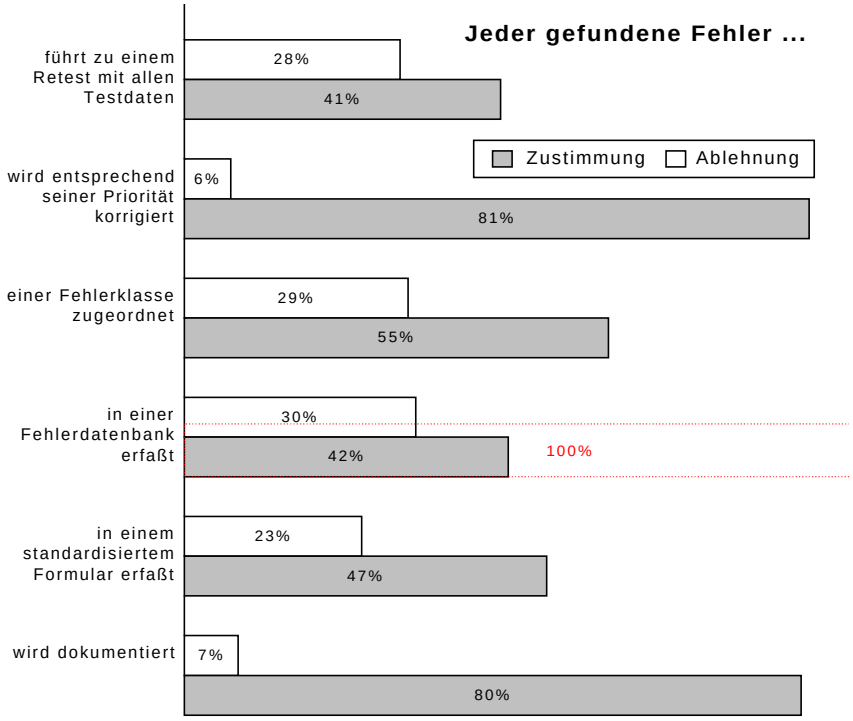


Abbildung 4.2-5: Fehlerhandling [Taube]

Das Fehlerhandling ist in den Unternehmen relativ gut ausgeprägt (vgl. Abb. 4.2-5). Nach einer Fehlerbehebung verwenden nur 41 % der Unternehmen alle Testdaten zu einem Retest. Die Unternehmen wurden befragt, wie sie die Testfallerstellung durchführen. Nur etwa ein Drittel aller Befragten unterstützen die Testfallerstellung durch Methoden und Techniken nach Abschnitt 3.3.4 sowie eine Testfall-Datenbank. Ähnliche Ergebnisse treten bei der Analyse der Aktivität Testdatenerstellung auf. 16 % der Unternehmen legen keine erwarteten Ergebnisse vor dem Test fest und in 22% der Fälle stehen diese nicht mehr für spätere Tests zur Verfügung. Einen Vergleich der Soll- mit den Ist-Ergebnissen führen nur 77% der Unternehmen regelmäßig durch.

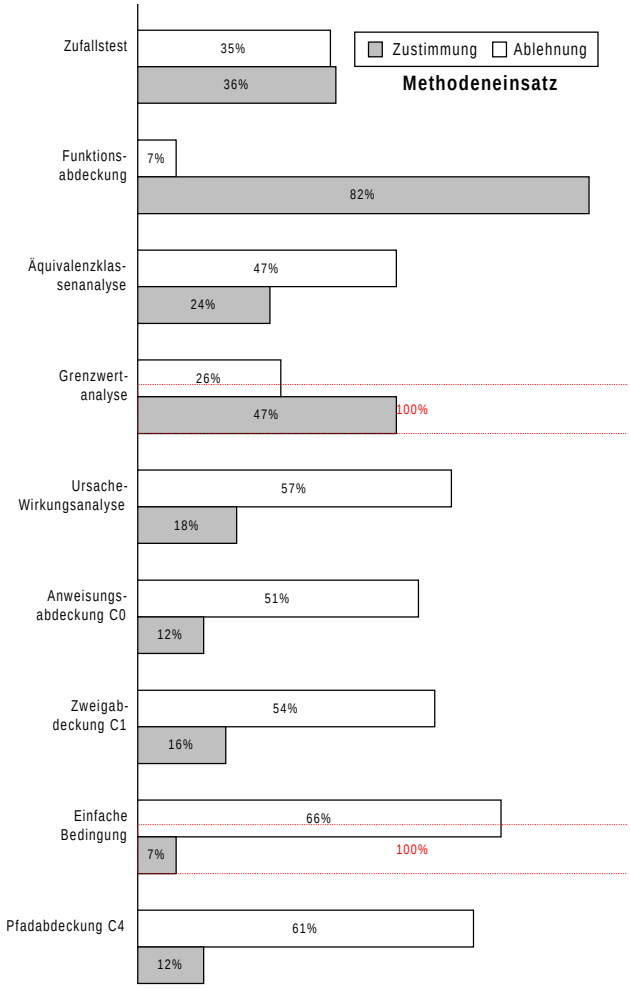


Abbildung 4.2-6: Methodeneinsatz [Taube]

Im Rahmen der statischen Analyse werden in 49 % der Unternehmen Dokumenten-Reviews durchgeführt, Code-Inspektionen oder Walkthroughs finden in weniger als einem Viertel der Unternehmen Verwendung.

Bei der funktionsorientierten Testfallermittlung hat die Methode der Funktionsabdeckung die größte Verbreitung. Die Grenzwertanalyse wenden nur 47% aller Befragten an. Die weiteren in [Abbildung 4.2-6:] angeführten funktionsorientierten Methoden haben noch geringere Verbreitung. Die strukturorientierten Methoden (sog. White-Box-Tests) werden derzeit kaum zum Messen von Überdeckungsmaßen verwendet. Hierbei spielt die Zweigabdeckung mit 16% die größte Rolle. Insgesamt ist die Anwendung der strukturierten und methodenbasierten Testfallerstellung nicht befriedigend.

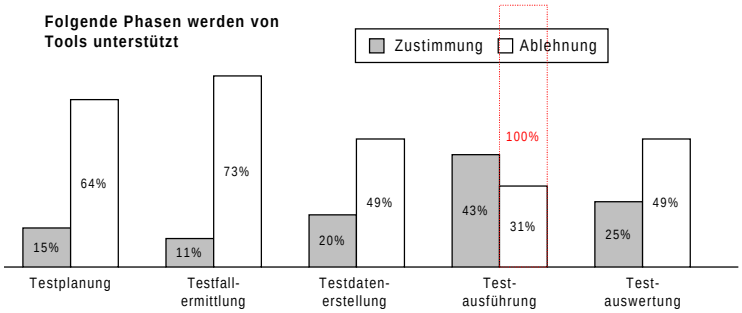


Abbildung 4.2-7: Werkzeugunterstützte Phasen [Taube]

43 % der Befragten setzen während der Testdurchführung Werkzeuge ein. Die vorgelagerten Phasen (Testfall- und Testdatenerstellung) sowie die Testauswertung werden nur von wenigen Unternehmen werkzeugunterstützt durchgeführt.

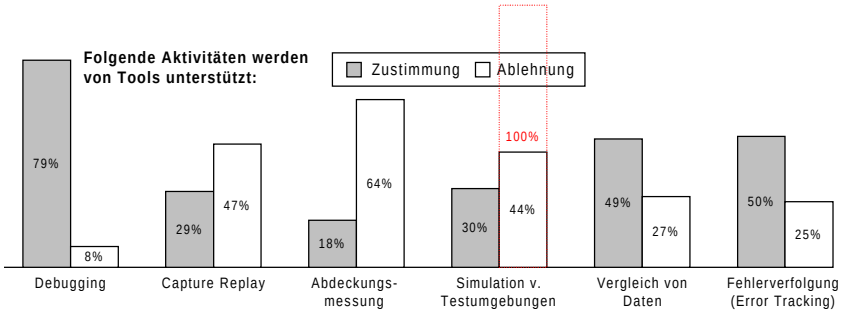


Abbildung 4.2-8: Werkzeugunterstützte Tätigkeiten [Taube]

Annähernd die Hälfte der Unternehmen automatisieren den Datenvergleich sowie die Fehlerverfolgung. Die Capture / Replay-Technik wird nur von knapp einem Drittel der Befragten werkzeuggestützt durchgeführt.

Entsprechend den genannten Änderungspotentialen liegen die Hauptanstrengungen der Unternehmen in der Verbesserung der Fehlerfindungsquote (37 %). Erst dann folgen Kostensenkung (19 %) sowie Erhöhung der Automatisierung (17 %) und der Standardisierung (12 %). Zu den am häufigsten genannten positiven Erfahrungen gehören Zeitersparnis durch Automatisierung und selbständige QS-Abteilungen. Negative Erfahrungen wurden vor allem durch zu geringes Engagement des Managements für QS-Maßnahmen, durch hohen Erstaufwand bei automatisierten Tests sowie die Nichtanwendung von Methoden zur Testfallermittlung gemacht.
Die Unternehmen arbeiten verstärkt an einem Ausbau der Prüf- und Testaktivitäten mittels einer durchgängigen Unterstützung des Entwicklungsprozesses durch Tools und die Automatisierung der Test sowie einer erhöhten Schulung und Anwendung von Methoden und Techniken zur Testfallerstellung durch qualifizierte Tester.

Trotz der kritischen Anmerkungen wurde insgesamt festgestellt, daß die Akzeptanz und die Qualität der Prüf- und Testprozesse in der Softwareentwicklung gewachsen sind und bei einer Wiederholung der Umfrage bessere Ergebnisse wahrscheinlich sind. [Taube]

6 Testen im Intranet-Testbed

Gemessen an der in der Studie aufgezeigten Situation bei den Mitbewerbern erfolgt das Testen im " Intranet Testbed " der Deutschen Telekom Berkom GmbH auf einem vergleichsweise hohen Stand. Die Berkom legt bei den durch sie auszuführenden Tests vor allem Wert auf (vgl. Abb.: 4.2-1 bis 4.2-8):

- frühzeitige Einbindung von QS-Maßnahmen in den Softwareentwicklungsprozeß,
- selbständige Rechnungseinheit, betriebswirtschaftlich denkend,
- acht von der Entwicklung unabhängige Tester in einer Fachabteilung,
- jeder Fehler wird kategorisiert und führt zu einem Retest,
- jeder Fehler wird dokumentiert, verfolgt und in einer Datenbank erfaßt,
- Einsatz von strukturierten und methodenbasierten Techniken zur Testfall- und Testdatenerstellung [siehe 3.],
- Formulierung der Testergebnisse,
- Kategorisierung der Testobjekte und Test nach Kritikalität und Komplexität,
  Kategorie A – Testfallermittlung nach wissenschaftlichen Methoden (Ursache-Wirkungsgraph und/oder Äquivalenzklassenbildung);
  Kategorie B – Funktionsabdeckung, Testdaten;
  Kategorie C – AdHoc Testfallermittlung ohne Testdaten;
- automatischer (toolunterstützter) Soll- / Istwert-Vergleich mit Testauswertung gegen die formulierten Ergebnisse,
- toolunterstütztes Capture / Replay und Defect Tracking.

Nachteilig stellt sich für das Testbed der Berkom heraus:

- Da das Intranet Testbed innerhalb der Berkom eine betriebswirtschaftliche Einheit, unabhängig von den Entwicklungsabteilungen ist, welche auch kostendeckend arbeiten sollen, ist die Entscheidung zur Nutzung der Testkapazitäten keine Konzern- oder Managemententscheidung, sondern abhängig von der Verantwortung des Projektleiters.
- Die ISO 9000 Zertifizierung ist erst für September 1999 geplant.
- Bei der Entwicklung von Prototypen ist die Führung und Pflege der Dokumentation / des Pflichtenheftes unzureichend, daher können diese häufig nicht als Prüfobjekte eingesetzt werden, gegen die getestet wird.
- Es erfolgt keine werkzeugunterstützte Testfall- oder Testdatenerstellung.

6.1 Beschreibung der Testaktivitäten

Aktivitäten	**Beschreibung**
Erteilung eines Testauftrages	Kontaktaufnahme des Kunden/ Auftraggebers mit dem Testbed
Teststufen festlegen/vereinbaren	Die Teststufen richten sich nach den aktuellen Entwicklungsphasen der zu testenden Software (in Zusammenarbeit mit dem Auftraggeber)
Testorganisation	Zeit- u. Ressourcenplanung (in Abstimmung mit dem Auftraggeber)
Aufnahme der Anforderungen an die Software	Einarbeiten in die notwendigen Dokumente: Beschreibung der Software, Aufträge an die Softwareerstellung als Sollvorgaben für die Funktionalität und Qualität der Software
Festlegen der Testendekriterien / Testabdeckung	Festlegen wie lange getestet wird, wie umfassend die Tests sein müssen, wie fehleranfällig oder entscheidend ist das zu testende Element im Verhältnis zum nötigen Aufwand (in Zusammenarbeit mit dem Auftraggeber)
Testfallermittlung	Festlegen, was im Einzelnen getestet werden soll (Szenario entwerfen, Checklisten erstellen, Daten definieren, Testumgebung festlegen)
Testumgebung Anpassen	Anpassen der Hard- u. Softwaregrundlage, Einrichten von Testwerkzeugen und Datenbankstrukturen
Testdaten Erstellen	Erzeugen und Einspeisen der Daten mit denen getestet werden soll
Durchführung der Tests	Abarbeiten der Checkliste für diesen Tests, Notieren der Ausgaben und Zustände des Systems
Testauswertung	Auswertung der Abweichungen (Soll/Ist), Zusammenfassung und Aufbereitung der Ergebnisse

Tabelle 6.1-1: Beschreibung der Testaktivitäten im ITB

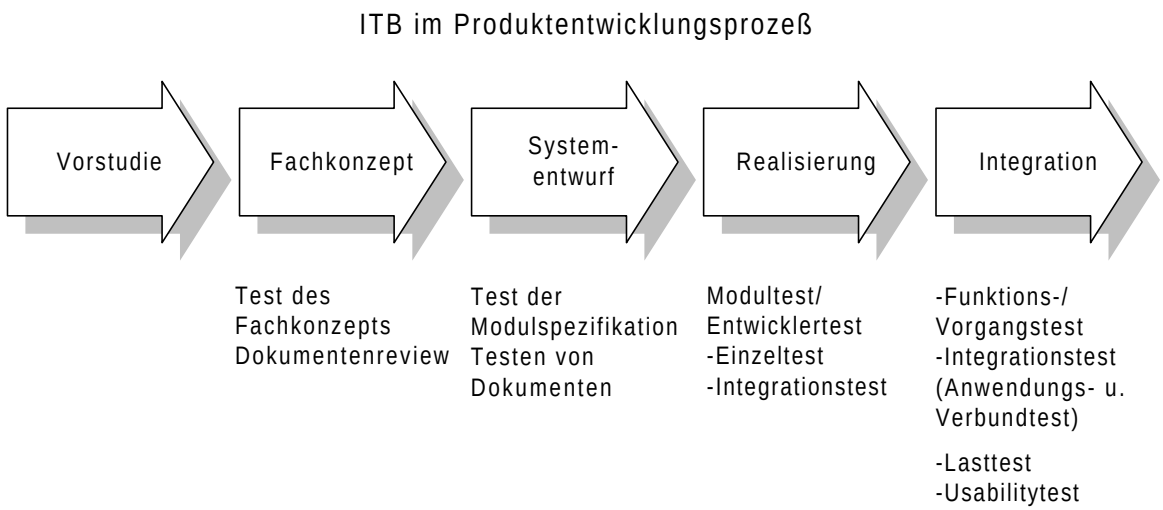


Abbildung 6.1-1: ITB im Produktentwicklungsprozeß [Vett]

Der Performance-Test

Die Lasterzeugung erfolgt mit Hilfe des Tools LoadRunner 5.0 der Firma Mercury. Es handelt sich hierbei um ein Tool, das auf der Clientmaschine (Windows NT oder Unix) installiert wird und die Möglichkeit bietet auf der Anwendungs- bzw. Kommunikationsprotokollebene (Winsocket; R3; etc.) alle Transaktionen, die zwischen dem echten Client und dem Server ausgetauscht werden, mitzuschneiden. Das Ergebnis dieser Mitschnitte sind Scripte, ähnlich der Programmiersprache ANSI-C. Die Scripte werden später parametrisiert, um damit die mitgeschnittenen Transaktionen mit einer beliebigen Zahl virtueller User ohne echtem Client durchzuführen. Zur Ausführung kommen die Scripte über den integrierten C-Interpreter. Nach Kombination der Scripts zu Szenarien und nach der Festlegung der Iterationen und Anzahl der virtuellen Benutzer kann die Testausführung durch Verteilung der Last auf mehrere Rechner erfolgen. Danach werden die Meßdaten von den beteiligten Maschinen eingesammelt und zu verschiedenen Reports (Meßprotokolle und Grafiken) umgewandelt. Es sind die Anzahl der ausgeführten Transaktionen und deren Zeiten, aber auch Serverressourcenverhalten und Netzwerkkomponentenverhalten im zeitlichen Bezug zu den Szenarien auswertbar. Die Kombination der Scripts zu Szenarien, die Testausführung und Ergebnisprotokollierung ist über den Loadrunner-Controller möglich.

Prinzipiell sind mit dem Tool Messungen der folgenden Art denkbar:

- Messungen um Serverengpässe aufzudecken,
- Messungen von Antwortzeiten der Transaktionen,
- Messung der Einflüsse von Netzwerkkomponenten.

Das Reporting der Meßergebnisse erfolgt in die Datenbank des TestDirectors.

Der LoadRunner, als Testequipment für die Protokoll-Ebene, bietet eine Vielzahl von Prozeßmonitoring-Funktionen z.B. Online Performance Monitor, Transaction Monitor, Server Monitor, Network Delay Monitor, SNMP Monitor, TUXEDO Monitor, Transaction

Breakdown Monitor. Weiterhin unterstützt er eine Reihe von Anwendungen / Umgebungen einschließlich:

- E-Business (JDBC, LDAP, CORBA, POP3, FTP, HTTP, SSL, Digital Certificates, NTLM)
- ERP (SAP, Oracle Applications, PeopleSoft, Baan)
- n-tier (Jolt, TUXEDO, DCOM)
- 2-tier (Oracle, Sybase, MS SQL Server, Informix, ODBC, Winsock)
- Legacy (3270, 5250, VT100-520, APPC (AS/400), X-Window).

[Bericht_Lasttest_Comenius_11f.rtf]

Die folgenden Ressourcen sollten an beiden Endpunkten der Messung (auch prozeß-bezogen) gemonitort werden:

- Hauptspeicherverbrauch / Virtueller Adressraum
- Systemlast / CPU-Zeit
- Sonstige Systemressourcen (z.B. Sockets, Ports, Filedeskriptoren, Prozeßanzahl, ...)

Lasttestvarianten:
Um das Verhalten der AuT (Application under Test) in den unterschiedlichsten Lastsituationen beurteilen zu können, können verschiedene Varianten der Lasttestdurchführung, die die Nutzerzahlen und die Testinfrastruktur variieren, durchgeführt werden. Im Folgenden sind einige näher betrachtet.

Variation der Nutzerzahlen
Im derzeitigen Ausbaustand ist es im Testbed möglich bis zu 500 parallele Nutzer zu simulieren. Der LoadRunner gestattet die Verteilung der parallelen Zugriffe in Szenarien. Es sind unter anderem die folgenden Verteilungen möglich:

- Lineare Verteilung der Nutzerzahlen
  Die Nutzerzahlen werden beginnend mit einem Nutzer in Einer-Schritten bis auf 500 Nutzer gesteigert.
- Logarithmische Verteilung der Nutzerzahlen
  Die Nutzerzahlen werden in Intervallen gesteigert (z.B. 10, 25, 50, 100, 250, 500).
- Stochastische Verteilungen
  Es wird von bis zu 500 Nutzern ausgegangen, wobei folgendes Verfahren gilt:
  - Ab einer Startzeit t_0 benutzen die Nutzer das System, wobei jeder Nutzer eine zufallsabhängige individuelle StartZeit t_0+dt besitzt (dt gleichverteilt in [o .. dt_{Max}])
  - Jede Sitzung besteht aus Teilszenarien von denen jedes eine zufallsgesteuerte Denkzeit (Gaußverteilung in [t_{Min} ..t_{Max}]) des Nutzers beinhaltet und eine zu protokollierende Reaktionszeit des Systems.
  - Die zeitliche Dauer T jeder Sitzung ist somit ebenfalls zufallsabhängig.

- Nach jeder Sitzung wird nach einer Pause dt, deren Länge zufallsabhängig ist, eine neue Sitzung gestartet.
- Das Systemverhalten wird über eine längere Zeit aufgezeichnet.

Eine Variation des Verhältnisses zwischen dt und der durchschnittlichen Dauer einer Sitzung T steuert die durchschnittliche Anzahl paralleler Dienstsitzungen (dt << T ... viele parallele Nutzer).

Variation der Infrastruktur

Der Infrastrukturtest nutzt die Möglichkeiten, die Testinfrastruktur zwischen den einzelnen Durchgängen des Lasttests zu verändern und damit Rückschlüsse auf die Perfomance der zugrundeliegenden Hardwarekomponenten zu erhalten. Leicht veränderbare Parameter des Infrastrukturtests sind der Prozessortyp (CISC- oder RISC-Prozessoren) oder deren Anzahl sowie die Größe des Hauptspeichers des Systems.

Dauerlast

Bei dem Dauerlasttest wird das Systems im Dauerbetrieb mit definierten Szenarien und Iterationen belastet. Ziel des Dauertest ist es, die Stabilität unter lange andauernden Lastsituationen beurteilen zu können.

[Lasttest NNK-PT-EAM.doc]

Generelle Vorgehensweise zur Performance-Testdurchführung:

- Identifikation der Lastbausteine, Denkzeiten und zu messende Transaktionen als Basis zur Erstellung der LoadRunner-Testscripts,
- Mitschnitt der LoadRunner-Testscripts und gleichzeitiges Einfügen der Denkzeiten und Transaktionsmeßpunkte,
- Testdatendefinition - Festlegung der Bestandsdaten (Sekundärdaten) und der während der Laufzeit der Tests benutzten Daten (Primärdaten),
- Anpassung und Parametrisierung der Scripts anhand der Entwicklerangaben und der Testdatendefinitionen,
- Test der einwandfreien Funktion der Scripte,
- Erstellung der Testausführungsszenarios, Dies bedeutet die Verteilung der Gesamtlast auf verschiedene Nutzerszenarien anhand des vom Auftraggeber ermittelten Nutzerverhaltens und der zu erwartenden Last während des zukünftigen Betriebes der Anwendung. Es wird ebenfalls der Zeitraum bzw. die Anzahl der Wiederholungen der Transaktionen und deren Durchführungsszeitpunkte definiert.
- Testausführung der Testausführungsszenarios und Verteilung der Last auf Clientsysteme,
- Protokollierung der Testergebnisse gemäß Auftraggeberwunsch,
- Reporting der Testergebnisse im Defect-Tracking-System,

Der Winrunner der Firma Mercury wird im "Intranet Testbed " (ITB) zur Ende-zu-Ende-Messung der GUI-Aktivitäten verwendet. Er schneidet die Aktionen der Windows-API-Objekte der Clientanwendung mit und erzeugt parametrisierbare Scripte. In Zusammenarbeit

mit dem Loadrunner können Transaktionszeiten der Winrunneraktivitäten gemessen werden. [Bericht_Lasttest_Comenius_11f.rtf]

6.2 Methodische Anforderungen

Aufgrund der herausragenden Bedeutung der Testfallermittlung für die Qualität des Tests beziehen sich die methodischen Anforderungen an einen effizienten Test im Wesentlichen auf die Ermittlung der relevanten Testfälle. Die wichtigsten Aspekte sind hierbei:

- die Leistungsfähigkeit der Methode in bezug auf die Möglichkeiten zur Fehleraufdeckung,
- die Systematik der Vorgehensweise,
- die an den praktischen Einsatz der Methode geknüpften Bedingungen,
- der mit dem Einsatz verbundene Aufwand bzw. die Skalierbarkeit des Aufwands,
- die Handhabbarkeit und die Erlernbarkeit der Methode,
- die Möglichkeiten, den beim Test erzielten Testumfang quantitativ zu beurteilen ("Meßbarkeit des Testumfangs") sowie
- die Verfügbarkeit von Rechnerunterstützung bzw. der Grad der Automatisierung der Methode.

Maßgeblich für die Klassifizierung der Testmethoden sind die Informationsquelle und die Kriterien, die zur Ermittlung der Testfälle herangezogen werden. Bei den Funktionstests werden die TF aus der funktionalen Spezifikation des TO mit dem Ziel abgeleitet, möglichst viele Funktionen des TO zu testen (versch. Kriterien der TF-Ermittlung in 4.1.1).
Bei den Strukturtests werden die TF aus der Struktur des TO abgeleitet. Die wichtigsten Ansätze gehen vom Kontroll- bzw. vom Datenfluß des TO aus. Die Strukturtests werden auch "White-Box-Tests" genannt. Bei den statistischen Tests werden Testdaten rein zufällig oder unter Annahme bestimmter Wahrscheinlichkeitsverteilungen für die Werte der Eingangsparameter innerhalb des Eingabedatenraums, ausgewählt.

6.2.1 Testplanung

Grundsätzlich ist jede fachliche Funktion ein Testobjekt. Die Testobjekte sind nach dem Lasten-, Pflichtenheft und der Spezifikation (Use-Case) katalogisiert und in die Qualitätskategorien A – C eingeteilt. Die Kategorisierung ist nach Sichten ausgerichtet, beispielsweise nach den Aspekten: Sicht der Users oder des Administrators und deren ausführbaren Aktionen/Funktionalitäten, der Abrechnung, der Datensicherheit und dem Daten- oder Systemschutz.

Die Qualitätskategorien werden differenziert:
Zur Aufwandsabschätzung werden zwei Faktoren berücksichtigt. Ein Faktor ist die Kritikalität des Testobjekts (Kr). Sie charakterisiert, wie kritisch die Fachseite das Testobjekt bewertet.

Kategorie A: direkter Einfluß auf Abrechnung und/oder auf Datenkonsistenz durch fachliche Funktionen
(sinnvolle Weiterverarbeitung nicht möglich)

Kategorie B: indirekter Einfluß auf Abrechnung und/oder auf Datenkonsistenz durch fachliche Funktionen
(sinnvolle Weiterverarbeitung nur bedingt möglich)

Kategorie C: kein Einfluß auf Abrechnung und/oder auf Datenkonsistenz durch fachliche Funktionen
(Weiterverarbeitung möglich)

Anhand der **Kategorie** wird die Testtiefe des jeweiligen Testobjektes bestimmt (Methode ITB-Berlin).
Den Qualitätskriterien/kategorien werden Testverfahren zugeordnet.

Kategorie A: Vollständige Testfallermittlung unter Berücksichtigung aller Eingabeelemente und -klassen der fachlichen Funktion und aller möglichen Wirkungen (Systemmeldungen, Fehlermeldungen, Ausgaben u.a.), Testdatendefinition für jeden Testfall und vollständige Testprozedurerstellung (Ziel ist die annähernd vollständige Testabdeckung und schnelle Reproduzierbarkeit im Regressionstest in der nächsten Version)

Kategorie B: AdHoc Testfallermittlung unter Berücksichtigung der kritischen Eingabeelemente und -klassen der fachlichen Funktion und Wirkungen, Testdatendefinition für jeden Testfall und vollständige Testprozedurerstellung (Ziel ist eine 50%ige Testabdeckung und schnelle Reproduzierbarkeit im Regressionstest in der nächsten Version)

Kategorie C: keine Testfallermittlung, aber direkte Testausführung unter Berücksichtigung der kritischen Eingabeelemente und -klassen der fachlichen Funktion und Wirkungen, **keine** Testdatendefinition, aber Erstellung einer Checkliste mit Testfällen während der ersten Testausführung (Ziel ist eine schnelle Testausführung)

Daraus ergibt sich folgender Testaufwand:

A = 3 Tage ; B = 2 Tage ; C = 1 Tage

Der zweite Faktor ist die **Komplexität** (Ko) des Testobjekts. Sie charakterisiert, wie umfangreich die Bearbeitung des Testobjekt zu bewerten ist. Kriterien sind die Anzahl der Masken, der Felder, der Elementklassen, deren Abhängigkeiten untereinander und Wirkungen. Besonders großen Einfluß auf die Komplexität haben der Aufwand der Testdatenorganisation und des Testdatenmanagements für die Sicherung, Bereinigung und Wiederherstellung definierter Datenzustände für die Wiederholbarkeit der Tests.

Komplexität I: Einem TO wird die Ko I zugeordnet, wenn es eine oder mehrere fachliche Funktionen und Eingabemasken mit insgesamt mehr als 10 Eingabeelementen und mindestens zwei Elementklassen je Eingabeelement und Abhängigkeiten zwischen den Eingabeelementen enthält. Für TO der Ko I ist eine Testdatenorganisation erforderlich. Das Testdaten-management ist aufwendig, da manuelle Anpassungen notwendig sind.

Komplexität II: Eine fachliche Funktion und eine Eingabemaske mit insgesamt mehr als 5 Eingabeelementen und mindestens zwei Elementklassen je Eingabeelement kennzeichnen ein TO der Ko II. Das Testdatenmanagement ist weniger aufwendig, da es mit Systemfunktionen der Anwendung durchgeführt werden kann aber eine Organisation der Testdaten ist erforderlich.

Komplexität III: Die Ko III wird einem TO zugeordnet, welches eine fachliche Funktion und eine Eingabemaske mit insgesamt weniger als 5 Eingabeelementen besitzt. Für diese TO ist keine Testdatenorganisation erforderlich. Das Testdatenmanagement ist weniger aufwendig, da es mit Systemfunktionen der Anwendung durchgeführt werden kann.

Daraus ergibt sich folgender Zusatzaufwand:

I = 2 Tage ; II = 1 Tage ; III = 0 Tage

Der Gesamtaufwand des Testobjektes ergibt aus der Summe seines Kritikalitätsfaktors und seines Komplexitätsfaktors.

Kr + Ko = Gesamtaufwand des Testobjektes

Definition der Fehlerklassen

Fehler der Klasse 1 (schwere Fehler):

- Fehler, die eine sinnvolle Weiterverarbeitung verhindern
- Fehler, die finanzielle Folgen haben (z.B.: falsche Ermittlung von Rechnungsbeträgen)

- Fehler mit Außenwirkung (z.B.: falsche Informationen zur Abrechnung an Kunden)

Fehler der Klasse 2 (mittlere Fehler):

- Fehler die eine sinnvolle Weiterverarbeitung nur mit erheblichen Aufwand ermöglichen (z.B.: Umgehungslösungen durch manuelle Datenmanipulation)
- sonstige fachlich falsche Ergebnisse
- falsche oder fehlende Hilfetexte

Fehler der Klasse 3 (leichte Fehler):

- Fehler die eine sinnvolle Weiterverarbeitung nicht behindern
- Oberflächenfehler (z.B.: falsche Buttons / Texte)

Innerhalb der Testplanung sind Annahme- und Abnahmekriterien zu definieren. Zur Empfehlung einer Freigabe darf die Anwendung 0 Fehler der Klasse 1; max. 3 Fehler der Klasse 2 und max. 10 Fehler der Klasse 3 enthalten (ITB). Im Folgenden sind für eine spezielle Anwendung (Office im Netz) die Tests feinspezifiziert.

Qualtity-Merkmale	**Quality-Teilmerkmale**
Security	***Sicherheitsaspekte;*** bezüglich Übertragung persönlicher Daten, Systemangreifbarkeit, datenschutzrechtliche Bestimmungen
Functionality	***Feature set;*** SW-Funktionalitätenkatalog laut Pflichtenheft, Funktion an Telekommunikationsnetz (ISDN, analog, TK-Anlage)
Usability	***Human factors***; Mensch-Maschine-Schnittstelle
	Ästhetics*;*
	Consistency; Schriften, Farben, Gestaltung der Elemente
	Documentity; Benutzer-Documentation, Inhalt, Gestaltung, Umfang
	Efficiency; Anzahl der Aktionen pro gewünschtem Ziel
Reliabilty	***Recoverability;*** nach Absturz, Komponentenausfall, Datenbankcrash
	Accuracy; Rechengenauigkeit, Währung, Zeit, Menge; Datenübertragung Menge, Inhalt
Performance	***Speed;*** Rechen-, Datenzugriffs-, Übertragungsgeschwindigkeit
	Resource consumtion; RAM, FP, CPU-Auslastung, Datenbank
	Response time; Seitenaufbau, Systeminaktivität
	Masse; Größe der Datenbank, Art und Umfang der Dateiablage, Art der Dateien (avi, doc, mpeg, tiff ...)
	Last; Anzahl gleichzeitiger Zugriffe
	Stress; Systemüberlastung, unzulassige Aktionen

Qualtity-Merkmale	**Quality-Teilmerkmale**
Supportability	***Service-Unterlagen;*** Inhalt, Konsistenz zu Benutzer-, Admin-Handbüchern
	Extensibilit; Skalierbarkeit, Erweiterbarkeit
	Maintainability; bei Teil-Komponentenausfall
	Compatibility; zu Vorgängerversionen
	Configurability; (Sprachen, Betriebssysteme, Basis-SW)
	Serviceability; Treiber-Update
	Installabiliy; Install-Routine (Handhabbarkeit der Installation und Erfolg nach Anweisung)
	Localizability; Zeichensätze, Währung, Tel-Netz, TK-Anlagen, Netzwerk,
Eurotest	***Umrechengenauigkeit;*** auf 3 Stellen genau, Zeichensatz
Y2000-Test	HW- und Bios, SW
Dokumentenprüfung	***Zulassungen;*** (el. Sicherheit, EMV, Telekommunikation)
	SW-Entwicklerdokumentation (Programmablaufplan, Architekturplan, Testplan und Ergebnisse) HW-Dokumentation

Tabelle 6.2-1: Feinspezifikation ausgewählter Tests [Tintern]

6.2.2 Funktionstest im ITB

Abhängig von der Art der vorhandenen Spezifikation (streng formal, Datenflußdiagramm, Textform) wird das TO entsprechend der methodischen Testfallermittlung dem Funktionstest unterzogen.

Funktionstest nach den Methoden der TF-Ermittlung:

- Beim Äquivalenzklassentest wird der Eingabedatenraum des TO auf der Basis der funktionalen Spezifikation in eine endliche Anzahl von disjunkten Wertemengen (Äquivalenzklassen) eingeteilt, so daß man annehmen kann, daß der Test mit einem Repräsentanten aus jeder Klasse einen Test mit allen anderen Werten dieser Klasse erübrigt.
- Basis des Grenzwerttests ist eine Äquivalenzklasseneinteilung sowohl des Eingabe- als auch des Ausgabedatenraums des TO. Ziel des Grenzwerttests ist es, Fehler aufzudecken, die bei Daten in den Grenzbereichen dieser Klassen auftreten.
- Ziel des Ursache-Wirkungsgraph-Tests (Cause Effect Graphing) ist es, die Funktionen des TO in allen möglichen Kombinationen zu testen. Hierzu werden Ursache und Wirkungen

abgeleitet und spezifiziert, wobei eine Ursache eine bestimmte Eingangsbedingung des TO und eine Wirkung eine Nachbedingung ist, die den Wert der Ausgangsgrößen des TO nach Ausführung mit bestimmten Eingabedaten beschreibt. Die logischen Beziehungen dieser Ursachen und Wirkungen werden in einem Booleschen Graphen spezifiziert. Zur Beherrschung der durch Kombinatorik schnell zunehmenden Unübersichtlichkeit werden Zwischenknoten eingeführt, die bestimmte logische Kombinationen von Ursachen zusammenfassen. Die Testfälle werden durch Umsetzung des Ursache-Wirkungs-Graphen in eine Entscheidungstabelle ermittelt.

- Die Category-Partition Method geht von einer Kombination unterschiedlicher Ursachen aus. Gegenüber des Ursache-Wirkungs-Graph-Tests werden jedoch keine erwarteten Wirkungen definiert. Damit stellt diese Testmethode prinzipiell keine Unterstützung für die Testauswertung dar.

6.3 Anforderungen in Bezug auf die Testautomatisation

Obwohl die TF-Ermittlung die wichtigste Aktivität im Rahmen eines Tests ist, bedarf es für einen effizienten Test in der Praxis auch für alle anderen beschriebenen Testaktivitäten einer weitreichenden Rechnerunterstützung. Durch ein dem Testwerkzeug zugrundeliegendes Vorgehensmodell, eine leistungsfähige Datenhaltung und eine einheitliche Benutzeroberfläche müssen die Übergänge zwischen den verschiedenen Testaktivitäten wirkungsvoll unterstützt und die Datenbestände zu jedem Zeitpunkt des Tests konsistent gehalten werden. Das Testwerkzeug muß eine dem Vorgehensmodell entsprechende Steuerung besitzen und den Tester leiten ohne ihn einzuschränken.

Eine weitere Anforderung an die Testautomatisierung ergibt sich aus der Host-/Zielrechner-Problematik. Eine effiziente Lösung ist ein Testwerkzeug, mit dessen Hilfe die eigentliche Testdurchführung über eine Host-/Ziel-Rechner-Verbindung in der realen Einsatzumgebung ermöglicht wird, die aufwendigen vor- und nachbereitenden Aktivitäten aber auf dem Hostrechner durchgeführt werden können. Im ITB kommt der in [Kapitel 6.1] beschriebenen LoadRunner zum Einsatz.

7 Normen und Standards

Als Instrument zur Steuerung der Softwareentwicklung bezüglich Arbeitsteilung und Austauschbarkeit, Kommunikation zwischen den Fachabteilungen, und die Vereinheitlichung von Begriffen und Vorgehensweisen, welche sich auf Produkte, Prozesse, Verfahren und Ressourcen beziehen, haben Standards und Normen einen hohen Stellenwert.

7.1 Testen nach ISO 9000

Zur Zeit wird vor allem die ISO 9000 Normenfamilie mit ihren Anforderungen an ein Qualitätsmanagementsystem (QMS) für die (Software-)Entwicklung und –wartung diskutiert. Eine Besonderheit dieser Norm ist die Tatsache, daß sich die Unternehmen die Übereinstimmung ihres QMS mit der Norm von unabhängigen Zertifizierern bestätigen lassen können.

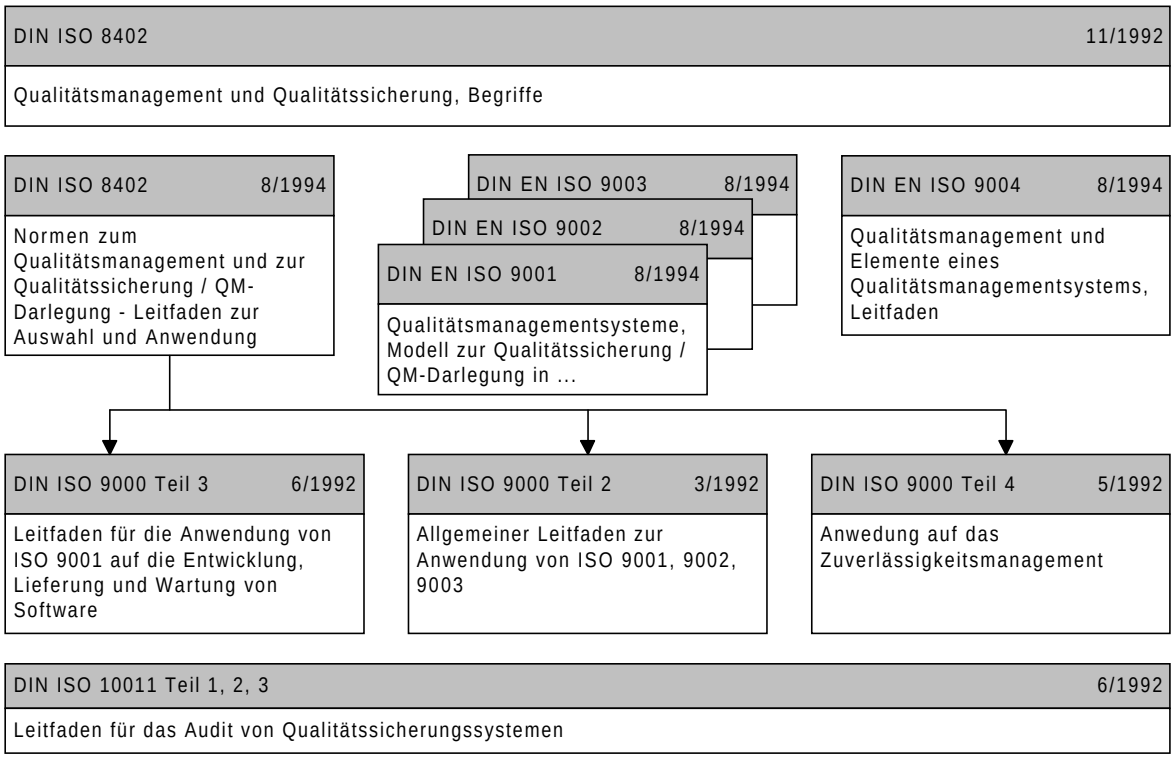


Abbildung 7.1-1: Inhalte der ISO 9000 bezüglich Testen

Die Norm DIN ISO 55350 definiert Qualität als die Gesamtheit von Eigenschaften und Merkmalen eines Produktes oder einer Tätigkeit, die sich auf deren Eignung zur Erfüllung gegebener Erfordernisse bezieht.

Qualitätsplanung beginnt danach bei der Auseinandersetzung mit den Kundenbedürfnissen (Kundenorientierung). Zur Qualitätsplanung gehört nach ISO 8402 (Begriffsnorm) auch die Planung aller spezifischen Qualitätspraktiken, Ressourcen und Vorgehensweisen. Die ISO 9000-3 fordert die Dokumentation dieser Planung in einem Qualitätsplan nach ISO/CD 9004-5. [ISO9000]

7.2 IEEE 730

Das IEEE (Institute of Electrical and Electronics Engineers) veröfflichte mit dem Standard 730 „IEEE-Standard für Softwarequalitätssicherungspläne“ eine Minimalanforderung an die zu erstellende Dokumentation und gibt die durchzuführenden Reviews und Audits vor. Weiterhin sind Festlegungen an das Management der Softwareentwicklung in dem Standard enthalten. IEEE entspricht dem Absatz „Designlenkung“ in ISO 9001.
[ISO9000]

7.3 CMM

Zur Standardisierung der Qualitätssicherung und des Prozeßmanagements bevorzugt man in den Vereinigten Staaten statt der ISO 9000 einen anderen Standard: CMM [Capability Maturity Model]. Diese Gruppe von Normen konzentriert sich ausschließlich auf den Lieferanten, nicht wie die ISO-Normen hauptsächlich auf die Kunden/Lieferanten-Vorgänge. ISO 9000 ff ist für diverse Unternehmen definiert - nicht primär für die Software-Industrie. Die zentrale Norm ISO 9001 über Qualitätsmanagementsysteme umfaßt fünf Seiten Text. ISO 9000 Teil 3, weitere elf Seiten, erklärt die Anwendung auf die Softwareentwicklung. Die 500 Seiten des CMM sind dagegen wesentlich konkreter und beschreiben den Entwicklungsprozeß im Detail.
Das Software Engenieering Institute (SEI) definierte für das amerikanische Verteidungungsministerium das CMM. Dieses inzwischen in Version 2.0 verfügbare Modell klassifiziert die Reife einer Softwareabteilung oder eines Software-Unternehmens als CMM-Level 1 bis 5.

Das unterste Level ist nicht standardisiert. Die Softwareentwicklung geschieht ohne Methodik. Der Designprozeß geht im Coding unter.

Das folgende CMM-Level 1 wird als 'initial' bezeichnet. Die Softwareentwicklung verläuft formlos und hängt von der Kompetenz weniger Personen ab. In der Designphase werden vor dem eigentlichen Programmierbeginn die Schnittstellen zwischen Softwaremodulen definiert. Es ist festgelegt, welche Informationen im Header-File und was im Quelltext stehen muß. Im CMM-Level 1 wird die theoretische Problemlösung und daran anschließende programmtechnische Lösung gefordert.

Level 2 des CMM wird als 'repeatable' bezeichnet. Dabei werden die Entwicklungsabläufe geregelt und sollen wiederholbar sein. Die Softwareentwicklung passiert nach den gängigen

Methoden von Analyse und Design. Es gibt ein allgemeines System für Projektmanagement und Projektüberwachung.

Im CMM Level 3 wird der Prozeß der Zusammenarbeit beschrieben. Er wird als 'defined' bezeichnet. Es gibt ein allgemeines System für Projektmanagement und Projektüberwachung. Alles ist festgelegt, das Programmieren wird industrialisiert, für jeden Schritt existieren Formulare und Protokolle. Getestet wird nicht mehr am fertigen Produkt, sondern permanent. Die Analyse, das Design, jedes Stück Code durchläuft definierte Testprozeduren. Die Tests erfolgen gegen die fachlichen Vorgaben im Pflichtenheft.

Das CMM-Level 4 wird als 'managed' bezeichnet. Der Prozeß der Softwareentwicklung ist stabil und bringt eine übereinstimmende Produktqualität. Messungen werden auch zur Überwachung des Entwicklungsprozesses verwendet. Die Frage nach den Ursachen von Abweichungen und die permanente Korrektur des Prozesses sind das wesentliche Merkmal der Stufe 4. Die Vorgänge während der Software-Produktentwicklung sind zerlegt in Analyse (das Problem begreifen), Design (Strategie zur Lösung erarbeiten), Kodierung und Test.

Die permanente Überprüfung dieser Phasen und ihrer Wechselwirkungen untereinander führt zu CMM-Level 5. Auch als 'optimizing' bezeichnet. Das 'Konstruktionsprinzip' des gesamten Software-Entwicklungsprozesses wird ständig untersucht und optimiert. Der Prozeß der Softwareentwicklung enthält seinen eigenen Verbesserungsprozeß.

Für den Mai 1998 meldet das SEI (Software Engineering Institute), das von knapp 3500 Projekten: 24% der untersuchten Organisationen Level 2 erreichten, Level 3 nur noch 15%, Level 4 erreichten 2% und Level 5 nur 0,6%.

Viele der Methoden aus der ISO finden sich im CMM verteilt über die Level 1 bis 3. Aus diesem Grund lassen sich ISO 9000 und das CMM im Prinzip parallel einsetzen und ergänzen einander.
[ct19/98] [ISO9000]

7.4 Qualitätsmanagement

Das Softwarequalitätsmanagement zerfällt in drei Teilaufgaben: Kontrolle der Qualität, Planung der Qualität und Steuerung der Qualität. Durch die prozeßorientierte Sicht des modernen Qualitätsmanagements kommt dem Prozeß und der Steuerung der Prozeßqualität eine höhere Bedeutung zu. In einem korrekt geplanten Entwicklungsprozeß gewinnt die

Zurückverfolgung von Fehlern durch Design und Fachkonzept an Bedeutung, um die Vollständigkeit der Korrektur aller Teilprodukte des Softwareprozesses sicherzustellen.

Eine zunehmende Bedeutung kommt damit der Analyse von Fehlern und Qualitätskosten (Vorbeugung, Prüfung, interne und externe Fehlerfolgekosten oder allgemeiner: Costs of Nonconformance) zu. Führende Softwarehersteller haben gezeigt, daß Kostensenkungen als Folge von Qualitätsverbesserungen (Fehlerfolgekosten momentan bis zu 10% des Umsatzes) möglich sind.

Die Implementierungsmethode der Software als strategische Entscheidung wird in Analogie zum Testen in Top-Down oder Bottom-Up-Implementierung unterschieden.

Die Top-Down-Implementierung ist der Prozeß, bei dem ein Softwareprodukt in ganzen Schritten erstellt wird, d.h., jede neue verbesserte Version des Produkts ist ein vollständiges Produkt. Dieses wird mit der eingebrachten Verbesserung in einem sogenannten Integrationstest überprüft.

Bottom-Up-Implementierung
Bei der Bottom-Up-Implementierung wird ein Softwareprodukt in kleinen Einzelkomponenten erstellt. Diese werden dann einzeln für sich geprüft und anschließend in ein vollständiges Produkt integriert.

Testen und Validierung als Maßnahmen zur Kontrolle der Qualität beziehen sich nach ISO 9000 vorrangig auf die Testdokumentation und nicht auf den Testvorgang selber. Ein wichtiges Thema bei ISO 9000-3 ist die Testplanung. Es folgen Auszüge:

- Der Testplan sollte Testfälle, Testdaten und erwartete Ergebnisse enthalten.
- Der Testplan sollte die Stufen der durchzuführenden Tests, z.B. Funktionstest, Test der Grenzbedingungen, Leistungstests und Brauchbarkeitstests beschreiben.
- Der Testplan sollte die Testumgebung, Werkzeuge und Testsoftware beschreiben.
- Die Überprüfung des Testplans sollte auch die Anwenderdokumentation, das erforderliche Personal und die Kriterien für die Vollständigkeit des Tests umfassen.

Hinsichtlich der Testdurchführung fordert ISO 9000-3:

- Die Testergebnisse sollten, wie in der Spezifikation festgelegt, aufgezeichnet werden.
- Alle entdeckten Probleme und ihre möglichen Auswirkungen auf andere Teile der Software sollten vermerkt und die Verantwortlichen informiert werden, damit alle Probleme angegangen werden können, bis sie gelöst sind.
- Bereiche, die von irgendwelchen Veränderungen betroffen sind, sollten gekennzeichnet und erneut getestet werden.

- Die Angemessenheit und Zweckdienlichkeit der Tests sollte bewertet werden.
- Die Hard- und Softwarekonfiguration sollte beachtet und dokumentiert werden.

Unabhängig davon, wie effektiv die Inspektion beim Herausfinden von Fehlern sein mag, basieren alle statischen Testmaßnahmen auf der Annahme, daß das Design und die Forderungen vom Grundsatz her richtig sind. Hingegen erlaubt das Testen des Softwareprodukts die Durchsicht der Ergebnisse. Der Prozeß der Fehlerbeseitigung in der Software muß verschiedene Techniken beinhalten, wobei jede Technik die andere ergänzt.

Als Managementaufgabe zur Steuerung der Qualität ist der Softwareentwicklungsprozeß an Programmierrichtlinien auszurichten.

- Behandlung von Ausnahmesituationen
- Diagnostische Bereitschaft (Debug-Code)
- Spielraum (Raum und Zeit)
- Aufzeichnung (Mitschreiben des Verhaltens des Programms)
- Begrenzung der Komplexität
- definierte Programmierpraktiken
- Fehlertoleranz

Aufgaben zur Organisation der Qualitätssicherung bestehen in:

- Rückverfolgen von Prozeßmerkmalen
- Prozeßverbesserung
- Reviews und Audits
- Produkt (-unabhängige) Tests
- Konfigurationsmanagement
- Änderungsverfolgung
- Vertragsmanagement (bei eventuellem Einsatz von Subauftragnehmer)
- Unabhängige Validierung und Verifizierung

8 Beschreibung der Testumgebung

Während der Testausführung fallen eine Vielzahl verschiedener Daten an, die aus folgenden Gründen abgespeichert, archiviert und analysiert werden müssen.

- Reproduzierbarkeit der Tests
- Wiederverwendbarkeit von Testdaten
- Testdokumentation
- Information über Bearbeitungsstände und ermittelte Abweichungen

Die Testaktivitäten im ITB lassen sich im wesentlichen in die folgenden vier Phasen unterteilen.

Phase 1:
In der ersten Phase wird die Vorbereitung der gesamten AuT (Application under Test) vorgenommen. Hier wird zu allererst ein Angebot an den Auftraggeber [AG] erarbeitet, das auf der Basis einer groben Testobjektabgrenzung erfolgt. Es werden bereits die vom AG festgelegten Kritikalitäten für fachliche Funktionen der Anwendung berücksichtigt, um eine Zuordnung der Kategorien der Testobjekte zu ermöglichen. Nach Vertragsabschluß findet eine detaillierte Testobjektabgrenzung statt, woraufhin der Testplan erstellt werden kann. Dann werden für jedes Testobjekt Testaufträge erstellt, die als "Pflichtenheft" an die Tester weitergegeben werden. Der Testauftrag gibt während des gesamten Testprojektes Auskunft über den aktuellen Bearbeitungsstand des Testobjektes.

Phase 2:
In der Phase 2 erfolgt die Testvorbereitung durch den Tester auf der Grundlage der fachlichen Vorgaben des Auftraggebers. Entsprechend den Festlegungen im Testplan werden jetzt die Testfälle mit unterschiedlichem Aufwand ermittelt und allgemeinverständlich beschrieben. Dabei treten je nach Detaillierungsgrad und Vollständigkeit des Fachkonzeptes Widersprüche und Fragen auf, die die Tester festhalten müssen, um vom Auftraggeber eine Berichtigung oder Vervollständigung seiner Vorgaben zu erwirken. Für die bearbeitbaren Testfälle werden nachfolgend die Testdaten definiert. Anschließend werden die Testprozeduren erarbeitet und beschrieben (Mitschnitte, Scripte, Jobs).

Phase 3:
Phase 3 beinhaltet die eigentliche Testausführung. Dafür müssen zeitliche Festlegungen für sequentielles Testen bezüglich der Sekundärdatenerstellung und Testprozeduren getroffen werden. Zur Unterstützung des parallelen Testens ist die Abgrenzung der Sekundärdaten für die unterschiedlichen Zyklen notwendig. Es dürfen sich keine Funktionen / Anwendungen gegenseitig stören oder deren Daten beeinflussen. Die bei der Testausführung entdeckten Abweichungen werden vom Tester detailliert beschrieben, um in der nachfolgenden Abweichungsbewertung Mißverständnisse zu vermeiden und die Reproduzierbarkeit zu erleichtern. Weiterhin ist es notwendig alle Istergebnisse (Datenbankinhalte, relevante Dateien) und Bildschirmein- und ausgaben Testzyklusbezogen abzuspeichern. In der Abweichungsbewertung wird entschieden, ob die Abweichung ein "echter Anwendungsfehler" ist oder vom Tester verursacht wurde und welche Priorität ihr bei der nachfolgenden Behebung zugewiesen wird. (Ad hoc Soll-Ist-Wert-Vergleich).

Phase 4:
In der Phase 4 werden alle nachfolgenden Regressionszyklen zusammengefaßt , da hier nur noch die Testausführung relevant ist. Je nach Eingangsbedingungen erfordert der jeweilige Regressionstestzyklus einen Neubeginn der Tests mit unterschiedlichen Testaktivitäten. Bedingt durch eine Fehleinschätzung der Komplexität der Testobjekte bei der Testobjektabgrenzung oder durch Veränderungen der Vorgaben des Auftraggebers kann eine Überarbeitung der Testobjekte notwendig sein.
[siehe Abbildung 3.7-1][Vett]

Für ein derartiges Konfigurations- und Testmanagement ist es im ITB erforderlich eine Verwaltung der Ergebnisse der Testvorbereitung und Testausführung zu definieren und zu realisieren. Aufgrund der Verwendung des Testdirectors im Intranet-Testbed zur Testverwaltung und Steuerung der anderen Mercury-Produkte (Loadrunner, Winrunner, XRunner) wird im Rahmen der Diplomarbeit ein Programm entwickelt, welches weiterhin die Sicherungen der Primärdaten und Sekundärdaten sowie der Sicherungsläufe verwaltet. Außerdem soll die Schnittstelle zur Remote-Steuerung anderer Rechner über Batch/Script/Jobs möglich sein. In einer weiteren Ausbaustufe soll es möglich sein, die verwendenten Datenbanken des Testdirectors mit den Tables des zu entwickelnden Programmes abzugleichen, um eine redundante Datenhaltung zu vermeiden. Die Rechteverwaltung soll über speziell definierte Ansichten definiert werden. Dafür ist zwischen den Rechten zum Anlegen/ Löschen/ Ändern von Projekten sowie dem Anlegen/ Löschen/ Ändern von Testfällen und dem bloßen Lesen der gesamten Daten zu unterscheiden.

Wie die [Abbildung 3.7-1] zeigt, verlangt der gesamte Testprozeß vom Angebot / Auftrag bis zum Testende gemäß den gesetzten Testendekritärien einige Verzweigungsmöglichkeiten, die eine hohe Flexibilität in der Strukturierung der programmtechnischen Verwaltung verlangen.
Zur flexiblen und strukturierten Verwaltung der Testdaten wird auf eine relationale Datenbank und die plattformunabhänige Programmiersprache Tcl/Tk zurückgegriffen. Damit ist es möglich, die simultane Verwendung einer Testverwaltung und die Testobjekte gleichzeitig auf jeder beliebigen Plattform zu betreiben. Momentan existiert ein Tcl/Tk-Interpreter für die Zielplattformen: Windows 9x/NT; Windows 3.x (win32s); Linux; HP-UX; SUN Solaris / OS; AIX; Macintosh u.a.. Damit sollte der erzeugte Code auf allen Plattformen portabel sein.

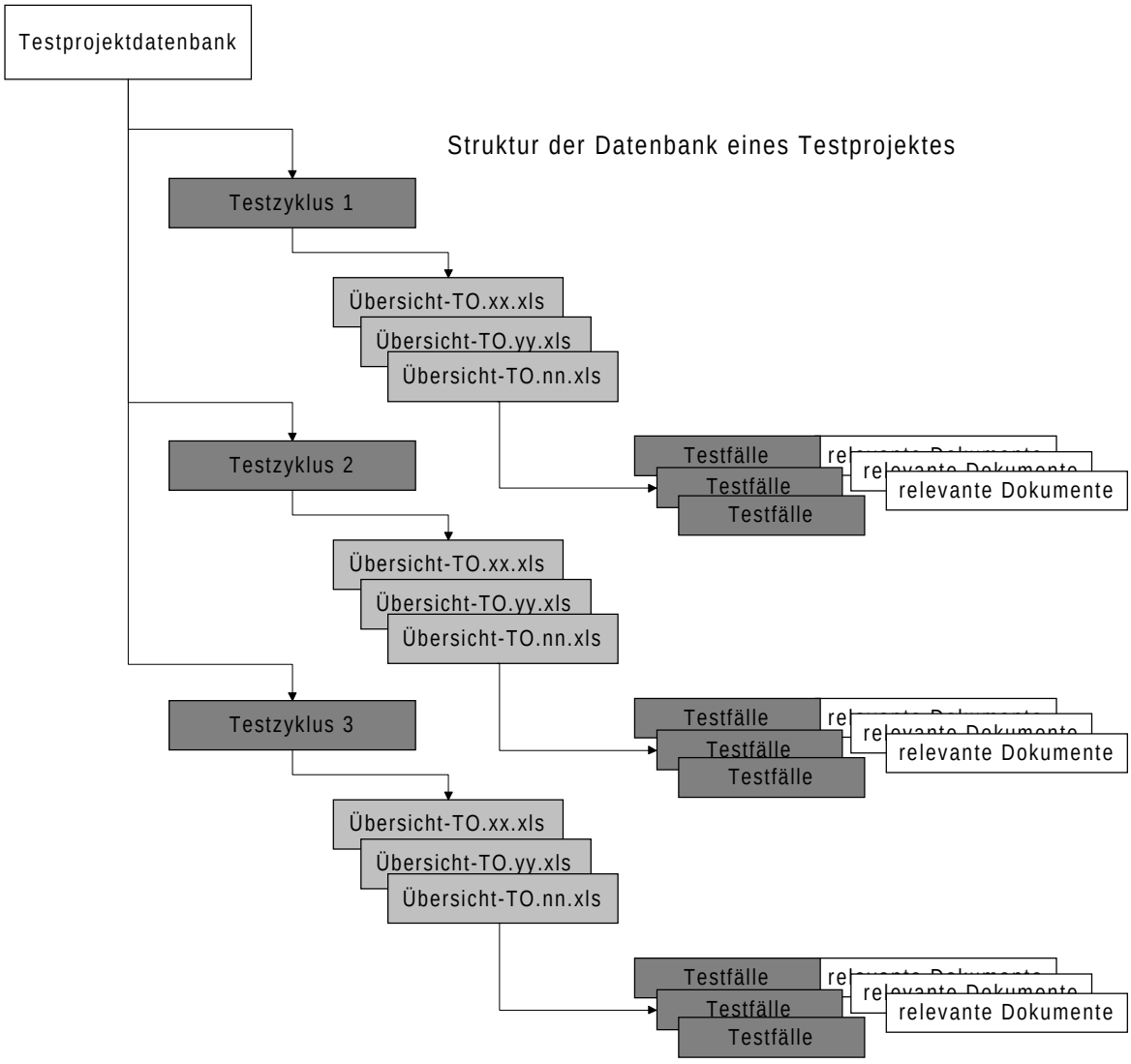


Abbildung 7.4-1: Projektdatenbank [Vett]

Die zu erzeugende Datenbank soll alle relevanten Daten eines Testobjekts sammeln und speziell gefiltert und sortiert ausgeben können. Das Grundgerüst der Datenbank soll eine Baumstruktur sein [siehe Abbildung 7.4-1]. Die Wurzel bildet das jeweilige Projekt, das in die zu bearbeitenden Testzyklen etc. verzweigt. In den Testobjekten sollen die Basis-Testfalldokumente enthalten sein. Mit jedem neuen Testzyklus werden die Zweige über die Testobjekte zu den Testfällen neu aufgebaut. Die Verwaltung aller Testzyklen in einer Testprojektdatenbank ist Grundlage für die Auswertungsfähigkeit der Daten über das gesamte Testprojekt. Das Programm soll teststufenunabhänig sein. Es soll nur eine Anpassung des Basis-Testfalldokuments von nöten sein, wenn eine neue Teststufe damit verwaltet wird.

8.1 Organisation der Testumgebung

Die Testumgebung wird zur Unterstützung der Testplanung und Testverwaltung eingesetzt. Alle Testaktivitäten werden darin geführt und dokumentiert. Damit ist zu jeder Zeit eine Aussage über den Bearbeitungsstand der Testobjekte möglich. Die Testplanung wird damit transparenter und erleichtert erheblich die Re-Tests. Abweichungen und Ergebnisse können damit dokumentiert und Fehler verfolgt werden (Defect-Tracking).

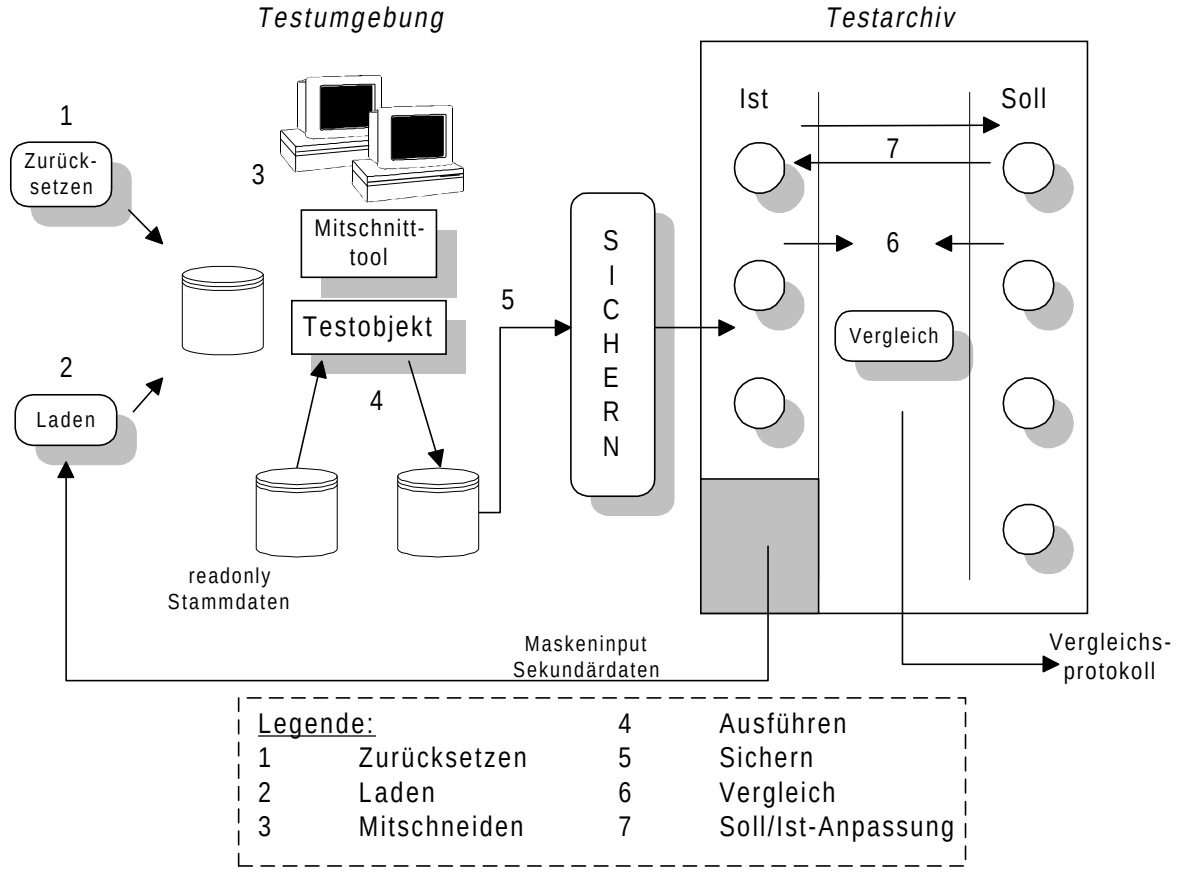


Abbildung 8.1-1: Testaktivitäten in der Umgebung

8.2 Die Testobjektverwaltung

Innerhalb der TO-Verwaltung werden die Testdaten mit den in der Testumgebung integrierten Mechanismen in das Zielformat überführt. Dies kann über ein Einspielen der gesicherten Testumgebung mittels des Archivierungssystems „Networker Legato“, über das Einspielen von neuen Datenbankinhalten mittels SQL-Scripten oder über Neuanlegen der Testdaten mittels Scripten, Batch oder parametrisierten RUN-Läufen der Softwarepakete „Win-Runner, Load-Runner“ erfolgen.

Die bei der Testausführung entstehen Ausgaben/Ist-Ergebnisse werden zur Ergebniskontrolle außerhalb der Testumgebung archiviert. Datei basierte Ergebnisse (Screenshots der Systemmeldungen, AVI‘s, Graphiken) werden über das File-Transfer-Protokoll in den, den Testlauf repräsentierenden Verzeichnispfaden auf einem zentralen Rechner abgespeichert und die eindeutigen Dateinamen (die Zugehörigkeit zu einem bestimmten Testlauf wird durch den Verzeichnispfad bestimmt) werden in die Verwaltungsdatenbank geschrieben. Pflichtfelder zur Verwaltung der Testläufe werden vom jeweiligen Tester bzw. von der Software automatisch geschrieben. Beim Übernehmen von vorhandenen Testläufen in neue, werden alle Ergebnisdaten (einschließlich der dateibasierten Ergebnisdaten innerhalb der FTP-Struktur) kopiert. Nur dadurch ist eine konsistente Datenhaltung möglich.
Korrekte oder weitestgehend korrekte Testergebnisse werden als Soll-Ergebnisse für automatische Re-Tests in der Testverwaltung abgelegt. Es ist eine komplette oder auch teilweise Re-Definition der Soll-Ergebnisse möglich.

Bei der Testausführung werden sie dann direkt als Vergleichsdaten in der Testverwaltung aufbereitet, sodaß der Testlauf bei Bedarf mit einem automatischen Soll- /Istwert-Vergleich abgeschlossen werden kann. Hierbei werden Differenzen zwischen den Ist- und Soll-Ergebnissen dokumentiert und über ein Webinterface präsentiert.

Zur Änderungskontrolle in der Verwaltungsdatenbank wird die Kombination der Feldwerte *?_wer* und *?_wann* in dem jeweiligen TABLE eingesetzt. Diese Eingaben erfolgen automatisiert durch die Software, der Tester hat keinen Einfluß auf den Feldinhalt. Um ein Undelete von Datensätzen zu ermöglichen, wird das Flag *?_aktiv* in jedem Datensatz benutzt. Gelöschte Datensätze werden in der Datenbank inaktiv gesetzt und über die Masken innerhalb der Software nicht angezeigt. Über pure-SQL sind sie sichtbar und können vom Administrator über spezielle Masken oder über cron-jobs zeitabhängig gelöscht oder wieder aktiv gesetzt werden. Eine Beschreibung der Grundlagen relationaler Datenbanken und eine Darstellung der Datenbankarchitektur zur Verwaltung der Testdaten/-umgebung ist in [Kapitel 12.1] ersichtlich.

8.3 Das Archivierungssystem

Zur Sicherung der Verwaltungsdatenbank stehen unterschiedliche Möglichkeiten zur Verfügung:
1. die online-Sicherung auf Filesystem-Ebene,
2. die offline-Sicherung auf Filesystem-Ebene,

3. der Datenexport mit den Tools der Oracle-Datenbank,
4. die Sicherung über das Oracle-PlugIn des Archivierungssystems „Networker Legato“.

Von der ersten Sicherung ist prinzipiell abzuraten, da Oracle aus Performancegründen den Table-Space nicht ständig konsistent hält. Die zweite Sicherungsart sollte nur bei „Ad hoc“-Sicherungen eingesetzt werden. Der Datenexport mittels den Oracle-Tools bietet die größtmögliche Sicherheit bezüglich der Datenintegrität, besitzt jedoch unzureichende Möglichkeiten hinsichtlich der Zeitsteuerung / Automatisierung. Die letzte Sicherungsmöglichkeit bietet die größtmögliche Flexibilität und Sicherheit, auch aufgrund des Sicherungsmediums (DLT 7000). Das Backup und Archivierungskonzept ist in [Kapitel 12.2] dargestellt.

9 Resümee und Ausblick

Das Testen erfolgt planmäßig, strukturiert und wird ausreichend dokumentiert. Der forcierte Ausbau der Testautomatisierung weist auf ein gewachsenes Problembewußtsein, wirtschaftliches Denken und auch eine Änderung der Einstellung zur Qualität in der Managementebene hin.

An der entstandenen Testumgebung wäre eine Weiterentwicklung bezüglich der Integration der Tools zur Steuerung der Primär- und Sekundärdatenbestände / Remote-Steuerung in die Test-Director-Verwaltung sinnvoll. Über das Test-Director-Programmierinterface (Visual-API) ist diese Integration bzw. der Abgleich der vom Testdirector benutzten Oracle-Datenbank mit den Datenbankeninhalten zur Verwaltung der Testumgebung, machbar. Weiterhin ist die Entwicklung von Reportgeneratoren über das Web-Interface der Oracle-Datenbank möglich. Diese Reports sollten dann Intranet/Internet-weit den Entwickler über den Stand der Tests informieren und gleichzeitig die Software plattformunabhängig um eine Druckfunktion erweitern.
Denkbar wäre auch die Definition eines XML-Styles zum Datenexport aller für den Entwickler relevanten Informationen für einen plattformunabhängigen Datenaustausch.

9.1 Evaluierung von Testwerkzeugen

Hersteller	**Name**	**Bemerkung**
ATS, Automated Testing Solutions	CTE	Graphischer Editor für die Aufbereitung von Testfällen für den Funktionstest
ATS, Automated Testing Solutions	Tessy	(c) Daimler-Benz AG, T-organisation, T-dateneingabe, Sollwerteingabe, T-fallermittlung, T-vorbereitung, T-durchführung, Monitoring, T-auswertung, T-dokumentation
Auto Tester Inc.	Auto Advisor	Testmanager
Auto Tester Inc.	Auto Tester	Capture Replay-Tool
Auto Tester Inc.	Client / Server	Client/Server Testtool
Auto Tester Inc.	Distributed Test Facility	Lade und Streßtest
Auto Tester Inc.	Test Library Manager	Testmanager
Auto Tester Inc.	Test Station	C/R-Tool für CUI-Oberflächen
Auto Tester Inc.	V-Test	Client/Server-Testtool
Bender & Associates	Softtest	Testfalldesign und –analyse
CA-Computer Associates	CA-Datamacs/II	Testdatengenerator
CA-Computer Associates	CA-Traps	C/R-Tool
Candle GmbH	Candle Command Center	Monitoring-Tool für SAP R/3
Cayenne Software Inc.	Ensemble Test	Testfallgenerierung; Metriken für C
Cayenne Software Inc.	Object Team Sim	Testsimulation
Centre Line Software Inc.	QC/Coverage	Testabdeckungsmessung
Centre Line Software Inc.	QC/Reply	GUI-Regressionstest; Streß- und Ladetest
Centre Line Software Inc.	QC/SIM	Simulation der Testbedingungen
Centre Line	Vista Reply	GUI-Regressionstest; Code-Abdeckungsmessung

Hersteller	**Name**	**Bemerkung**
Software Inc.		
Centre Line Software Inc.	Vista SIM	Simulator
Centre Line Software Inc.	Vista Test	statische und dynamische Code-Analysen
Centre Line Software Inc.	Test Center	Registrierung von Laufzeitfehlern und Speicherlöchern
Compuware Corp.	Abend-AID/XLS	Abnormal-End-Handling-Tool für die Analyse, Diagnose und Reports
Compuware Corp.	Compare-XPERT	Daten-Vergleichs-Programm für Client/Server
Compuware Corp.	Conversion-XPERT	Daten-Konvertierungs-Programm
Compuware Corp.	Data-XPERT	Daten-Manipulations-Programm für Client/Server
Compuware Corp.	DBA-XPERT	Datenbank-Änderungs-Management
Compuware Corp.	EcoCONNECT	Integriert alle Eco Tools in einer Oberfläche
Compuware Corp.	EcoPMON	Real-Time Prozess-monitoring
Compuware Corp.	EcoTOOLS	Integriertes System-Management für Client/Server Applikationen
Compuware Corp.	File-AID	Daten-Manipulations-Programm
Compuware Corp.	Preditor	Oberflächen-Programmier-Tool
Compuware Corp.	QACenter	Gesamtpaket (QARun, QAPlan, QATrack)
Compuware Corp.	QADirector	Testmanagement
Compuware Corp.	QALoad	DB-Ladetest
Compuware Corp.	QAHiper-station	C/R-Tool
Compuware	QAPlan	Testmanagement

Hersteller	**Name**	**Bemerkung**
Corp.		
Compuware Corp.	QARun	C/R-Tool
Compuware Corp.	QAStreß	Streßtest
Compuware Corp.	QATrack	Fehlerverfolgung
Compuware Corp.	Related-XPERT	Daten-Migrations-Programm für Client/Server
Compuware Corp.	Remote Control/2	Laufzeitkontrolle für OS2
Compuware Corp.	UNIFACE	Entwicklungstool für Client/Server-Applikationen
Compuware Corp.	XPediter	Dynamsiches Debugging Tool
ConSol	CallManager	Helpdesk- und Callcenter-Software
ConSol	xlintf	Prüfprogramm für Programmierrichtlinien (Style Guide)
ConSol	xtest	Umgebungsgenerator für Regressionstest
Coopers & Lybrand	Diagnostic 2000	Datei und Programm-Scanner für Datumsfelder
CYRANO	Impact	Client/Server Performance-Test
CYRANO	Insight	Visualisierung des Client/Server-Datenaustauschs
CYRANO	Workbench	Real-Time-Analyse der Transaktionen von SQL-Servern
Dassault Electronique	Devisor	Debugger, Testabdeckungsmessung
Dassault Electronique	Sylvie	Entwicklungswerkzeug
DATAMAT	Valid/X	Komplettes Tool für das dynamische Testen
Eastern System Inc.	Test Designer/ Planer	Client/Server Regressionstest, Benchmarking, Ladetest, Kompatibilitätstest mit Netzwerkunterstützung
Eastern System Inc.	Evaluator	Regressionstest, Streß- und Ladetest
Elverex	Evaluator FT	C/R-Tool, Unittestunterstützung,

Hersteller	**Name**	**Bemerkung**
		Performancemessungen, T-Plan-Schnittstelle
Fastcase Technology Ltd.	Insure++	Run-Time-Debugger, Tracing, Speichermonitor, Testabdeckungsmessung (C,C++)
Geodesic Systems	Great Circle	Speicherverwaltungstest für MS Visual C++
IDE	StP/T	Regressionstest (plattformunabhängig), Streß-und Ladetest,
IntegriSoft Inc.	Hindsight Flow Charter	Flußdiagrammerstellung auf der Codebasis
IntegriSoft Inc.	Hindsight Module Test Environment	Modultest-Automatisierung, Generierung von Treibern und Stubs
IntegriSoft Inc.	Hindsight Report Generator	
IntegriSoft Inc.	Hindsight Simulation	Generierung von Error-Handling-Routinen
IntegriSoft Inc.	Hindsight Software Assurance Toolkit	Visualisiert Software-Metriken
IntegriSoft Inc.	Hindsight Structure and Logic Analysis	Logik-Analysator mit Visualisierung für den Source-Code
IntegriSoft Inc.	Hindsight Test Coverage Analysis	Testabdeckungsanalyse
IntegriSoft Inc.	iSight++	Code Inspection für C, C++ und Fortran
Intersolv	PVCS Configura-tion Builder	Entwicklungswerkzeug
Intersolv	PVCS Developer Toolkit	Entwicklungswerkzeug
Intersolv	PVCS Production Gateway	Projektverwaltungstool
Intersolv	PVCS Reporter	Generator für projekt- und kundenspezifische Berichte
Intersolv	PVCS Tracker	Organisations- und Überwachungstool für den Softwareentwicklungsprozeß
Intersolv	PVCS Version Manager	Konfigurationsmanagement
IPL Information	AdaTest	Code-Inspection, statistische Analyse

Hersteller	**Name**	**Bemerkung**
Processing Ltd.		
IPL Information Processing Ltd.	AdaTest 95	Dynamisches Testen, Testabdeckungsmessung, Code-Inspection
IPL Information Processing Ltd.	Cantala	Dynamisches Test, Testabdeckungsmessung, Code-Inspection
LDRA Ltd.	LDRA Testbed	Code Inspection, statische und dynamische Analysen
Marben	ATTOL / System Test	Testunterstützung bei verteilten Systemen
Marben	ATTOL / Unit Test	Unittestunterstützung für C/C++ und Ada
McCabe & Assiciates	McCabe Visual Toolset	Code-Inspection
Mercury Interactive Ltd.	LoadRunner	Lade- und Streßtest, Website-Test auf Protokoll-Ebene
Mercury Interactive Ltd.	SQL-Inspector	Analysiert SQL-Anfragen, Debugging von datenbankspezifischen Fehlern
Mercury Interactive Ltd.	Test Director	Testmanagement und Fehlerverfolgung
Mercury Interactive Ltd.	Test Suite	Komplettpaket aller Testtools
Mercury Interactive Ltd.	WinRunner	C/R-Tool, Website-Test auf GUI-Ebene (Windows)
Mercury Interactive Ltd.	XRunner	C/R-Tool, Website-Test auf GUI-Ebene (Unix)
Microsoft Corp.	Visual-Test	C/R-Tool
microTOOL GmbH	objektiF	Integrierte Softwareentwicklungsumgebung
MID GmbH	innovator	Werkzeug zur Entwicklung von integrierten, unternehmernsweiten Informationssystemen, CASE-Tool
Nu-Mega Technologies	Bounds Checker	Fehlerermittlung
Performance Awareness Corp.	preVue	Regressionstest; Streß- und Ladetest
Performance	preVue-X	Regressionstest; Streß- und Ladetest

Hersteller	**Name**	**Bemerkung**
Awareness Corp.		
Performance Software	V-Test	Regressionstest; Streß- und Ladetest, Multiusertest
Precision Software	ERwin	Datenbank Design- und Modellierungstool
Precision Software	Graphical Designer	Graphisches Werkzeug für objektorientierte Entwicklung
Precision Software	MainWin XDE	Windows-Entwicklungsumgebung für Unix
Precision Software	NobleNet RPC	Compiler Toolkit für RPC-Programmierung
Precision Software	Software Test Works	C/R-Tool, Testabdeckungsmessungen
Precision Software	VisiBroker	Object Request Broker für C++ und Java
Precision Software	VisiODBC	Zugriffsprogrammierung für SQL-Datenbanken
Precision Software	XVT	GUI-Entwicklungstool für C/C++
Programming Research Ltd.	QA C	Code Inspection für C, Unittest-Automatisierung
Programming Research Ltd.	QA C++	Code Inspection für C++, Unittest-Automatisierung, OO-Metriken
Programming Research Ltd.	QA Manager	Lifecycle Control System
Programming Research Ltd.	QC-Coverage	Testabdeckungsmessung (C/C++)
Programming Research Ltd.	QC/Reply	C/R-Tool
Promark Inc.	Rhobot/ Classic	Regressionstest; Streß- und Ladetest; Multiusertest
Promark Inc.	Rhobot/ Client-Server	Steßtest für Client/Server-Anwendungen
Pure Software	Pure-Coverage	Identifizierung von ungeprüftem Programm-Code
Pure Software	Pure-DDTS	dezentrales Fehlerverfolgungssystem, Berichts- und Managementstatistikgenerator
Pure Software	Pure-Link	inkrementeller Linker

Hersteller	**Name**	**Bemerkung**
Pure Software	Pure-Performix /Web /CS /X /TTY	Testtool für Performancemessungen von Web-Servern, Client/Server-Systemen, X-Windows-Anwendungen, TTY-Anwendungen
Pure Software	Pure-TestExpert	Automatisiertes Testmanagement-System mit dem Tests, Teststrukturen und Ergebnisberichte erstellt, verwaltet und durchgeführt werden
Pure Software	Pure-Vision	Automatischer Test
Pure Software	Purify	Ermittlung von Laufzeitfehlern und Speicherlöchern
Pure Software	Quantify	Performancetestwerkzeug
Quality Engineering Software	QES Architect	Multiuser Regressionstest, Streßtest, Ressourcenverbrauchsmessung
Quality Engineering Software	QES EZ for GUI	Regressionstest, Test der Anwendungslogik
Reed Software Inc.	ProTerm	Testtool
Rogue Wave Software GmbH		diverse Entwicklungswerkzeuge
Segue Software Inc.	GO!	Testausführung
Segue Software Inc.	QA DBTester	Testdatenbewertung
Segue Software Inc.	QA Organizer	Testmanagement
Segue Software Inc.	QA Partner	C/R-Tool
Segue Software Inc.	QA Radar	Automatische Fehlererkennung und –lokalisierung in SQL-basierten Abfragen
Segue Software Inc.	Quality Works for Windows	Gesamtpaket aller Produkte
Segue Software Inc.	SURF!	WebTest
SELECT Software Tools Inc.	SELECT Enterprise	objektorientierte Modellierungsumgebung

Hersteller	**Name**	**Bemerkung**
Softbridge Inc.	ATF/ Basissystem	Netzwerktest
Softbridge Inc.	ATF/ TestLab Command Center	Multi-User-Test
Softbridge Inc.	ATF/ TestWright	C/R-Tool, Regressionstest
Software Quality Assurance Ltd.	Tplan	Test-Planung, -Design und Management
Software Research Inc.	STW/ Advisor	statische Analyse, Testdatengenerierung
Software Research Inc.	STW/ Coverage	Testabdeckungsmessung
Software Research Inc.	STW/ Regression	Client/Server Regressionstest, objektorientiertes oder echtzeit-C/R, Testmanagement
Software Research Inc.	STW/ TestWorks	Gesamtpaket aller Produkte
SQA Inc.	SQA Client/ Server Edition	Gesamtpaket aus LoadTest und TeamTest
SQA Inc.	SQA LoadTest	Lade-,Streß- und Multiuser-Testsystem
SQA Inc.	SQA Manager	Testmanagement
SQA Inc.	SQA Process	Testfallermittlungswerkzeug
SQA Inc.	SQA Robot	C/R-Tool
SQE	STEPMaster	Test-Management
SQS GmbH	Test-Cadett	systematische Testfallermittlung
SQS GmbH	Test-Dat	Testdatendefinition/ -erstellung (formatunabhängig)
SQS GmbH	Test-Man	Testauftrags- und Abweichungsverwaltung
SQS GmbH	Test-Proc	Datenbanken laden, Eingabedaten treiben
Sterling Software	COMPAREX	Tool zum Datenvergleich
Sterling Software	TestPro	C/R-Tool
STSc, Software Testing Science	GUI-SE	GUI-Design und Testtool
STSc, Software Testing Science	SSTM	Tool zur Strukturanalyse
STSc, Software Testing Science	TestMaster	CAST-Tool
Telelogic	ITEX	Testtool zur Erweiterung der DIE
Verilog SA	Adele	Konfigurations- /Version- und Prozeßmanagement
Verilog SA	Logiscope	Code-Analyse, Testabdeckung,

Hersteller	**Name**	**Bemerkung**
		Programmierrichtlinien
Vermont Creative Software	Vermont High Test	Regressionstest
Via Inc.	SmartTest	Quellcodetest, Debugger
Zott + Co. GmbH	Performance	Streß- und Lasttest
Zott + Co. GmbH	Qs_atum	Testmanagement
Zott + Co. GmbH	s_atum	Regressionstest
Zott + Co. GmbH	Xs_atum	Regressionstest

Tabelle 9.1-1: Evaluierung von Testwerkzeugen [SQS-QT]

10 Literatur

[Balz]	Lehrbuch der Software-Technik; Software-Entwicklung	H. Balzert; 1. Auflage; Spektrum-Akademischer Verlag; 1996
[ct05/99]	Windows-Nachrichten filtern mit Hooks-Funktionen	K.-U. Mrkor; ct – Magezin für Computertechnik; Heise Verlag 1999
[ct19/98]	Software-Engineering bändigt Entwicklungsprojekte	A. Bleul, Dr. J. Loviscach; ct – Magazin für Computertechnik; Heise Verlag 1998
[DBLin]	Datenbanken mit Linux	Dr. B. Röhrig; Computer & Literaturverlag; Vaterstetten 1998
[Explor]	Exploring Expect	Don Libes; O'Reilly & Associates; 1997
[FHBiel]	Compilerbau	FH Bielefeld; FB Elektrotechnik; http://parallel.fh-bielefeld.de/pv/vorlesung/cpt/script/kap1.htm
[Grimm]	Systematisches Testen von Software – Eine neue Methode und eine effektive Teststrategie	K. Grimm; GMD-Forschungszentrum Informations-technik GmbH; GMD-Bericht Nr. 251; R.Oldenbourg Verlag 1995
[ISO9000]	ISO 9000 und Softwarequalität	Ö. Oskarsson, R. Glas; Prentice Hall, München 1997
[Leitf]	Leitfaden Test von IV-Anwendungen;	Deutsche Telekom AG; Bonn 1996
[Müllerb]	Test, Analyse und Verifikation von Software	M. Müllerburg u.a.; GMD-Forschungszentrum Informations-technik GmbH; GMD-Bericht Nr. 260; R. Oldenbourg Verlag
[relDB]	Funktion und Arbeitsweise einer relationalen Datenbank	H. Bujak; Media Transfer GmbH; Funktion und Arbeitsweise einer relationalen Datenbank am Beispiel Oracle
[SQS-QT]	Testorganisation,-verfahren,-Tools	SQS, Gesellschaft für Software-Qualitätssicherung mbH; Stollwerckstraße 11; D-51149 Köln
[STZMag]	Software-Testzentrum der Uni Magedburg	http://ivs.cs.uni-magdeburg.de/sw-eng/us/metrics/gcontents.shtml
[Taube]	Stand der kontinuierlichen Verbesserung in der	F. Taube; Diplomarbeit; Uni Köln; Wirtschaftsinformatik; http://www.informatik.uni-

	Softwareentwicklung	koeln.de/winfo/prof.mellis/publications/taubedpl.zip
[Traub]	SW-Qualitäts-Sicherung; Konstruktive und analytische Maßnahmen	H. Trauboth; 2. Auflage; Oldenbourg 1996
[TestDat]	Testdatendefinition mit TestDat Version 0.1	SQS, Gesellschaft für Software-Qualitätssicherung mbH; Stollwerckstraße 11; D-51149 Köln
[Tintern]	Projektname: Qffice im Netz	PNr.: HTP E80450 ; DLV 10194 Erstellt von: M. Wolf (T-Berkom GmbH)
[Myers]	Methodisches Testen von Programmen	G. J. Myers; 5.Auflage; Oldenburg 1995
[Vett]	Verwaltung der Ergebnisse der Testvorbereitung und Testausführung	M. Wilke; Berkom GmbH; Berlin 1998
[Wallm]	Ganzheitliches Qualitäts-management in der Informationsverarbeitung	E. Wallmüller; München-Wien 1995

Tabelle 10-2: Literaturverzeichnis

11 Glossar

Blackboxtest	Datengetriebener oder Ein / Ausgabetest- nur die Spezifikation ist ausschlaggebend für den Test und nicht die interne Programmstruktur, dafür ist der vollständige Eingabetest entscheidend. **Vorteil**: untersucht echte spezifikationsbezogene Programmeigenschaften **Nachteil**: Das Testen beginnt sehr spät im Entwicklungsprozeß **Methoden**: Äquivalenzklassenbildung; Grenzwertanalyse; Ursache-Wirkungsgraph; Fehlererwartung (error guessing)
Datenbank	Eine konkrete Anwendung bzw. Applikation in einem Datenbanksystem.
Datenbanksystem	Die Menge aller Programme die zum Betrieb einer Datenbank notwendig sind.
Design	Design beschreibt, wie ein System etwas tun soll. Dies kann entweder allgemein (als Design-Spezifikation) oder detailliert (als Quellcode in einer High-Level-Sprache) beschrieben werden.
Disjunktheit	Kein denkbaren Zustand kann mehreren Elementklassen zugeordnet werden.
Diversifizierender Test	(Mutationstest) Ermittlung von Testfällen anhand von Varianten des TOs. Durch Veränderung von arithmetischen oder relationalen Operatoren werden Varianten des TO (Mutanten) aufgedeckt. [Grimm]
Fremdschlüssel	Ein Attribut oder eine Attributkombination einer Relation heißt Fremdschlüssel, wenn das Attribut bzw. Attributkombination in einer anderen Relation Primärschlüssel ist. Über die Fremdschlüssel werden Beziehungen zwischen den Relationen dargestellt.
Performance-Test	Beim Performance-Tests werden Volumen- und Streßtest parallel mit dem Ziel ausgeführt, das System unter allen erdenklichen Belastungssituationen einzuschätzen. Die System-Performance ist damit ein allgemeines Maß für Antwortzeiten und Durchsatz-Raten unter verschiedenen Verarbeitungsgeschwindigkeiten und Konfigurationen.

Primärdaten	Bewegungsdaten; Testdaten für ein TO, die die Verarbeitung wesentlich steuern, z.B. Maskeneingaben, Parameter.
Primärschlüssel	Der Primärschlüssel einer Relation besteht aus einem oder mehreren Attributen, die mit ihren Werten jedes Tupel der Relation eindeutig identifizieren. Über den Primärschlüssel ist der direkte Zugriff auf die Tupel der Relation möglich (Zugriffshilfe).
Prüfen	Die Inspektion, auch Prüfen genannt, ist ein Vorgang, bei dem ein entstehendes Softwareprodukt geprüft wird, um festzustellen, ob Mängel statisch (also ohne das Programm auszuführen) entdeckt werden können.
Qualitätsmanagement	Alle Tätigkeiten der Gesamtführungsaufgabe, welche die Qualitätspolitik, Ziele und Verantwortungen festlegen sowie diese durch Mittel der Qualitätsplanung, Qualitätslenkung, Qualitätssicherung und Qualitätsverbesserung im Rahmen des QM-Systems verwirklichen. [Wallm]
Re-Test	Regression Testing (rückwärtsausgerichtetes Testen): Wird ein Programm während der Wartung geändert, muß mehr als nur die Auswirkung der beabsichtigten Änderung getestet werden. Das rückwärtsausgerichtete Testen benutzt eine Sammlung von bekannten Testfällen zusammen mit einer korrekten Testvorhersage zu jeder neuen Softwareversion. So wird sichergestellt, daß Änderungen keine Auswirkungen auf darüber hinaus funktionierende Programmteile haben.
Requirement	Requirements beschreiben, was ein System tun soll und was die Anforderungen an das System sind.
Sekundärdaten	Bestandsdaten; Testdaten, die in der Testumgebung zur Ausführung von Tests an TO vorhanden sein müssen / können und nicht zu den Primärdaten zählen, z.B. Tabellen, Stammdaten, Referenzdaten etc. Diese Daten können testobjektspezifisch, projektspezifisch oder projektübergreifend verwaltet werden.
Sekundärschlüssel	Sekundärschlüssel einer Relation sind Attribute oder Attributkombinationen, für die Zugriffshilfen erstellt werden. Sekundärschlüssel können eindeutig oder mehrdeutig sein.
Sicherungsbestand	Menge aller zu einem Zeitpunkt gesicherten Dateien / Daten; Der Sicherungsbestand ermöglicht damit eine Wiederholbarkeit der Testläufe mit den gleichen Bestandsdaten (Re-Test).
Software-Test (effizient)	Unter einem effizienten Software-Test aus Sicht der industriellen Praxis versteht man einen Test, bei dem mit vertretbarem

	Aufwand eine hohe Fehleraufdeckungsrate erzielt wird.
Streßtest	Der Zweck des Streß-Tests ist der Nachweis, daß das System die Kapazität zur Behandlung von großen Mengen an Prozeß-Transaktionen während der "Spitzenzeiten" hat.
Stub	Platzhalter; Testwerkzeug zur Simulation von aufgerufenen Softwareteilen.
Testen	Das Testen ist ein Vorgang, bei dem ein entstehendes Softwareprodukt ausgeführt bzw. zum Laufen gebracht wird, um herauszufinden, ob die von der Software bereitgestellten Ergebnisse auch so erwartet wurden. Damit ist Testen ein dynamischer Prozeß.
Testfall	Menge aller Eingabedaten (NICHT konkret), die für ein TO relevant sind sowie eine Beschreibung des erwarteten Ergebnisses.
Testobjekt	Der jeweilige Testgegenstand (Dokument, Software, Hardware) mit der Beschränkung auf den betrachteten (zu testenden) Funktionsablauf.
Testumgebung	Umfaßt die Gesamtheit aller Werkzeuge die zum Testen, Messen und Prüfen notwendig sind oder erleichternd wirken.
Volumentest	Das Ziel des Volumentests ist das Auffinden von Schwächen des Systems bezüglich des Handlings von großen Datenmengen innerhalb festgelegten Zeiten.
Whiteboxtest	Logisch orientiertes Testen untersucht die interne Struktur des Programmes (meist) unter Vernachlässigung der Spezifikation. Es werden ausgewählte Programmpfade getestet (Anweisungen und Entscheidungen); **Vorteil**: Frühzeitiger Testbeginn **Nachteil**: echte (spezifikationsbezogene) Programmfehler werden nicht erkannt **Methoden**: Erfassung (Ausführung; Coverage) aller Befehle; Entscheidungen; Bedingungen; Mehrfachbedingungen