Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Klaus Franz

Handbuch zum Testen von Web-Applikationen

Testverfahren, Werkzeuge, Praxistipps

Mit 28 Abbildungen und 26 Tabellen

Springer

Klaus Franz
Am Hammelsberg 18b
64521 Groß-Gerau
webtesting.klaus.franz@gmx.de

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

ISSN 1439-5428
ISBN 978-3-540-24539-1 Springer Berlin Heidelberg New York

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

Satz: Druckfertige Daten des Autors
Herstellung: LE-TEX, Jelonek, Schmidt & Vöckler GbR, Leipzig
Umschlaggestaltung: KünkelLopka Werbeagentur, Heidelberg
Gedruckt auf säurefreiem Papier 33/3100 YL – 5 4 3 2 1 0

Für Inge und Daniel

Vorwort

Intention

Auslöser zu diesem Buch war meine Suche nach Informationen zum Testen von web-basierten Anwendungen. Seit vielen Jahren bin ich in der Qualitätssicherung in der Anwendungsentwicklung tätig, erst auf Großrechnern, dann auf Client-Server-Systemen. In diesen Umgebungen haben sich die „klassischen" Testverfahren seit langem bewährt. Aber was muss darüber hinaus beim Testen von Web-Anwendungen getan werden? Müssen die etablierten Methoden den neuen Technologien angepasst werden? Gibt es neue Verfahren und Hilfsmittel? Welche Testtools sind besonders im Web-Umfeld von Bedeutung?

Ich habe keine Literatur ausfindig machen können, die mir persönlich in kompakter Form zufriedenstellende Antworten auf diese Fragen geben konnte. Daher habe ich in diesem Buch Methoden, Hilfsmittel und Werkzeuge, die zur Qualitätssicherung von Web-Anwendungen beitragen, zusammengestellt.

Mit diesem Buch möchte ich ein Nachschlagewerk zum Testen von – nicht nur – web-basierten Anwendungssystemen bereitstellen, das zum einen auf die neuen Technologien eingeht und zum anderen die klassischen Testverfahren nicht aus den Augen verliert.

Danksagung

Das Schreiben eines Buches hat unverkennbare Parallelen zu einem Software-Entwicklungsprojekt: Der Autor (Projektleiter) hat Anforderungen an sein zu verfassendes Werk und gibt sich unter Berücksichtigung aller bekannten Risiken einen Zeitplan vor. Wie in (fast) jedem IT-Projekt verändern sich die Anforderungen und auch die Rahmenbedingungen. Und natürlich treten alle bekannten Risiken ein. Ressourcenengpässe und Terminverschiebungen sind die Folge.

Daher möchte ich mich bei meiner Familie und dem Springer-Verlag bedanken, die das Projekt in entscheidenden Momenten unterstützt und mir bei meinen Aktivitäten zur Seite gestanden haben.

Mein besonderer Dank geht an meine Kollegen und Freunde Siegfried Brauer, Klaus Gockel und Jochen Schneider, die mir mit

ihrem fachkundigen Rat geholfen haben, dieses Buch fertig zu stellen, sowie an meinen Sohn Daniel, der seine Weihnachtsferien für das Korrekturlesen geopfert hat.

Kontakt

Über Verbesserungsvorschläge zum Buch und Tipps für neue Checklisten an webtesting.klaus.franz@gmx.de würde ich mich sehr freuen.

Klaus Franz

Groß-Gerau
Januar 2007

Inhaltverzeichnis

1 Einleitung 1

1.1 Wieso dieses Buch? 1
1.2 Wem nutzt dieses Buch wie? 2
1.3 Wie ist dieses Buch zu lesen? 3
1.4 Welche Testwerkzeuge werden genannt? 5
1.5 Was liefert dieses Buch nicht? 6
1.6 Wer sollte das Buch unbedingt lesen?
Oder: Moti*v*ation für autofahrende Qualitätsskeptiker .. 6

Teil I Handwerkszeug 11

2 Definitionen zur Qualität 13

2.1 Normen und Qualitätsmerkmale 13
2.2 Qualitätsmerkmale für Web-Applikationen 17
2.3 Qualitätsanforderungen 18
2.4 Qualitätssicherungsmaßnahmen 20
2.5 Zusammenfassung 21

3 Begriffe zum Testen 23

3.1 Definitionen zum Testen 23
3.2 Box-Tests 27
3.2.1 ..... Blackbox-Test 28
3.2.2 ..... Whitebox-Test 28
3.2.3 ..... Greybox-Test 28
3.3 Zusammenfassung 29

4 Testfallentwurfsverfahren 31

4.1 Blackbox-Verfahren 31
4.1.1 ..... Äquivalenzklassenanalyse 32
4.1.2 ..... Grenzwertanalyse 36
4.1.3 ..... Ursache-Wirkungs-Analyse 38
4.1.4 ..... Zustandsbasierte Testfallermittlung 42
4.1.5 ..... Anwendungsfallbasierte Testfallermittlung 46
4.1.6 ..... Werkzeuge für die Testspezifikation 48
4.1.7 ..... Qualitätsanforderungen zu den Blackbox-Verfahren 49
4.1.8 ..... Empfehlungen zu den Blackbox-Verfahren 50
4.2 Whitebox-Verfahren 51
4.2.1 ..... Testabdeckungsgrade 51
4.2.2 ..... Weitere Testabdeckungsgrade 56
4.2.3 ..... Werkzeuge für die Testabdeckungsgradmessung 57
4.2.4 ..... Qualitätsanforderungen zu den Whitebox-Verfahren 58
4.2.5 ..... Empfehlungen zu den Whitebox-Verfahren 59
4.3 Fehlererwartung 60
4.4 Zusammenfassung 61

5 Risikoanalyse 63

5.1 Ziele der Risikoanalyse 63
5.2 Grundlagen der Risikoanalyse 64
5.3 Risikoanalyse in der Software-Entwicklung 64
5.4 Werkzeuge für die Risikoanalyse 68
5.5 Zusammenfassung 68

6 Checklisten 69

6.1 Ziele des Einsatzes von Checklisten 69
6.2 Werkzeuge für die Checklistenverwaltung 70
6.3 Empfehlungen zum Einsatz von Checklisten 71
6.4 Zusammenfassung 72

Teil II Testtypen 73

7 Prüfungen von Dokumenten 75

7.1 Dokumententest 75
7.1.1 ..... Formale Prüfung 77
7.1.2 ..... Review-Sitzung 80
7.1.3 ..... Schriftliche Stellungnahme 82
7.2 Spezielle Dokumententests 83
7.3 Werkzeuge für den Dokumententest 84
7.4 Qualitätsanforderungen zum Dokumententest 84
7.5 Empfehlungen zum Dokumententest 85
7.6 Zusammenfassung 86

8 Tests zur Funktionalität 87

8.1 Klassentest 88
8.1.1 ..... Klasseneinzel- und Klassenintegrationstest 88
8.1.2 ..... Abgrenzung und Empfehlung zum Klassentest 89
8.1.3 ..... Werkzeuge für den Klassentest 90
8.1.4 ..... Qualitätsanforderungen zum Klassentest 91
8.2 Komponententest 91
8.2.1 ..... Schritte des Komponententests 93
8.2.2 ..... Werkzeuge für den Komponententest 94
8.2.3 ..... Qualitätsanforderungen zum Komponententest 94
8.3 Integrationstest 94
8.3.1 ..... Strategien der Integration 95
8.3.2 ..... Integration externer Komponenten 100
8.3.3 ..... Werkzeuge für den Integrationstest 101
8.3.4 ..... Qualitätsanforderungen zum Integrationstest 101
8.3.5 ..... Empfehlungen zum Integrationstest 102
8.4 Funktionaler Systemtest 102
8.4.1 ..... Testszenarien zum funktionalen Systemtest 102
8.4.2 ..... Werkzeuge und Qualitätsanforderungen zum funktionalen Systemtest 103
8.5 Link-Test 104
8.5.1 ..... Link-Typen 104

8.5.2 Werkzeuge für den Link-Test 105
8.5.3 Qualitätsanforderungen zum Link-Test 106
8.5.4 Empfehlungen zum Link-Test 106
8.6 Cookie-Test 106
8.6.1 Überprüfung von Cookies 106
8.6.2 Werkzeuge für den Cookie-Test 108
8.6.3 Qualitätsanforderungen und Empfehlungen zum Cookie-Test 108
8.7 Plugin-Test 109
8.7.1 Szenarien zum Plugin-Test 109
8.7.2 Werkzeuge für den Plugin-Test 112
8.7.3 Qualitätsanforderungen zum Plugin-Test 112
8.7.4 Empfehlungen zum Plugin-Test 112
8.8 Sicherheitstest 113
8.8.1 Schritte des Sicherheitstests 114
8.8.2 Werkzeuge für den Sicherheitstest 119
8.8.3 Qualitätsanforderungen zum Sicherheitstest 119
8.8.4 Empfehlungen zum Sicherheitstest 120
8.9 Zusammenfassung 120

9 Tests zur Benutzbarkeit 123

9.1 Content-Test 123
9.1.1 Erfüllung der Benutzererwartungen 124
9.1.2 Einhaltung der Gesetze 127
9.1.3 Erfüllung der Aufklärungspflicht 128
9.1.4 Qualitätsanforderungen zum Content-Test 130
9.1.5 Empfehlungen zum Content-Test 130
9.2 Oberflächentest 131
9.2.1 Dialogrichtlinien und Standard-funktionalitäten 131
9.2.2 Werkzeuge für den Oberflächentest 136
9.2.3 Qualitätsanforderungen zum Oberflächentest 138
9.2.4 Empfehlungen zum Oberflächentest 138
9.3 Browser-Test 139
9.3.1 Auswahl der zu testenden Browser 139
9.3.2 Tests mit alten Browser-Versionen 143
9.3.3 Test der Browser-Einstellungen 144
9.3.4 Stichprobentest der Oberfläche 145
9.3.5 Werkzeuge für den Browser-Test 146

9.3.6..... Qualitätsanforderungen zum Browser-Test .. 147
9.3.7..... Empfehlungen zum Browser-Test 147
9.4 Usability-Test 148
9.4.1..... Usability-Labor 148
9.4.2..... Befragung 150
9.4.3..... Auswertung Usability-Labor und Befragungen 150
9.4.4..... Online-Umfrage 152
9.4.5..... Blickregistrierung 152
9.4.6..... Werkzeuge für den Usability-Test 152
9.4.7..... Qualitätsanforderungen zum Usability-Test . 154
9.4.8..... Empfehlungen zum Usability-Test 155
9.5 Zugänglichkeitstest 155
9.5.1..... Zugänglichkeitsanforderungen 156
9.5.2..... Werkzeuge für den Zugänglichkeitstest 157
9.5.3..... Qualitätsanforderungen zum Zugänglichkeitstest 159
9.5.4..... Empfehlungen zum Zugänglichkeitstest 159
9.6 Auffindbarkeitstest 160
9.6.1..... Namensgebung der Web-Adresse 160
9.6.2..... Suchmaschinenoptimierung 161
9.6.3..... Werkzeuge für den Auffindbarkeitstest 162
9.6.4..... Qualitätsanforderungen zum Auffindbarkeitstest 162
9.6.5..... Empfehlungen zum Auffindbarkeitstest 163
9.7 Zusammenfassung 163

10 Test zur Änderbarkeit und Übertragbarkeit 165

10.1 Code-Analysen 165
10.1.1... Code-Walkthrough 166
10.1.2... Code-Inspektion 167
10.1.3... Schreibtischtest 171
10.1.4... Statische Code-Analyse durch Werkzeuge ... 171
10.1.5... Qualitätsanforderungen zu Code-Analysen .. 172
10.1.6... Empfehlungen zu Code-Analysen 173
10.2 Installationstest 174
10.2.1... Installationsphasen 174
10.2.2... Werkzeuge für den Installationstest 177
10.2.3... Qualitätsanforderungen zum Installationstest 177
10.2.4... Empfehlungen zum Installationstest 177

10.3 Zusammenfassung 178

11 Tests zur Effizienz und Zuverlässigkeit 179

11.1 Performanz-/Lasttests 180
11.1.1 ... Performanztest 180
11.1.2 ... Lasttest 182
11.1.3 ... Skalierbarkeitstest 183
11.1.4 ... Speicherlecktest 184
11.1.5 ... Automatisierung der Performanz-/Lasttests 184
11.1.6 ... Metriken der Performanz-/Lasttests 186
11.1.7 ... Planung der Performanz-/Lasttests 190
11.1.8 ... Werkzeuge für Performanz-/Lasttests 192
11.1.9 ... Qualitätsanforderungen zu Performanz-/Lasttests 193
11.1.10 Empfehlungen zu Performanz-/Lasttests 194
11.2 Ausfallsicherheitstest 195
11.2.1 ... Redundanzen und Failover-Verfahren 195
11.2.2 ... Failover-Test 197
11.2.3 ... Failback-Test 197
11.2.4 ... Restart-/Recovery-Test 198
11.2.5 ... Werkzeuge für den Ausfallsicherheitstest 199
11.2.6 ... Qualitätsanforderungen zum Ausfallsicherheitstest 200
11.2.7 ... Empfehlungen zum Ausfallsicherheitstest 200
11.3 Verfügbarkeitstest 201
11.3.1 ... Definitionen zur Verfügbarkeit 201
11.3.2 ... Qualitätsanforderungen und Empfehlungen zum Verfügbarkeitstest 202
11.4 Zusammenfassung 202

Teil III Testmanagement 205

12 Testwiederholungen 207

12.1 Fehlernachtest 208
12.2 Regressionstest 208
12.2.1 ... Regressionstests nach Fehlerbehebung 209
12.2.2 ... Regressionstests nach Funktionsveränderungen 209
12.2.3 ... Regressionstests nach Systemveränderungen 210

12.3 Wartungstest 211
12.4 Werkzeuge für die Testwiederholung 212
12.4.1 ... Load Test Tools (Lasttestwerkzeuge) 212
12.4.2 ... Capture Replay Tools (Testroboter) 213
12.5 Empfehlungen zur Testwiederholung 215
12.6 Zusammenfassung 216

13 Planung der Testtypen 219

13.1 Bewertung der Testtypen 219
13.2 Bereitstellung der Testmittel 222
13.2.1 ... Bereitstellung von Checklisten 223
13.2.2 ... Bereitstellung der Testwerkzeuge 223
13.3 Planung des Testteams 226
13.4 Zusammenfassung 227

14 Planung der Teststufen 229

14.1 Teststufe Entwicklertest 230
14.2 Teststufe Komponententest 230
14.3 Teststufe Integrationstest 231
14.4 Teststufe Systemtest 232
14.5 Teststufe Abnahmetest 233
14.5.1 ... Abnahmephasen 233
14.5.2 ... Abnahmekriterien 234
14.5.3 ... Testtypen im Abnahmetest 234
14.6 Teststufe Betrieb 235
14.6.1 ... Pilotbetrieb 235
14.6.2 ... Qualitätsprüfungen im laufenden Betrieb 236
14.6.3 ... Planung der Testzyklen 237
14.7 Zusammenfassung 238

Anhang 239

A Anhang: Fragebögen zum Usability-Test 241

B Anhang: Checklisten BITV zum Zugänglichkeitstest ... 245

B.1 Checkliste BITV Priorität I 245

B.2 Checkliste BITV Priorität II 252

Abkürzungen 257

Synonyme 261

Glossar 263

Quellen 269

Literatur 269
Normen und Standards 270
URLs 270

Sachverzeichnis 277

1 Einleitung

„Eine Web-Anwendung oder Web-Applikation ist ein Computer-Programm, das auf einem Web-Server ausgeführt wird, wobei eine Interaktion mit dem Benutzer ausschließlich über einen Webbrowser erfolgt. Hierzu sind der Computer des Benutzers (Client) und der Server über ein Netzwerk, wie das Internet oder über ein Intranet miteinander verbunden, so dass die räumliche Entfernung zwischen Client und Server unerheblich ist.“

Zitat: http://de.wikipedia.org/wiki/Web-Anwendung

1.1 Wieso dieses Buch?

Eine Web-Anwendung muss vielfältigen technischen und fachlichen Anforderungen genügen. Eine nicht vorhersagbare Anzahl von unbekannten Anwendern mit unterschiedlichem Fach- und Technikwissen greift weltweit mit diversen Browsern und Betriebssystemen auf eine Web-Anwendung zu. Die Qualitätssicherung einer Web-Anwendung ist daher eine komplexe Aufgabe und muss professionell durchgeführt werden.

Der Einsatz der klassischen Testmethoden aus der Welt der Großrechneranwendungen, Transaktionsmonitore und Client/Server-Systeme ist auch für web-basierte Anwendungen notwendig, aber nicht hinreichend, um die geforderte Qualität sicherzustellen.

Die bekannten und erprobten Testmethoden müssen an die neuen Technologien angepasst und um neue Verfahren ergänzt werden.

Genau dieses Ziel verfolgt das vorliegende Handbuch zum Webtesting. Es soll Testern und Qualitätsverantwortlichen eine umfassende, aber kompakte Arbeitshilfe zum Testen von Web-Anwendungen geben. Neben der Beschreibung der verschiedenen Testtypen werden Hilfestellungen in Form von Tipps und Checklisten gegeben und die Einsatzmöglichkeiten von Testtools erläutert.

1.2 Wem nutzt dieses Buch wie?

Dieses Buch richtet sich an Personen, die mit der Qualitätssicherung web-basierter Anwendungen befasst sind. Dabei kann es sich um informative Websites bis hin zu komplexen Web-Applikationen zur Abwicklung von Geschäften über das Internet handeln. Somit ist das Buch für Qualitätsmanager, Testmanager und Tester sowie qualitätsbewusste Entwickler, Projektleiter und Webmaster gleichermaßen interessant. Sie sollen mit dem Lesen dieses Buches folgende Lernziele erreichen:

Lernziele

- Sie können die Qualitätsmerkmale, die eine Web-Anwendung erfüllen sollte, beschreiben.
- Sie können Qualitätsanforderungen für ihre Web-Anwendung definieren.
- Sie können systematisch Testfälle für die Tests ihrer Web-Anwendung entwerfen.
- Sie können die zur Qualitätssicherung ihrer Web-Anwendung notwendigen Testverfahren festlegen, priorisieren und anwenden.
- Sie wissen, welche Testtools bei welchen Tests eingesetzt werden können, und kennen deren Funktionsweise.
- Sie können im Rahmen ihrer Qualitätssicherungsmaßnahmen Risikoanalysen durchführen und Checklisten effektiv einsetzen.
- Sie wissen, wo weiterführende und nützliche Informationsquellen zum Testen von Web-Anwendungen zu finden sind.
- Sie wissen, welche Faktoren bei der Planung von Tests berücksichtigt werden müssen.

Einige der in diesem Buch beschriebenen Testtypen, wie zum Beispiel Dokumententest und Performanztest, sind nicht spezifisch für Web-Anwendungen. Daher ist das Buch auch für Leser nützlich, die

für die Qualitätssicherung von Software, die nicht im Web-Umfeld eingesetzt wird, verantwortlich sind.

1.3 Wie ist dieses Buch zu lesen?

Dieses Buch konzentriert sich auf die Qualitätssicherungsmaßnahmen, die sich direkt auf die Qualität einer Web-Anwendung auswirken. Der Leser soll mit diesem Buch eine schnelle, effektive Unterstützung für seine durchzuführenden Tests erhalten.

Im ersten Teil des Buches werden die Grundlagen der Qualitätssicherung vermittelt. Im zweiten Teil, der sich insbesondere an die Praktiker richtet, werden Vorgehensweisen und Werkzeuge zum Testen beschrieben. Im dritten Teil werden Maßnahmen zur Planung und Steuerung der Tests aus Sicht des Testmanagers betrachtet.

Teil I: Handwerkszeug

für Interessierte

Wer die Begriffe aus der Welt der IT-Qualitätssicherung noch nicht beherrscht, der darf den ersten Teil dieses Buches nicht vernachlässigen, denn darin wird das Handwerkszeug eines Testers bereitgestellt. Dazu gehören die Normen zur Qualitätssicherung, die Definitionen zum Testen, die Testfallentwurfsmethoden, die Risikoanalyse und der Einsatz von Checklisten.

Teil II: Testtypen

für Praktiker

Im zweiten Teil des Buches werden die einzelnen Testtypen, die zur Qualitätssicherung von Web-Anwendungen eingesetzt werden, detailliert beschrieben. Zu jedem Testtyp werden wertvolle Tipps zum Vorgehen und zum Einsatz von Testwerkzeugen gegeben sowie Checklisten für die Praxis bereitgestellt. Mit diesen Informationen wird der Leser in die Lage versetzt, seine Tests zielgerichtet vorbereiten und effektiv durchführen zu können.

Wer das Handwerkszeug der Qualitätssicherung parat hat und sofort mit den Testtypen starten möchte, um seine Web-Anwendung zu testen, kann den ersten Teil des Buches überspringen und mit dem zweiten Teil beginnen.

Die Tabelle 1.1 zeigt eine Übersicht der im Teil II beschriebenen Testtypen. Darin sind in der Spalte „Checkliste“ die Checklisten aufgeführt, die in diesem Buch zum jeweiligen Testtyp beschrieben sind. In der Spalte „Nur Web?“ ist vermerkt, ob ein Testtyp nur für den Test von Web-Anwendungen relevant ist. Mit „Nein“ gekennzeichnete Testtypen kommen für jede Art von Software zum Tragen, zum Beispiel für Client-/Server-Applikationen.

Tabelle 1.1: Testtypen und Checklisten

Testtyp	Checkliste	Nur Web?
Dokumententest	Dokument, Datenmodell	Nein
Tests zur Funktionalität		
Klassentest		Nein
Komponententest		Nein
Integrationstest		Nein
Funktionaler Systemtest		Nein
Link-Test	Link-Test	Ja
Cookie-Test	Cookie-Test	Ja
Plugin-Test	Plugin-Test	Ja
Sicherheitstest	Sicherheitstest, Penetrationstest (extern), Firewall (extern)	Nein[1]
Tests zur Benutzbarkeit		
Content-Test	Web-Content, Web-Recht, Aufklärungspflicht	Ja
Oberflächentest	Oberflächentest	Nein
Browser-Test	Browser-Einstellungen, Stichprobe Browser-Test	Ja
Usability-Test	Fragebogen Nutzungsverhalten Bewertungsbogen Usability	Nein
Zugänglichkeitstest	Zugänglichkeit nach WAI, Zugänglichkeit nach BITV (beide extern)	Ja
Auffindbarkeitstest	Web-Adresse	Ja
Tests zur Änderbarkeit und Übertragbarkeit		
Code-Analysen	Code-Inspektion	Nein
Installationstest	Installationstest	Nein
Tests zur Effizienz und Zuverlässigkeit		
Performanz-/Lasttests	Planung Performanz-/Lasttest	Nein
Ausfallsicherheitstest	Ausfallsicherheitstest	Nein
Verfügbarkeitstest		Nein

[1] Die im Buch beschriebenen Verfahren und Checklisten zum Sicherheitstest sind allerdings sehr web-spezifisch.

Teil III: Testmanagement

für Manager

Im dritten Teil des Buches werden die im Teil II beschriebenen Testtypen aus Sicht des Testmanagers betrachtet, der die anstehenden Qualitätssicherungsmaßnahmen planen und steuern muss. Hier wird beschrieben, wie welche Teststufen, Testtypen, Testmittel und Testressourcen im Rahmen der Testplanung auszuwählen, zu bewerten und einzusetzen sind.

Personen, die einen Überblick zum Testen von Web-Applikationen erhalten möchten, können sich diesen im Teil III verschaffen und bei Bedarf die Details in den Kapiteln der ersten beiden Teile dieses Buches nachschlagen.

Zum Glossar und Quellenverzeichnis

Fachbegriffe, die nicht im Text des Buches erklärt werden, aber eventuell nicht jedem Leser geläufig sind, sind im Text mit [GL] gekennzeichnet und im Glossar aufgenommen.

Quellenangaben, die sich im Quellenverzeichnis wiederfinden, sind im Text in rechteckige Klammern [...] gefasst. Dabei handelt es sich um Literaturquellen, Normen und Standards sowie Internet-Links. Letztere besitzen das Präfix URL ([URL: ...]).

1.4 Welche Testwerkzeuge werden genannt?

Für fast jeden in diesem Buch beschriebenen Testtyp werden Testwerkzeuge benannt. Dabei handelt es sich um Werkzeuge, die ich aus meiner Projektarbeit kenne. Das kann bedeuten, dass Werkzeuge, die vielleicht für die jeweils beschriebene Aufgabe ebenso so gut geeignet sind, nicht namentlich aufgeführt werden.

Umfassende und aktuelle Übersichten von frei verfügbaren und kommerziellen Testwerkzeugen sind zu finden unter:

- [URL: TestToolsFAQ]
- [URL: TestToolsJava]
- [URL: TestToolsOPENSOURCE]
- [URL: TestToolsSQA]

1.5 Was liefert dieses Buch nicht?

Aussagen wie *„Spät gefundene Fehler sind die teuersten."* sind richtig, aber bekannt, schon häufig diskutiert und ausführlich beschrieben. Daher werde ich auf Beweisführungen, warum und wieso Software rechtzeitig und intensiv getestet werden muss, in diesem Buch verzichten.

Auf Vorgehensmodelle (wie Wasserfallmodell und V-Modell), Prozessverbesserungsmodelle (wie CMMI, SPICE) und spezielle Themen zum Testmanagement (wie Konfigurations- und Fehlermanagement) gehe ich nur so weit ein, wie es zum Verständnis der beschriebenen Inhalte notwendig ist. Dass diese Themen hier etwas stiefmütterlich behandelt werden, liegt nicht daran, dass ich sie für unwichtig halte. Im Gegenteil, jeder, dem Software-Qualität am Herzen liegt, muss sich diese Themen auf die Fahnen schreiben. Denn Qualität von Software kann nur erzeugt werden, wenn alle Software-Produktionsprozesse geplant und gesteuert werden, ineinander greifen und einer permanenten Verbesserung unterliegen.

Informationen

Weil diese Themen aber den Rahmen dieses Buches sprengen würden, verweise ich an dieser Stelle auf weiterführende Literatur (V-Model - [URL: V-Modell]), CMMI - [Kneuper_2003], SPICE - [Hörmann_2006], Testmanagement - [Spillner_2005] und [Spillner_2006]).

1.6 Wer sollte das Buch unbedingt lesen? Oder: Motivation für autofahrende Qualitätsskeptiker

Bei der Entwicklung und Einführung neuer Kraftfahrzeuge sind intensive Qualitätssicherungsmaßnahmen selbstverständlich, in der Software-Entwicklung leider immer noch nicht.

für QS-Skeptiker

Nicht selten sind Entwickler und IT-Manager anzutreffen, die das Testen von Software für Zeit- und Geldverschwendung halten. Diesen Personen ist der folgende Vergleich gewidmet, in der Hoffnung, dass sie mehr Sensibilität gegenüber der Qualitätssicherung im IT-Projektgeschäft entwickeln. Wie würden sie reagieren, wenn ihr neues Auto ungeprüft ausgeliefert würde, sie die meisten Mängel erst selbst beim Fahren feststellen würden und auf Grund dessen Dauerkunde in ihrer KFZ-Werkstatt wären?

Eine Qualitätssicherungsanalogie

Ein neues PKW-Modell mit dem Arbeitstitel „WaF-Brauser“ (WaF = Work and Family) soll den Automobilmarkt bereichern. Das Management entscheidet, dass ein sportlicher Kleintransporter mit Schiebedach eine Marktlücke schließen soll.

Der neue, etwas luxuriöse WaF-Brauser soll für kleine Firmen interessant sein, deren Inhaber ihr Auto sowohl geschäftlich als auch privat nutzen.

Die Anforderungen an den WaF-Brauser werden zusammengetragen. Dazu gehören:

- Mindestens fünf Sitze
- Große Ladefläche
- Dieselmotor mit geringem Verbrauch
- Hoher Sicherheitsstandard

Mit diesen Angaben werden die Konstrukteure konfrontiert. Diese sind nicht glücklich damit und stellen viele Fragen.

→ Review

- Was heißt „mindestens fünf Sitze“? Können es auch sechs, sieben oder neun Sitze sein?
- Muss die „große Ladefläche“ und die Anzahl von fünf Sitzen gleichzeitig möglich sein oder kann die „große Ladefläche“ auf Kosten ausgebauter Sitze gehen?
- Was genau ist eigentlich eine „große Ladefläche“?
- Und was ist für Sie „flotte Beschleunigung“ mit „geringem Benzinverbrauch“?
- Hoher Sicherheitsstandard bedeutet was?

Die Fragen zeigen die Problematik für die Konstrukteure. Es sind zwar Merkmale genannt worden, die der neue WaF-Brauser erfüllen soll, aber keine konkreten Vorgaben. Um konkrete Konstruktionspläne erstellen zu können, werden detailliertere Informationen benötigt.

→ Qualitätsmerkmale

Die Auftraggeber gehen also in Klausur und machen sich Gedanken über die Details. Konstrukteure und Designer werden in die Arbeitsgruppe mit aufgenommen. Als Ergebnis sind die Anforderungen nun mit eindeutig messbaren Größen beschrieben:

→ Qualitätsanforderungen

- Der Laderaum muss eine Höhe von mindestens 1,85 m haben.

- Bei einer freien Ladefläche von mindestens 2,20 m x 1,50 m müssen 6 Personen transportiert werden können.
- Bei einer um 80% vergrößerbaren Ladefläche müssen noch 3 Personen transportiert werden können.
- Der durchschnittliche Kraftstoffverbrauch ohne Beladung soll bei maximal 7,5 Litern Dieselkraftstoff pro 100 Kilometer liegen.
- Die standardisierten Crash-Tests müssen den Nachweis der höchsten Sicherheitsstufe (fünf Sterne) erbringen.

→ *Dokumententest*

In der nächsten Entwicklungsphase werden die Baupläne konstruiert, das Design wird entworfen. Natürlich werden diese Dokumente und Pläne mehreren Prüfungen durch Experten unterzogen.

→ *Usability-Test*

Bevor der WaF-Brauser in Produktion geht, wird ein Prototyp entwickelt und potentiellen Kunden vorgeführt. Diese sollen zum Beispiel folgende Fragen beantworten:

- Sind die Sitze bequem?
- Ist die Ladefläche leicht zu beladen?
- Ist das Cockpit für kleine Personen überschaubar?
- Erlaubt die Öffnung der Heckklappe das Beladen mit großen Gegenständen?
- Sind alle Knöpfe und Schalter ergonomisch angeordnet?

Die Testpersonen dürfen Probe fahren und werden anschließend über ihre Meinungen und Erwartungen befragt. Es ergeben sich neue Anforderungen:

- Bis zu 8 Sitzplätze sollen verfügbar sein.
- Dachreling und Navigationssystem sollen Standard sein.
- Standheizung soll nur auf Wunsch ausgeliefert werden und nachrüstbar sein.
- Einen Kühlschrank braucht niemand.
- Große Schiebetüren werden seitlich hinten gefordert.

Nachdem alle Anforderungen erneut beschrieben, geprüft und besiegelt sind, kann der WaF-Brauser produziert werden.

Ein Auto besteht aus einer großen Anzahl von Bausätzen (Komponenten). Von jedem internen und externen Lieferanten wird ge-

fordert, dass er seine Komponenten nachweisbar qualitätsgesichert hat.

Sowohl für jedes Einzelteil eines Bausatzes (z.B. Bremsschlauch) als auch für das Zusammenspiel der Einzelteile (die Bremse als Ganzes) muss eine Kontrolle nachgewiesen werden. → *Komponententest*

Das gilt auch für die Verträglichkeit (Bremsflüssigkeit) und Kommunikationsfähigkeit mit anderen Komponenten (z.B. ABS). → *Integrationstest*

Bei den Prüfungen stellt sich heraus, dass Mikroprozessoren eines bestimmten Typs fehlerhaft arbeiten. Nach Lieferung neuer Prozessoren werden alle Komponenten, in die sie integriert sind, neuen Prüfungen unterzogen. → *Regressionstest*

Die Komponenten werden nun sukzessive zusammengesetzt und nach jeder Phase überprüft, bis am Ende ein WaF-Brauser fertiggestellt ist. Einige Exemplare werden von Werksfahrern auf kurzen Strecken und unter normalen Bedingungen Probe gefahren. Dabei werden alle möglichen Funktionen ausprobiert. → *Systemtest*

Anschließend dürfen einige gute Kunden den WaF-Brauser im täglichen Gebrauch fahren und bewerten. Aufgrund der bisher hervorragenden Qualitätssicherungsmaßnahmen treten nur kleine Mängel zu Tage, die schnell im Produktionsprozess eliminiert werden können. → *Pilotphase*

Bevor der WaF-Brauser in großer Stückzahl in Serie gehen kann, sind noch einige Tests durchzuführen. Das Fahrzeug muss auf längeren Strecken beweisen, dass Beschleunigung, Kraftstoffverbrauch, Fahrgeräusche und Sitzkomfort den Qualitätsanforderungen entsprechen. → *Performanztest*

Weiterhin muss sichergestellt sein, dass sich das Fahrzeug in extremen Situationen erwartungsgemäß verhält und die Sicherheit der Insassen gewährleistet:

- Der Wagen läuft auch bei Höchstgeschwindigkeiten ruhig. → *Lasttest*
- Die Beschleunigung ist in kritischen Situationen hinreichend.
- Der Öl- und Kraftstoffverbrauch überschreitet auf längeren Schnellfahrten nicht die angegebenen Werte.
- Straßenlage und Bremsverhalten entsprechen bei hohem Tempo und bei voller Beladung den Vorgaben (Elchtest).
- Es gehen keine Betriebsstoffe (Sprit, Öl, Wasser, Bremsflüssigkeit) verloren. → *Speicherlecktest*
- Wenn wichtige Komponenten kurz vor dem Versagen stehen oder ausgefallen sind, wird der Fahrer direkt informiert (ABS, ESP, Öldruck,...). → *Ausfallsicherheitstest*

→ *Sicherheitstest*

- Bei einer Vollbremsung ohne Aufprall wird der Airbag nicht ausgelöst, sondern erst beim Auftreffen auf ein Hindernis ab der vorgegebenen Kraft.
- Der Crash-Test eines unabhängigen Gutachterinstituts erbringt die Sicherheitsstufe vier. Die Konstruktion der Fahrerkabine wird nachgebessert. Erst danach werden die fünf Sterne für höchste Sicherheit vergeben.
- Die Wegfahrsperre und die Diebstahlsicherung entsprechen den Normen.

→ *Zugänglichkeitstest*

Wenn der WaF-Brauser für Rollstuhlfahrer geeignet sein soll, müssen dazu die gesetzlichen Vorschriften eingehalten und überprüft werden.

→ *Plugin-Test*

Wie steht es mit den Sonderausstattungen? Dachgepäckträger, Heckträgersysteme für Fahrräder, Anhängerkupplungen und ähnliche Ergänzungen dürfen Fahrkomfort und Sicherheit nicht beeinträchtigen.

→ *Abnahmetest*

Wird der Wagen erstmals für den Straßenverkehr zugelassen oder werden nachträglich zusätzliche Teile am Wagen eingebaut, ist eine TÜV-Abnahme erforderlich.

→ *Wartungstest*

Wenn der WaF-Brauser in Betrieb ist und gefahren wird, muss er regelmäßig zum TÜV.

→ *Verfügbarkeitstest*

In der Zentrale des WaF-Herstellers werden Statistiken erhoben, wie oft ein Fahrzeugtyp mit Mängeln in der Werkstatt ist und wie lang die Reparaturzeiten sind, um eventuell Verbesserungen für die nächste Version einzuplanen oder um notfalls Rückrufaktionen zu starten.

Wenn Autos so nachlässig getestet würden,
wie manche (oder viele?) Software-Anwendungen,
würde die Menschheit zu Fuß gehen.

Teil I
Handwerkszeug

2. Definitionen zur Qualität
3. Begriffe zum Testen
4. Testfallentwurfsverfahren
5. Risikoanalyse
6. Checklisten

Im ersten Teil des Buches werden die Grundlagen zur Qualitätssicherung von Software vermittelt. Dazu gehören die Definitionen zur Qualität und zum Testen, die Verfahren zum Entwurf von Testfällen, die Risikoanalyse und der Einsatz von Checklisten. Sie sind das methodische Handwerkszeug eines Testers und werden für das Verständnis der Teile II und III dieses Buches benötigt.

Testwerkzeuge gehören zum technischen Handwerkszeug des Testers und unterstützen ihn bei speziellen Testaufgaben. Daher werden Testwerkzeuge bei jedem im Teil II dieses Buches beschriebenen Testtyp aufgeführt und nicht in einem eigenständigen Kapitel.

2 Definitionen zur Qualität

„Qualität ist das Gegenteil des Zufalls.“

Klaus Zumwinkel (*1943)

In diesem Kapitel werden Normen zur Software-Qualität beschrieben, welche die Grundlage für die Qualitätssicherung von (nicht nur) web-basierten Applikationen darstellen.

Es gibt vier wichtige Fragen zur Qualitätssicherung von Web-Anwendungen, die in den folgenden Kapiteln beantwortet werden:

1. Was ist Qualität von Software im allgemeinen?
2. Was ist Qualität für Web-Anwendungen im Besonderen?
3. Wie kann Qualität gemessen werden?
4. Wie kann Qualität erzeugt werden?

2.1 Normen und Qualitätsmerkmale

Die Qualität von Software wird in mehreren Normen festgelegt. Die für den Inhalt dieses Buches relevanten Normbegriffe werden im Folgenden vorgestellt.[1]

[1] Die vollständigen Normtexte sind beim Beuth-Verlag ([URL: Beuth]) erhältlich.

Software-Qualität nach ISO/IEC 9126

Die ISO/IEC 9126 (Software-Engineering - Qualität von Software-Produkten, [ISO/IEC 9126]) legt Begriffe aus dem Bereich Software-Entwicklung fest und definiert Software-Qualität folgenderweise:

„Software-Qualität ist die Gesamtheit von Eigenschaften und Merkmalen eines Software-Produkts oder einer Tätigkeit, die sich auf deren Eignung zur Erfüllung gegebener Erfordernisse bezieht".

Qualitätsmerkmale nach DIN 66272

Insbesondere legt die 1994 aus der ISO/IEC 9126 hervorgegangene DIN 66272 (Bewerten von Softwareprodukten – Qualitätsmerkmale und Leitfaden zu ihrer Verwendung, [DIN 66272]) die sechs Hauptqualitätsmerkmale Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit einer Software-Anwendung fest:

Funktionalität:

Vorhandensein von Funktionen, die festgelegte und vorausgesetzte Erfordernisse erfüllen.
Teilmerkmale: Angemessenheit, Interoperabilität[GL], Ordnungsmäßigkeit, Richtigkeit, Sicherheit

Zuverlässigkeit:

Fähigkeit einer Software/eines Systems, ihr/sein Leistungsniveau unter festgelegten Bedingungen in einem festgelegten Zeitraum zu halten.
Teilmerkmale: Fehlertoleranz, Reife, Wiederherstellbarkeit

Benutzbarkeit:

Fähigkeit einer Software, unter festgelegten Bedingungen für einen Benutzer verständlich, erlernbar, anwendbar und attraktiv zu sein.
Teilmerkmale: Bedienbarkeit, Erlernbarkeit, Verständlichkeit

Effizienz:

Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen.
Teilmerkmale: Verbrauchsverhalten, Zeitverhalten (Performanz)

Änderbarkeit[2]:

Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist. Änderungen können Korrekturen von Fehlern, Anpassungen

[2] Änderbarkeit und Wartbarkeit werden synonym verwendet.

an veränderte Anforderungen oder eine veränderte Systemumgebung und die Verbesserung der Wartung einschließen.
Teilmerkmale: Analysierbarkeit, Modifizierbarkeit, Prüfbarkeit, Stabilität

Übertragbarkeit[3]:
Leichtigkeit, mit der eine Software von einer Umgebung in eine andere übertragen werden kann. Umgebung kann die organisatorische Umgebung, die Hardware-Umgebung oder die Software-Umgebung bedeuten.
Teilmerkmale: Anpassbarkeit, Austauschbarkeit, Installierbarkeit, Konformität

Nach diesen sechs Qualitätsmerkmalen sind die Kapitel des zweiten Teiles dieses Buches gegliedert, in denen die Tests zur Überprüfung eben dieser Merkmale beschrieben werden.

Hinweis

Ergonomie nach ISO 9241

Neben der ISO/IEC 9126 definiert die DIN EN ISO 9241 („Ergonomische Anforderungen an Bürotätigkeiten mit Bildschirmgeräten“) Software-Qualitätsmerkmale. Die Norm besteht aus 17 Teilen von denen für Tester von Web-Applikationen die beiden Teile 10 „Grundsätze der Dialoggestaltung“ und 11 „Anforderungen an die Gebrauchstauglichkeit (Allgemeine Leitsätze)“ relevant sind.

ISO 9241-10: Grundsätze der Dialoggestaltung

Wenn es darum geht, welche Inhalte auf einer Website erscheinen sollen, wie die Kundenzufriedenheit sichergestellt und die Anwendung optimal bedient werden kann, ist die ISO 9241 Teil 10 hilfreich ([ISO 9241-10]). Sie belegt die „Grundsätze der Dialoggestaltung“ mit folgenden Merkmalen (Zitate):

Aufgabenangemessenheit:
„Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen.“

Erwartungskonformität:
„Ein Dialog ist erwartungskonform, wenn er konsistent ist und den Merkmalen des Benutzers entspricht, zum Beispiel seinen Kenntnissen aus dem Arbeitsgebiet, seiner Ausbildung und seiner Erfahrung sowie den allgemein anerkannten Konventionen.“

[3] Übertragbarkeit und Portabilität werden synonym verwendet.

Fehlertoleranz:

„Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben mit keinem oder mit minimalem Korrekturaufwand durch den Benutzer erreicht werden kann."

Individualisierbarkeit:

„Ein Dialog ist individualisierbar, wenn das Dialogsystem Anpassungen an die Erfordernisse der Arbeitsaufgabe sowie an die individuellen Fähigkeiten und Vorlieben des Benutzers zulässt."

Lernförderlichkeit:

„Ein Dialog ist lernförderlich, wenn er den Benutzer beim Erlernen des Dialogsystems unterstützt und anleitet."

Selbstbeschreibungsfähigkeit:

„Ein Dialog ist selbstbeschreibungsfähig, wenn jeder einzelne Dialogschritt durch Rückmeldung des Dialogsystems unmittelbar verständlich ist oder dem Benutzer auf Anfrage erklärt wird."

Steuerbarkeit:

„Ein Dialog ist steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist."

Eine Web-Anwendung ist zum Beispiel

- aufgabenangemessen, wenn keine aufgabenfremden, unnötigen Eingaben gefordert werden (z. B. irrelevante, persönliche Angaben zu einer Bestellung),
- erwartungskonform, wenn unterstrichene Wörter stets Hyperlinks sind,
- fehlertolerant, wenn die Eingaben in ein Formular vor dem Abschicken überprüft werden und nur die fehlerhaften Eingaben erneut erfasst werden müssen,
- individualisierbar, wenn der Nutzer die Inhalte für seinen persönlichen Newsletter bestimmen und sein persönliches Benutzerprofil speichern kann,
- lernförderlich, wenn eine Guided Tour durch die Web-Applikation existiert und Testbuchungen vorgenommen werden können,

- selbstbeschreibungsfähig, wenn der Status der Bearbeitung am unteren Bildrand und die Anzahl der Treffer nach einer Suche angezeigt werden,
- steuerbar, wenn für Suchvorgänge die Anzahl der anzuzeigenden Treffer pro Seite festgelegt und Download-Prozesse unterbrochen werden können.

ISO 9241-11: Gebrauchstauglichkeit

Die Gebrauchstauglichkeit (Usability) einer Anwendung wird in der ISO 9241 - Teil 11 ([ISO 9241-11]) definiert (Zitat):

Gebrauchstauglichkeit:
„Das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen."

Und weiter erklärt die Norm

- Effektivität als die Genauigkeit und Vollständigkeit, mit der Benutzer ein bestimmtes Ziel erreichen, *Effektivität*
- Effizienz als den im Verhältnis zur Genauigkeit und Vollständigkeit eingesetzten Aufwand, mit dem Benutzer ein bestimmtes Ziel erreichen, *Effizienz*
- Zufriedenstellung als die Freiheit von Beeinträchtigungen und positive Einstellungen gegenüber der Nutzung des Produkts. *Zufriedenstellung*

Die Anforderungen an die Dialoggestaltung einer Software sind wie die Gebrauchstauglichkeit dem Qualitätsmerkmal Benutzbarkeit zuzuordnen. Ihnen wird insbesondere im Oberflächentest (Kap. 9.2) und im Usability-Test (Kap. 9.4) Rechnung getragen.

2.2 Qualitätsmerkmale für Web-Applikationen

Die Erfüllung der im vorangehenden Kapitel beschriebenen Qualitätsmerkmale wird von jeder Software erwartet, sei es im Großrechner-, im Client-/Server- oder im Web-Umfeld. Eine web-basierte Anwendung muss jedoch zusätzlichen Anforderungen genügen. Das sind, um auf die zweite Eingangsfrage nach besonderen Merkmalen web-basierter Software zu kommen, Auffindbarkeit, Barrierefreiheit und Rechtskonformität. Diese Qualitätsmerkmale werden der in der DIN 66272 geforderten Benutzbarkeit zugeordnet und dementsprechend durch die im Kapitel 9 beschriebenen Testtypen geprüft.

Auffindbarkeit:

Der Benutzer muss die Anwendung finden.

Eine Website muss leicht auffindbar sein. Das bedeutet zum einen, dass die Web-Adresse einen sprechenden, leicht zu merkenden Namen hat und zum anderen, dass sie von Suchmaschinen unter den Top-Treffern angezeigt wird.

Der Auffindbarkeitstest (Kap. 9.6) stellt die Auffindbarkeit eines Web-Angebotes sicher.

Barrierefreiheit (Accessibility):

Soll eine Web-Anwendung auch für behinderte Personen gebrauchstauglich sein, so muss sie barrierefrei realisiert werden. Barrierefreiheit nach §4 des Behindertengleichstellungsgesetzes (BGG) ist wie folgt definiert (Zitat aus [URL: BGG]):

„Barrierefrei sind bauliche und sonstige Anlagen, Verkehrsmittel, technische Gebrauchsgegenstände, Systeme der Informationsverarbeitung, akustische und visuelle Informationsquellen und Kommunikationseinrichtungen sowie andere gestaltete Lebensbereiche, wenn sie für behinderte Menschen in der allgemein üblichen Weise, ohne besondere Erschwernis und grundsätzlich ohne fremde Hilfe zugänglich und nutzbar sind."

Der Benutzer muss die Anwendung erreichen.

Barrierefreiheit (Accessibility) entspricht somit dem in der ISO 9241 definierten Qualitätsmerkmal Gebrauchstauglichkeit (Usability) für einen Benutzerkreis, der behinderte Menschen mit einschließt. Diese spezielle Ausprägung des geforderten Qualitätsmerkmals der Benutzbarkeit einer Website wird im Zugänglichkeitstest (Kap. 9.5) getestet.

Rechtskonformität:

Der Web-Auftritt muss Rechtens sein.

Die weltweit einsehbaren Inhalte von Websites müssen rechtlich „wasserdicht" sein. Eine Website darf zum Beispiel nicht gegen Urheberrechte verstoßen oder verbotene Links einsetzen. Eine Website ist rechtskonform, wenn die dargestellten und verlinkten Inhalte gegen keine gesetzlichen Vorschriften verstoßen.

Die Rechtskonformität einer Website wird im Content-Test (Kap. 9.1) geprüft.

2.3 Qualitätsanforderungen

Kommen wir zur dritten Frage, die anfangs gestellt wurde: Wie kann Qualität gemessen werden? In den beiden vorherigen Abschnitten wurden Qualitätsmerkmale von Web-Anwendungen beschrieben.

Aber ohne weitere Maßangaben kann die Erfüllung eines Qualitätsmerkmals wie zum Beispiel Benutzerfreundlichkeit oder Performanz weder positiv noch negativ beschieden werden.

↓

Exkurs: Qualitätsmerkmale eines KFZs

Das Problem ist vergleichbar mit der Anforderung, dass ein neues Auto, dass Sie kaufen wollen, wenig Sprit verbrauchen soll. Diese Aussage wird Ihnen jeder Autoverkäufer sofort als besonderes Qualitätsmerkmal aller seiner Fahrzeuge bestätigen. Aber sie beschreibt nur ein subjektives Merkmal des Fahrzeugs, aus der keine Konsequenzen oder Maßnahmen abgeleitet werden können.

Erst die konkrete Vorgabe, dass ein Auto im Durchschnitt höchstens 6,5 Liter Kraftstoff auf 100 gefahrene Kilometer verbrauchen darf, hilft weiter. Aus dem Merkmal „Geringer Spritverbrauch“ wird so eine messbare Anforderung, deren Nichterfüllung/Erfüllung durch eine konkrete Verbrauchsmessung nachgewiesen werden kann.

↑

In der Software-Entwicklung verhält es sich so, wie bei dem Auto mit geringem Spritverbrauch. Was bedeutet zum Beispiel die Anforderung, dass die Web-Applikation „ohne“ Wartezeiten reagieren soll? Was hilft dem Tester die Forderung, dass die Anwendung performant sein soll?

Eine qualifizierte Aussage zum Qualitätsmerkmal Performanz kann erst mit einer Maßangabe getroffen werden, wie zum Beispiel: „Der Download eines 500 KB großen PDF-Dokumentes muss bei einer 128-KB ISDN-Verbindung in spätestens 7 Sekunden abgeschlossen sein.“

Diese Aussage stellt eine messbare Qualitätsanforderung dar, deren Abweichung oder – hoffentlich – Nichtabweichung durch einen Test konkret in Sekunden angegeben werden kann.

> Eine Qualitätsanforderung besteht aus
> Qualitätsmerkmal und Qualitätsmaß!

In Software-Entwicklungsprojekten ist es nicht beliebt, Qualitätsmerkmale eines Anwendungssystems mit konkreten Maßangaben zu belegen. Zum einen bedeutet es Aufwand, sich frühzeitig über fachliche und technische Anforderungen detaillierte Gedanken zu machen und dann auch noch festzuschreiben, wie sie zu messen sind. Zum anderen sind dokumentierte Qualitätsanforderungen verbindli-

che Entscheidungen, mit denen Auftraggeber und Projektleitung in die Verantwortung genommen werden können.

Spätestens zu dem Zeitpunkt, an dem ein IT-Projekt in Terminnöte gerät – und welches IT-Projekt tut das nicht? – , wird das Management versuchen, beim Testen Aufwand zu sparen. Und weniger Tests kann bedeuten, dass die ursprünglich gestellten Qualitätsanforderungen nicht eingehalten werden können. Schlechtere Qualität wird geliefert. In solchen Fällen müssen die durch die Sparmaßnahmen entstehenden Konsequenzen und Risiken (s. Kap. 5, Risikoanalyse) bedacht werden. Das gelingt aber nur, wenn zum Projektbeginn die Anforderungen an das zu realisierende Anwendungssystem detailliert und verbindlich festgelegt worden sind.

Aus diesem Grund und um die Qualität einer Anwendung nachweisen zu können, muss der Auftraggeber im Projektauftrag seine Qualitätsanforderungen auch – oder gerade besonders – für alle nicht-funktionalen Qualitätsmerkmale eindeutig festlegen.

Der Auftraggeber legt die Qualitätsanforderungen fest.

nicht-funktionale Qualitätsmerkmale

Nicht-funktionale Qualitätsmerkmale sind diejenigen Merkmale, die sich nicht direkt auf die geforderte Funktionalität beziehen. Dazu gehören Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit sowie für Web-Anwendungen Auffindbarkeit, Barrierefreiheit und Rechtskonformität.

Qualitätsmerkmale priorisieren

Bei der Festlegung der Qualitätsanforderungen an eine Software muss berücksichtigt werden, dass sich Qualitätsmerkmale widersprechen können. Zum Beispiel erfordert hohe Benutzerfreundlichkeit in der Regel eine Vermehrung des Programmcodes und mindert damit die Effizienz. Hohe Effizienz geht zu Lasten der Änderbarkeit und der Übertragbarkeit eines Programms, weil der Quellcode optimiert werden muss und somit nicht mehr strukturiert und einfach lesbar ist. Aus diesem Grund müssen die Qualitätsmerkmale vom Auftraggeber priorisiert werden.

2.4 Qualitätssicherungsmaßnahmen

In der Qualitätssicherung werden konstruktive und analytische Qualitätssicherungsmaßnahmen unterschieden.

konstruktive QS-Maßnahmen

Weil Qualität nicht im nachhinein in ein Produkt hineingeprüft werden kann, ist es wichtig, Fehler durch konstruktive Qualitätssicherungsmaßnahmen zu vermeiden. Konstruktive Qualitätssicherungsmaßnahmen sind zum Beispiel:

- Auswahl, Einführung und Einsatz von Testmethoden
- Auswahl, Einführung und Einsatz von Testwerkzeugen
- Schulung[4] der handelnden Personen
- Vorgabe von Standards und Beispielen
- Vorgabe verbindlicher Checklisten

Konstruktive Qualitätssicherungsmaßnahmen werden in den frühen Projektphasen vor der Erstellung der Arbeitsergebnisse eingesetzt.

analytische QS-Maßnahmen

Im Gegensatz dazu greifen analytische Qualitätssicherungsmaßnahmen, wenn die zu prüfenden Objekte fertiggestellt sind. Sie haben die Bewertung der Qualität der Prüfgegenstände zum Ziel. Prüfungen von Dokumenten und Tests von Software sind analytische Qualitätssicherungsmaßnahmen.

> Es ist effizienter, Fehler zu vermeiden, als Fehler zu finden.

Die letzte der vier Eingangsfragen, wie Qualität für web-basierte Anwendungen erzeugt werden kann, lässt sich nur mit dem Wissen um die konstruktiven und analytischen Qualitätssicherungsmaßnahmen beantworten. Dieses Wissen ist die Essenz dieses Buches und wird in den folgenden Kapiteln vermittelt. Die erschöpfende Antwort auf die vierte Qualitätsfrage kann der Leser also nach dem Durcharbeiten dieses Buches geben.

2.5 Zusammenfassung

- Die Normen ISO/IEC 9126 (DIN 66272) und DIN EN ISO 9241 (Teil 10 und 11) definieren die Qualität von Software anhand von Merkmalen.

[4] Womit das Lesen dieses Buches zu den konstruktiven Qualitätssicherungsmaßnahmen gezählt werden kann ☺

- Die Hauptqualitätsmerkmale von Software sind Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit.
- Von einer dialogorientierten Anwendung wird gefordert, dass sie aufgabenangemessen, erwartungskonform, fehlertolerant individualisierbar, lernförderlich, selbstbeschreibungsfähig und steuerbar ist.
- Gebrauchstauglichkeit bedeutet, dass der Benutzer effektiv und effizient mit einer Anwendung arbeiten kann und mit dem „Gesamtkunstwerk“ zufrieden ist.
- Web-Applikationen bedürfen aufgrund ihrer unbekannten und großen Nutzergemeinde besonderer Anstrengungen, um die zusätzlichen web-spezifischen Qualitätssicherungsmerkmale Auffindbarkeit, Barrierefreiheit und Rechtskonformität nachweisen zu können.
- Erst wenn jedem Qualitätsmerkmal eine entsprechende Messlatte zugeordnet ist, sind die Qualitätsanforderungen an eine Applikation definiert, und erst dann kann die Qualitätssicherung die Abweichungen vom Soll messen.
- Die Qualitätssicherung muss mit geeigneten konstruktiven Qualitätssicherungsmaßnahmen dafür Sorge tragen, dass Fehler vermieden werden.
- Die Qualitätssicherung muss mit angemessenen analytischen Qualitätssicherungsmaßnahmen nachweisen, dass die Qualitätsanforderungen an eine Software erfüllt werden oder nicht erfüllt werden.

Qualität ist das Maß der Nichtabweichung des Ists vom Soll.[5]

[5] Der Autor auf die Frage: „Was ist Qualität?“

3 Begriffe zum Testen

„Wer testet, ist feige."

Zitat eines unverbesserlichen Entwicklers

In diesem Kapitel werden die in diesem Buch verwendeten Begriffe zum Thema Testen definiert und die Box-Tests erläutert, mit dem Ziel, eine Basis für die anschließenden Kapitel zu schaffen. Das ist notwendig, weil erfahrungsgemäß die Begriffe rund ums Testen von Unternehmen zu Unternehmen, von Projekt zu Projekt, von Tester zu Tester unterschiedlich interpretiert werden.

3.1 Definitionen zum Testen[1]

Ein Begriff, der in diesem Kapitel erstmals benutzt, aber weiter unten im Text erklärt wird, ist mit einem Pfeil (→) gekennzeichnet.

Hinweis

Prüfung ist der Oberbegriff für alle analytischen Qualitätssicherungsmaßnahmen unabhängig von Methode und Prüfgegenstand.

Prüfung

[1] Die Definitionen orientieren sich am Glossar des International Software Testing Qualification Boards ([ISTQB_2006]).

Testen Testen ist der Prozess, der sämtliche (Test-)Aktivitäten umfasst, welche dem Ziel dienen, für ein Software-Produkt die korrekte und vollständige Umsetzung der Anforderungen sowie das Erreichen der festgelegten Qualitätsanforderungen nachzuweisen.

Zum Testen gehören in diesem Sinne auch Aktivitäten zur Planung, Steuerung, Vorbereitung und Bewertung, welche der Erreichung der genannten Ziele dienen.

Fehler Ein Fehler ist die Nichterfüllung einer festgelegten Forderung. In der Software-Entwicklung fasst ein Fehler die drei Begriffe →Fehlhandlung, →Fehlerzustand und →Fehlerwirkung zusammen.

Fehlhandlung Eine Fehlhandlung (error) ist die menschliche Handlung, die zu einem →Fehlerzustand in einer Software führt.

Fehlerzustand Ein Fehlerzustand (fault) ist der Zustand eines Softwareprodukts oder einer Softwarekomponente, der unter spezifischen Bedingungen eine geforderte Funktion des Softwareprodukts beeinträchtigen und zu einer →Fehlerwirkung führen kann. Ein Fehlerzustand wird auch als Defekt, innerer Fehler oder Fehlerursache bezeichnet.

Fehlerwirkung Eine Fehlerwirkung (failure) ist die Wirkung eines Fehlerzustands, die bei der Ausführung der Software/des →Testobjekts für den Benutzer/Tester sichtbar in Erscheinung tritt. Eine Fehlerwirkung wird auch als äußerer Fehler bezeichnet.

Fehlermaskierung Eine Fehlermaskierung ist ein Umstand, bei der ein Fehlerzustand oder mehrere Fehlerzustände die Aufdeckung eines anderen Fehlerzustandes verhindern, d.h. seine Fehlerwirkung kompensieren.

Test Ein Test ist eine Aktivität, bei der ein →Testobjekt mittels geeigneter →Testmethoden →statisch überprüft oder →dynamisch ausgeführt wird, mit dem Ziel, Fehler aufzudecken und die korrekte und vollständige Umsetzung der →funktionalen und →nicht-funktionalen Anforderungen nachzuweisen.

Testobjekt Ein Testobjekt (Testeinheit) ist ein Dokument, eine Klasse, ein Programm, eine Komponente, ein Teilsystem oder das gesamte System, das einem Test unterzogen wird.

Die drei Testobjekte →Website, →Web-Seite und →Homepage werden manchmal fälschlicher Weise synonym verwendet. Als Merksatz zur Unterscheidung der drei Begriffe dient diese Qualitäts-

sicherungsanforderung: Jede Web-Seite einer Website muss einen Link auf die Homepage enthalten.

Web-Seite

Eine Web-Seite (engl. Webpage) ist eine einzelne Internet-Seite, d.h. ein Teil einer →Website.

Website

Eine Website (dt. Web-Präsenz oder Web-Auftritt) ist die Gesamtheit aller Seiten eines Web-Angebotes, d.h. alle Web-Seiten bilden zusammen eine Website.

Homepage

Eine Homepage ist die Startseite einer Website.

statischer Test

Ein statischer Test ist die Prüfung eines Testobjektes ohne Rechnerunterstützung.

dynamischer Test

Ein dynamischer Test ist die Prüfung eines Testobjektes mit Rechnerunterstützung, d.h. unter Ausführung der Software.

funktionale Anforderung

Funktionale Anforderungen beschreiben das korrekte Ein- und Ausgabeverhalten und die Verarbeitung der Daten für eine Software-Komponente.

nicht-funktionale Anforderung

Nicht-funktionale Anforderungen beschreiben das allgemeine Verhalten einer Software-Komponente, wie zum Beispiel die Performanz oder Benutzerfreundlichkeit.

funktionaler Test

Ein funktionaler Test ist ein dynamischer Test, der die vollständige und korrekte Umsetzung der funktionalen Anforderungen an ein Testobjekt prüft. Die Testfälle für den funktionalen Test werden aus den funktionalen Anforderungen hergeleitet.

nicht-funktionaler Test

Ein nicht-funktionaler Test ist ein dynamischer Test, der die Einhaltung der nicht-funktionalen Anforderungen an ein Testobjekt prüft. Die Testfälle für den nicht-funktionalen Test werden aus den nicht-funktionalen Anforderungen hergeleitet.

Testtyp

Ein Testtyp (Testart) ist eine Gruppe von →Testmethoden und Testaktivitäten, die gemeinsam ausgeführt und verwaltet werden, mit dem Ziel, ein Testobjekt auf spezielle Qualitätsmerkmale1 hin zu

[1] Qualitätsmerkmale einer Web-Anwendung sind in den Kapiteln 2.1 und 2.2 definiert.

überprüfen. Ein Testtyp kann sich über mehrere →Teststufen erstrecken.

Teststufe

Eine Teststufe (Testphase) fasst diejenigen Testaufgaben in einem Software-Entwicklungsprojekt zusammen, die ein bestimmtes Testziel verfolgen. Teststufen nach dem V-Modell sind Komponententest, Integrationstest, Systemtest und Abnahmetest.

Teststufen und Testtypen werden oftmals gleich benannt (wie zum Beispiel Komponententest, Integrationstest oder Systemtest), obwohl sie per Definition unterschiedliche Bedeutungen haben. Testtypen beschreiben Verfahren zur Überprüfung bestimmter Qualitätsmerkmale und können in mehreren Teststufen angewendet werden. Teststufen beschreiben Testaktivitäten, die ein bestimmtes Testziel erreichen sollen und dazu eventuell mehrere Testtypen einsetzen.

Hinweis

Die einzelnen Testtypen sind Inhalt des zweiten Teils dieses Buches. Im Kapitel 14 im dritten Teil werden die einzelnen Teststufen beschrieben und die Testtypen aufgeführt, die in einer Teststufe eingesetzt werden können.

Testmethode

Eine Testmethode (Testverfahren) ist ein regelwerkbasiertes, systematisches Vorgehen zur Herleitung oder Auswahl von →Testfällen und →Testszenarien, zur Erstellung von →Testspezifikationen und zur Durchführung von Tests.

Testfall

Ein Testfall ist eine Anweisung zur Durchführung der Überprüfung einer Anforderung an das Testobjekt und umfasst:

- die Vorbedingungen, die zur Ausführung notwendig sind
- die Eingabedaten
- die Aktionen, die zur Eingabe der Daten notwendig sind
- die erwarteten Ergebnisse – sowohl Werte als auch Systemzustände
- die Aktionen, die zur Überprüfung der Ergebnisse notwendig sind

Testszenario

Ein Testszenario umfasst mehrere, fachlich oder technisch zusammengehörende Testfälle, die gemeinsam in einer vorgegebenen Reihenfolge in einem →Testlauf durchgeführt werden.

Testlauf

Ein Testlauf ist die Ausführung eines Testszenarios oder mehrerer Testszenarien mit einer definierten Version eines bestimmten Testobjekts.

Testspezifikation

Eine Testspezifikation beschreibt und begründet die Testfälle und Testszenarien, die in der Testdurchführung zu berücksichtigen sind, und legt die zur Testdurchführung notwendigen Aktivitäten fest.

Metrik

Eine Metrik (Maß) beschreibt allgemein eine Messvorschrift, um Daten über eine zu untersuchende Einheit zu einer bestimmten Eigenschaft erheben zu können.

Software-Metrik

Eine Software-Metrik ist eine Größe zur Messung eines Qualitätsmerkmals für ein Testobjekt.

Testmetrik

Eine Testmetrik ist die messbare Eigenschaft eines Testfalls oder Testlaufs mit Angabe der zugehörigen Messvorschrift, wie zum Beispiel eines Testabdeckungsgrades1. Durch Testmetriken werden →Testendekriterien definiert.

Testende-kriterium

Testendekriterien legen fest, unter welchen Voraussetzungen die Tests zu einem Testobjekt eingestellt werden. Zum Beispiel wird der Test einer Komponente beendet, wenn 100% aller festgelegten Testfälle und 80% aller Programmanweisungen ausgeführt worden sind.

3.2 Box-Tests

Beim Testen gibt es unterschiedliche Sichten auf bzw. in das Testobjekt, die durch die Box-Tests (Blackbox, Whitebox, Greybox) beschrieben werden. Gelegentlich werden die Box-Tests mit den Box-Verfahren verwechselt, weil die gleichnamigen Tests und Verfahren in engem Bezug zueinander stehen. Der folgende Abschnitt erläutert die Unterschiede.

[1] Testabdeckungsgrade sind im Kap. 4.2.1 beschrieben.

3.2.1 Blackbox-Test

Blackbox-Test

> Ein Blackbox-Test ist ein funktionaler oder nicht-funktionaler Test, der ohne Nutzung von Informationen über Interna des Testobjektes ausgeführt wird.

Im Blackbox-Test wird die fehlerfreie und vollständige Umsetzung einer Anforderung überprüft. Dazu werden Tests durchgeführt, ohne dass die internen Strukturen der Programme betrachtet werden, d.h. der Tester sieht ein Testobjekt als Blackbox. Beim Blackbox-Test werden nur die Eingaben und Ausgaben des Testobjektes analysiert. In jeder Teststufe können Tests als Blackbox-Test durchgeführt werden.

Wenn von Blackbox-Tests gesprochen wird, sind oft die im Kapitel 4.1 beschriebenen Blackbox-Verfahren zur systematischen Erstellung von Testfällen gemeint. Die beiden Begriffe dürfen aber nicht verwechselt werden. Blackbox-Verfahren sind Testmethoden zur Erstellung von Testfällen, welche anschließend in einem Blackbox-Test ausgeführt werden können.

3.2.2 Whitebox-Test

Whitebox-Test

> Ein Whitebox-Test prüft die innere Struktur eines Testobjektes auf Korrektheit und Vollständigkeit, d.h. Systemstrukturen und Programmabläufe sind bekannt und der Programmcode wird analysiert.

Strukturtest

Der Whitebox-Test wird daher auch Strukturtest genannt.

Der Begriff Whitebox-Test wird oft synonym zu den Whitebox-Verfahren verwendet, was nicht richtig ist. Whitebox-Tests sagen etwas über die Art der Testdurchführung aus und können in jeder Teststufe eingesetzt werden. Whitebox-Verfahren dienen der Ermittlung von Testfällen, die in Whitebox-Tests ausgeführt werden können. Whitebox-Verfahren werden im Kapitel 4.2 beschrieben.

3.2.3 Greybox-Test

Man ahnt es schon. Ein Greybox-Test ist weder eine Testphase, noch liefert er eigene Verfahren zur Ermittlung von Testfällen. Der

Greybox-Test vereint die beiden Sichtweisen des Blackbox- und des Whitebox-Tests auf das zu testende Objekt.

Greybox-Test

Der Greybox-Test prüft eine Greybox auf die richtige Umsetzung der Anforderungen, den korrekten Ablauf und den Aufbau der inneren Strukturen.

Greybox

Eine Greybox ist eine Komponente, die über ihre Schnittstellen nach außen definiert wird und dessen innere Struktur trotzdem beobachtet werden kann.

Es stellt sich die Frage, in welchen Fällen ein Greybox-Test durchgeführt wird. Für bereits existierende Systemkomponenten, die geprüft und produktiv im Einsatz sind, oder für Standardsoftware, die in ein System eingebunden wird, ist es nicht notwendig, das komplette Testgeschütz aufzufahren. Es genügt in solchen Fällen, die benötigten Schnittstellen mit den notwendigen Testdaten aufzurufen und dabei die für den Test relevanten internen Strukturen zu berücksichtigen. Die notwendigen Testfälle werden durch Blackbox- und Whitebox-Verfahren ermittelt. Integrationstests, die unterschiedliche Web-Komponenten zusammenführen, sind klassische Greybox-Tests (s. Kap. 8.3).

Da der Greybox-Test keine eigenen Methoden verpflichtet, können ihm weder spezifische Tools noch eigene Qualitätsanforderungen zugeordnet werden – grau eben.

3.3 Zusammenfassung

- Das Glossar des ISTQB (International Software Testing Qualification Board) [ISTQB_2006] legt die Testbegriffe auf internationaler Ebene fest.
- Testtypen beschreiben Verfahren zur Überprüfung bestimmter Qualitätsmerkmale und können in mehreren Teststufen angewendet werden.
- Teststufen beschreiben Testaktivitäten, die ein bestimmtes Testziel erreichen sollen und dazu eventuell mehrere Testtypen einsetzen.
- Blackbox-Tests, Whitebox-Tests und Greybox-Tests beschreiben unterschiedliche „Erleuchtungsgrade“ eines Testobjekts zur Testdurchführung, d.h. die interne Programmstruktur wird beim

Test nicht (schwarz), ein wenig (grau) oder vollständig (weiß) ausgeleuchtet.

- Für den Tester ist der Programmcode beim Blackbox-Test eine schwarze Kiste und beim Whitebox-Test ein offenes Buch.
- Die Blackbox- und Whitebox-Verfahren sind Methoden zur systematischen Erstellung von Testfällen.
- Es gibt keine Greybox-Verfahren zu Ermittlung von Testfällen. Der Greybox-Test vereint die beiden Sichtweisen des Blackbox- und des Whitebox-Tests auf das zu testende Objekt.

4 Testfallentwurfsverfahren

„Die Mittelmäßigen klopfen sich zu dem Zeitpunkt auf die Schulter, wo die Könner anfangen zu arbeiten."

Matthias Scharlach (*1947)

Um alle Tests, die zur Qualitätssicherung eines Software-Produktes notwendig sind, bei kalkulierbarem und bezahlbarem Aufwand durchführen zu können, müssen die „richtigen" Testfälle gefunden und spezifiziert werden. Die im Folgenden beschriebenen Blackbox- und Whitebox-Methoden zum systematischen Testfallentwurf werden schon viele Jahre praktiziert. Sie sind keine Erfindung des Webtestings, liefern aber Testfälle für viele web-spezifische Testtypen. Daher muss jeder, der Testfälle entwirft, diese Methoden beherrschen.

Ein intuitives Vorgehen zur Ermittlung von Testfällen ist die Fehlererwartungsmethode, die ebenfalls vorgestellt wird.

4.1 Blackbox-Verfahren

Ein vollständiger Test, der alle möglichen Ein- und Ausgaben zu einem Programm überprüft, ist nicht durchführbar. Um die Anzahl der Testfälle auf ein sinnvolles, aber hinreichendes Maß zu reduzieren, werden die Blackbox-Verfahren angewendet.

anforderungsbasierte Testfallermittlung

Blackbox-Verfahren sind anforderungsbasierte Methoden zur Testfallerstellung. Das bedeutet, sie analysieren die Anforderungsspezifikationen und leiten daraus Testfälle ab. Weil sie dabei die Testobjekte als schwarze Box, also nicht die inneren Programmstrukturen betrachten, werden sie Blackbox-Verfahren genannt. Dazu gehören die Äquivalenzklassen-, Grenzwert- und Ursache-Wirkungs-Analyse.

An dem Beispiel eines Rechners zur Autofinanzierung werden die einzelnen Schritte, die im Idealfall bei der Anwendung der Blackbox-Verfahren durchgeführt werden, erläutert. Beginnen wir mit der Anforderung:

1. Schritt: Anforderungen lesen

Beispiel Finanzierungsrechner

Zu testen ist das Programmmodul <Finanzierungsrechner>, das die Monatsraten für eine Autofinanzierung berechnet.

Der Gesamtfahrzeugpreis wird dem <Finanzierungsrechner> vom Modul <Autokonfigurator> übergeben, nachdem dort das Wunschfahrzeug zusammengestellt worden ist. Die konkreten Anforderungen an den Finanzierungsrechner sind in der Konzeptionsphase beschrieben – und natürlich qualitätsgesichert – worden. Sie lauten:

Der Kunde muss eine Anzahlung zwischen 2.000 und 10.000 Euro als ganze Zahl eingeben. Die Vertragslaufzeit wird durch drei Radio-Buttons vorgegeben: 12, 24 oder 36 Monate. Die Laufzeiten haben unterschiedliche Zinskonditionen. Je länger die Laufzeit ist, desto höher sind die jährlichen Zinsen (2%, 3%, 4%), welche für den Betrag der Differenz von Fahrzeugpreis und Anzahlung zu zahlen sind.

Zusätzlich kann in einer Checkbox angekreuzt werden, ob ein Altfahrzeug in Zahlung gegeben werden soll (ist standardmäßig nicht angekreuzt). Falls nicht, wird ein Rabatt von 3% auf den Kaufpreis gewährt, aber nur wenn die Laufzeit nicht 36 Monate beträgt.

Ein Preisnachlass von 1.000 Euro wird gewährt, wenn der Fahrzeugpreis über 30.000 Euro liegt und die Vertragslaufzeit auf 12 Monate festgelegt ist.

4.1.1 Äquivalenzklassenanalyse

Bei der Äquivalenzklassenanalyse wird die Menge der möglichen Testfälle anhand der in den Anforderungsspezifikationen beschriebenen Bedingungen in eine endliche Zahl von äquivalenten Klassen unterteilt.

Eine Äquivalenzklasse ist eine Menge von Eingabewerten, die ein identisches funktionales Verhalten eines Testobjektes auslösen, bzw. eine Menge von Ausgabewerten, die ein gleichartiges Verhalten eines Testobjektes aufzeigen.

Äquivalenzklasse

Repräsentanten

Für alle Elemente aus einer Äquivalenzklasse wird angenommen, dass sie bei einer Testausführung dieselbe Wirkung erzielen, d.h. die Ergebnisse äquivalent zueinander sind. Daher genügt es für den Test, pro Äquivalenzklasse nur einen Repräsentanten auszuwählen. Zwei Repräsentanten (Testdaten) einer Äquivalenzklasse kommen entweder zu einem gleichen Testergebnis oder decken dieselbe Fehlerwirkung auf. So wird einerseits die Anzahl der möglichen Testfälle systematisch reduziert und andererseits erhält man eine hinreichende Anzahl von Testfällen, um – zumindest aus Sicht der Blackbox-Verfahren – die vollständige und korrekte Umsetzung der Anforderungen nachweisen zu können.

zulässige und unzulässige Äquivalenzklassen

Im Rahmen der Äquivalenzklassenanalyse wird jede in den Anforderungen beschriebene Bedingung in Äquivalenzklassen umgeschrieben. Dabei wird jede Äquivalenzklasse entweder den zulässigen Äquivalenzklassen oder den unzulässigen Äquivalenzklassen zugeordnet. Zulässige Äquivalenzklassen beschreiben den regulären Ablauf des Programms. Unzulässige Äquivalenzklassen beschreiben geplante Unterbrechungen und Fehlermeldungen im Ablauf eines Programms.

2. Schritt: Äquivalenzklassen definieren

Zu jeder in den Anforderungen gestellten Bedingung werden Äquivalenzklassen gebildet. Dabei bleiben Abhängigkeiten der Bedingungen untereinander erst einmal unberücksichtigt.

Unser Finanzierungsrechner hat die drei Eingabemöglichkeiten <Anzahlung>, <Laufzeit> und <Inzahlungnahme Altfahrzeug>. Dazu werden die entsprechenden Äquivalenzklassen (Tabelle 4.1) gebildet.[1] Die Äquivalenzklassen werden durchnummeriert und mit einem Präfix versehen, um nachweisen zu können, dass jede Äquivalenzklasse in mindestens einem Testfall geprüft wird. Zulässige Äquivalenzklassen erhalten das Präfix „Z“, unzulässige das Präfix „U“.

Zu einer Bedingung müssen nicht immer unbedingt ungültige Äquivalenzklassen existieren. Für das Eingabefeld <Inzahlungnahme Altfahrzeug> gibt es zum Beispiel nur gültige Eingaben.

[1] Um das Beispiel nicht überzustrapazieren, wird im Folgenden auf Formatprüfungen wie Dezimalpunkte und Nachkommastellen verzichtet.

Tabelle 4.1: Äquivalenzklassen

Bedingung	zulässige Äquivalenzklasse	Nr.	unzulässige Äquivalenzklasse	Nr.
Anzahlung	Ganze Zahl 2.000 bis 10.000	Z1	Keine Eingabe Kleiner als 2.000 Größer als 10.000 Nicht ganze Zahl Nicht numerische Eingabe	U1 U2 U3 U4 U5
Laufzeit	12 24 36	Z2 Z3 Z4	Keine Eingabe	U6
Inzahlungnahme Altfahrzeug	Nein Ja	Z5 Z6		

3. Schritt: Testfälle und Testszenarien zu den Äquivalenzklassen beschreiben

Im nächsten Schritt werden Testfälle und Testszenarien zu den Äquivalenzklassen nach folgenden Regeln festgelegt:

1. Jede Äquivalenzklasse wird von mindestens einem Testfall abgedeckt. Der Nachweis der Vollständigkeit der Testfälle erfolgt durch die Nummerierungen der Äquivalenzklassen, die in der Testfallbeschreibung referenziert werden (Z1, Z2,... ,U1, U2,...).
2. Für jeden Testfall werden die zu erwartenden Testergebnisse festgelegt und beschrieben.
3. Die Testfälle für zulässige Äquivalenzklassen werden zu Testszenarien zusammengefasst, die möglichst viele Testfälle enthalten. Wenn man bedenkt, dass zu den Testfällen auch Testdaten und eventuell automatisierte Testskripte zur Durchführung erstellt werden müssen, kann der Aufwand auf diese Weise minimiert werden.
4. Fehlertestfälle, d.h. Testfälle für unzulässige Äquivalenzklassen, müssen einzeln in Testszenarien geprüft werden, weil sich Fehlersituationen sonst überlagern könnten (Fehlermaskierung).

Aus diesen Regeln ergeben sich für unser Beispiel vorerst drei Testszenarien für die gültigen und sechs für die ungültigen Äquivalenz-

klassen (Tabelle 4.2). Da der Finanzierungsrechner die Finanzierungsrate nur berechnet, wenn alle drei Eingaben im Eingabefenster vorhanden sind, werden einige Äquivalenzklassen in mehreren Testszenarien aufgeführt.

Nr. Testszenario[2]	Äquivalenzklassen	Erwartetes Ergebnis / Erwartete Reaktion
1	Z1, Z2, Z5	Monatsrate wird richtig berechnet.
2	Z1, Z3, Z5	Monatsrate wird richtig berechnet.
3	Z1, Z4, Z6	Monatsrate wird richtig berechnet.
10	**U1**, Z2, Z5	Hinweis: Sie haben keine Anzahlung eingegeben.
11	**U2**, Z2, Z5	Hinweis: Die Anzahlung muss zwischen 2000 und 10000 Euro liegen.
12	**U3**, Z2, Z5	Hinweis: Die Anzahlung muss zwischen 2000 und 10000 Euro liegen.
13	**U4**, Z4, Z6	Eingabe nicht möglich, in dem Feld sind nur ganze Zahlen eingebbar.
14	**U5**, Z4, Z6	Eingabe nicht möglich, in dem Feld sind nur Zahlen eingebbar.
15	**U6**, Z1, Z6	Eingabe nicht möglich, die Laufzeit wird mit Radio-Buttons ausgewählt, Standard ist 24 Monate.

Tabelle 4.2: Testszenarien zu Äquivalenzklassen

Mit diesen neun Testszenarien werden alle Eingaben in den Finanzierungsrechner überprüft. Für die Berechnung der korrekten Leasingraten wird allerdings der Kaufpreis, der vom aufrufenden Modul <Autokonfigurator> an den Finanzierungsrechner übergeben wird, benötigt. Die in Tabelle 4.3 aufgeführten Äquivalenzklassen zum Kaufpreis müssen daher noch bei der Testfallerstellung berücksichtigt werden.

Bedingung	Zulässige Äquivalenzklasse	Nr	Unzulässige Äquivalenzklasse	Nr
Kaufpreis	13.000 bis 30.000	Z7		
	größer als 30.000	Z8		

Tabelle 4.3: Äquivalenzklassen zum Kaufpreis

[2] Die Nummerierung ist nicht fortlaufend, weil später eventuell noch Testszenarien zu gültigen Äquivalenzklassen ergänzt werden müssen.

Ungültige Äquivalenzklassen zum Kaufpreis müssen hier nicht definiert und gestestet werden, weil das die Aufgabe der Tests zum Autokonfigurator ist. Für den Test des Finanzierungsrechners können wir also davon ausgehen, dass der Autokonfigurator immer korrekt einen ganzzahligen Kaufpreis von mindestens 13.000 Euro übergibt. Die in Tabelle 4.2 festgelegten Testszenarien werden um die neuen Äquivalenzklassen Z7 und Z8 ergänzt, wie in Tabelle 4.4 exemplarisch für die ersten drei Szenarien dargestellt ist.

Tabelle 4.4: Testszenarien mit Kaufpreis

Test-szenario	Äquivalenzklas-sen	Erwartetes Ergebnis / Erwartete Reaktion
1	Z1, Z2, Z5, **Z7**	Monatsrate wird richtig berechnet.
2	Z1, Z3, Z5, **Z8**	Monatsrate wird richtig berechnet.
3	Z1, Z4, Z6, **Z7**	Monatsrate wird richtig berechnet.

Für unser Beispiel sind nun alle Äquivalenzklassen mit Testfällen abgedeckt. Aber der engagierte Tester möchte natürlich noch prüfen, ob das Programm auch an den Rändern der möglichen Eingaben, d.h. an den Grenzen der Äquivalenzklassen, richtig funktioniert.

4.1.2 Grenzwertanalyse

Grenzwert-analyse

> Die Grenzwertanalyse stellt sicher, dass Fehler in den kritischen Grenzbereichen der Äquivalenzklassen gefunden werden.

Dazu muss jeder Rand einer gültigen bzw. ungültigen Äquivalenzklasse mit mindestens einem Testfall abgedeckt werden. Für den Kaufpreis unseres Finanzierungsrechners ergeben sich zum Beispiel die Grenzwerte 12.999, 13.000, 30.000 und 30.001.

Grenzwerte, die mit Hilfe von Äquivalenzklassen gefunden werden, stellen Repräsentanten im Sinne der Äquivalenzklassenanalyse dar. Sie sollten aber nicht die Repräsentanten aus den Äquivalenzklassen ersetzen, die nicht auf den Grenzen liegen, sondern diese ergänzen.

4. Schritt: Grenzwerte festlegen

Für unser Beispiel werden die in Tabelle 4.5 dargestellten Grenzwerte festgelegt, die je Äquivalenzklasse mit „GW“ gekennzeichnet und durchnummeriert werden. Aus dem Inhalt der Tabelle 4.5 ergeben sich einige Erkenntnisse:

1. Gültige Äquivalenzklassen können auch Grenzwerte darstellen, wie im Beispiel die Äquivalenzklassen Z2 und Z4, welche die Grenzen zur Eingabe der Laufzeit darstellen.
2. Nicht zu jeder Äquivalenzklasse gibt es sinnvolle Grenzwerte, wie im Beispiel zu den Äquivalenzklassen Z5, Z6, U1, U4, U5, U6.
3. Bei der Grenzwertanalyse werden mit den Grenzwerten die konkreten Testdaten für einen Testfall vorgegeben, wie zum Beispiel 2.000 in Z1-GW1. Das ist bei den Testfällen für die „normalen" Äquivalenzklassen nicht der Fall (s. dazu „Beispiel 9. Schritt: Konkrete Testfälle definieren").

Tabelle 4.5: Grenzwerte zu Äquivalenzklassen

Bedingung	Grenzwert zulässige Äquivalenzklasse	Nr.	Grenzwert unzulässige Äquivalenzklasse	Nr.
Anzahlung	2.000 10.000	Z1-GW1 Z1-GW2	0 1.999 10.001	U2-GW1 U2-GW2 U3-GW1
Kaufpreis	12.999 13.000 30.000 30.001	Z7-GW1 Z7-GW2 Z8-GW1 Z8-GW2		

Tabelle 4.6: Testszenarien mit Grenzwerten

Nr. Testszenario	Äquivalenzklassen mit Grenzwerten	Erwartetes Ergebnis / Erwartete Reaktion
4	**Z1 GW1**, Z2, Z5, **Z7 GW2**	Monatsrate wird richtig berechnet.
5	**Z1 GW2**, Z2, Z5, **Z8 GW1**	Monatsrate wird richtig berechnet.
6	Z1, Z2, Z5, **Z8 GW2**	Monatsrate wird richtig berechnet.
16	**U2 GW1**, Z2, Z5, Z7	Hinweis: Die Anzahlung muss zwischen 2000 und 10000 Euro liegen.
17	**U2 GW2**, Z2, Z5, Z7	Hinweis: Die Anzahlung muss zwischen 2000 und 10000 Euro liegen.
18	**U3 GW1**, Z2, Z5, Z8	Hinweis: Die Anzahlung muss zwischen 2000 und 10000 Euro liegen.

5. Schritt: Testszenarien um Grenzwerte ergänzen

Die bisher beschriebenen Testszenarien werden um die Grenzwerttestfälle ergänzt.

Das Ergebnis zu unserem Beispiel ist in Tabelle 4.6 dargestellt. Der Grenzwert des Kaufpreises von 12.999 Euro zur gültigen Äquivalenzklasse Z7 muss an dieser Stelle nicht betrachtet werden, weil beim Test des Autokonfigurators sichergestellt werden muss, dass dieser Wert niemals an den Finanzierungsrechner übergeben wird.

4.1.3 Ursache-Wirkungs-Analyse

Bisher wurden keine Abhängigkeiten zwischen den Bedingungen der Anforderungen berücksichtigt. Aber gerade die möglichen Kombinationen ergeben die „interessanten" Testfälle, die in der Analyse von Ursache und Wirkung untersucht werden.

Ursache-Wirkungs-Analyse

Mit Hilfe der Ursache-Wirkungs-Analyse werden Kombinationen von zulässigen Äquivalenzklassen, die unterschiedliche Programmreaktionen zur Folge haben, auf ihr korrektes Zusammenspiel hin getestet.

Weil Tests zu den Testfällen, die auf unzulässigen Äquivalenzklassen basieren, in jedem Fall Fehlermeldungen als Ergebnis liefern und keine korrekte Durchführung der Funktionalitäten zulassen, werden bei der Ursache-Wirkungs-Analyse nur zulässige Äquivalenzklassen betrachtet.

Ursache-Wirkungs-Graphen

In der Literatur wird für die Testfallerstellung von Eingabekombinationen die Methode der Ursache-Wirkungs-Graphen beschrieben (zum Beispiel bei [Myers_1987]). Dieses ist eine formale Sprache, die in komplexen Graphen die kombinatorischen Zusammenhänge beschreibt.

Entscheidungstabellen

Bei der Testfallermittlung kommerzieller Anwendungen haben sich Entscheidungstabellen zur Darstellung von Kombinationen möglicher Ursachen und den daraus resultierenden Wirkungen bewährt. Zudem eignen sich Entscheidungstabellen aufgrund ihrer guten Lesbarkeit optimal zur Kommunikation mit Fachabteilungen.

Eine Entscheidungstabelle besteht aus vier Blöcken:

Ursachen	Regeln
Wirkungen	Entscheidungen

- **Ursachen**
  sind Situationen oder Bedingungen, die eine Entscheidung beeinflussen. Für die Testfallerstellung sind das die zulässigen Äquivalenzklassen.
- **Wirkungen**
  listen die möglichen Ausgaben, Programmreaktionen oder Systemzustände auf.
- **Regeln**
  beschreiben die Kombinationsmöglichkeiten der Ursachen. Sie werden für die Testfallermittlung aus den Anforderungsbeschreibungen abgeleitet und bilden die fachlichen Abhängigkeiten der zulässigen Äquivalenzklassen untereinander ab.
- **Entscheidungen**
  beschreiben, welche konkreten Wirkungen aufgrund einer Regel erwartet werden.

6. Schritt: Entscheidungstabelle mit Abhängigkeiten aufstellen

Die bei Äquivalenzklassen- und Grenzwertanalyse vernachlässigten Abhängigkeiten von Anforderungsbedingungen werden nun explizit analysiert.

Ursachen			**Regeln**											
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Laufzeit	12 = 2 %	Z2	X	X	X	X								
	24 = 3 %	Z3					X	X	X	X				
	36 = 4 %	Z4									X	X	X	X
Inzahlungnahme Altfahrzeug	Nein	Z5	X	X			X	X			X	X		
	Ja	Z6			X	X			X	X			X	X
Kaufpreis	<= 30.000	Z7	X		X		X		X		X		X	
	> 30.000	Z8		X		X		X		X		X		X
Wirkungen			**Entscheidungen**											
3% Rabatt auf Kaufpreis		W1	J	J	N	N	J	J	N	N	N	N	N	N
Preisnachlass 1.000		W2	N	J	N	J	N	N	N	N	N	N	N	N

Tabelle 4.7: Testfallkombinationen

Für unseren Finanzierungsrechner ergeben sich aus den beschriebenen Anforderungen 12 Kombinationen für <Laufzeit>, <Inzahlungnahme Altfahrzeug> und <Kaufpreis> (Tabelle 4.7). Die <Anzah-

lung> hat keinen Einfluss auf die Entscheidungsfindung und wird daher in der Tabelle 4.7 nicht berücksichtigt, was nicht heißt, dass die Höhe der Anzahlung keinen Einfluss auf die zu berechnende Monatsrate hat.

7. Schritt: Entscheidungstabelle reduzieren

Entscheidungstabellen können sehr komplex werden. Bei n binären Bedingungen erhält man 2^n Kombinationen und bei mehr als zwei Möglichkeiten zu einer Bedingung noch wesentlich mehr.

In der Praxis wird man in solchen Fällen, falls das fachliche Risiko tragbar ist, eine repräsentative Auswahl aus allen Kombinationsmöglichkeiten treffen. Diese Auswahl kann durch die Fachabteilung geschehen, die festlegt, welche kombinatorischen Testfälle aus fachlicher Sicht besonders wichtig bzw. zu vernachlässigen sind. Ein systematisches Vorgehen zur Reduzierung der Regeln einer komplexen Entscheidungstabelle ist die Pairwise-Methode.

Pairwise-Methode

Bei der Pairwise-Methode werden in einer Entscheidungstabelle die Regeln gestrichen, die doppelte Paare von gültigen Äquivalenzklassen enthalten.

Auf diese Weise werden zumindest alle Kombinationen von Äquivalenzklassenpaaren getestet. Bei den Streichungen muss darauf geachtet werden, dass jede mögliche Wirkung mindestens einmal zum Tragen kommt, da die Testabdeckung sonst nicht vollständig wäre.

In unserem Beispiel wird mit dieser Methode die Anzahl der zu testenden Kombinationen des Finanzierungsrechners auf vier reduziert:

- In der Tabelle 4.7 bilden die Äquivalenzklassen Z2 und Z5 das erste doppelte Paar in Regel 1 und Regel 2. Die Regel 2 wird gestrichen.
- Das nächste doppelte Paar gültiger Äquivalenzklassen sind Z2 und Z6 in den Regeln 3 und 4. Regel 3 wird gestrichen, da mit Streichung der Regel 4 die Wirkung W2 nie zum Zuge käme.
- In Regel 5 sind dann Z5 und Z7 doppelt mit Regel 1, so dass auch die Regel 5 gestrichen wird.
- Regel 6 kann wieder bleiben.
- Nach dieser Vorgehensweise werden bis auf Regel 11 alle weiteren Regeln gestrichen.

Die Tabelle 4.8 zeigt das Ergebnis aller Streichungen. Darin treten Paare von gültigen Äquivalenzklassen nicht mehr doppelt auf und jede Wirkung kommt mindestens einmal vor.

Ursachen			**Regeln**			
			1	**4**	**6**	**11**
Laufzeit	12 = 2 %	Z2	X	X		
	24 = 3 %	Z3			X	
	36 = 4 %	Z4				X
Inzahlungnahme Altfahrzeug	Nein	Z5	X		X	
	Ja	Z6		X		X
Kaufpreis	<= 30.000	Z7	X			X
	> 30.000	Z8		X	X	
Wirkungen			**Entscheidungen**			
Preisnachlass 1.000		W1	J	N	J	N
3% Rabatt auf Kaufpreis		W2	N	J	N	N

Tabelle 4.8: Reduzierte Testfallkombinationen

In einem realen Projekt würde das Pairwise-Verfahren sicher nicht für eine Tabelle mit 12 Einträgen eingesetzt werden, da der Testaufwand für diese Anzahl von Testfällen vertretbar ist. Aber dieses Beispiel zeigt, dass sich ohne die doppelten Paare zulässiger Äquivalenzklassen die Anzahl der durchzuführenden Tests drastisch reduziert.

Um den Testaufwand im Rahmen zu halten, ist bei umfangreichen Entscheidungstabellen ein solches Reduzierungsverfahren notwendig, dass dann allerdings von einem Werkzeug unterstützt werden sollte (s. Kap. 4.1.7).

Steht kein Testwerkzeug für die Reduzierung von Entscheidungstabellen zur Verfügung, müssen die Streichungen von Personen aus der Fachabteilung vorgenommen werden. Dazu wird in der Regel nicht die Pairwise-Methode angewendet, sondern die Eingabekombinationen werden auf Basis einer Risikobewertung gestrichen.

8. Schritt: Testszenarien um Kombinationen ergänzen

Wenn man die Regeln in Tabelle 4.8 mit den Testszenarien in Tabelle 4.4 und Tabelle 4.6 vergleicht, stellt man fest, dass die Kombination der gültigen Äquivalenzklassen Z2, Z6 und Z8 in Regel 4 in keinem der Testszenarien enthalten ist. Somit ergibt sich ein zusätzliches Szenario für den Test der Regel 4 (Tabelle 4.9):

Tabelle 4.9: Testszenario Ursache-Wirkung

Test-szenario	Äquivalenzklas-sen	Erwartetes Ergebnis / Erwartete Reaktion
7	Z1, Z2, Z6, Z8	Monatsrate wird richtig berechnet.

9. Schritt: Konkrete Testfälle definieren

aus logischen Testfällen werden konkrete

Bisher wurden bis auf einige Grenzwerttestfälle nur logische Testfälle beschrieben. Im Rahmen der Testvorbereitung müssen für alle Testfälle die konkreten Werte festgelegt werden, mit denen die Tests durchgeführt werden sollen. Besonders für Berechnungen ist es wichtig, die Eingabewerte und Ergebnisse konkret zu bestimmen. Denn dann kann das Testergebnis vom Tester eindeutig überprüft oder bei automatisierten Tests vom Testwerkzeug mit dem Sollwert verglichen werden.

Für das erste Testszenario unseres Finanzierungsrechners können folgende konkrete Werte für den Test vorgegeben werden:

Eingabeparameter	Äquivalenzklasse	Eingabewert
Anzahlung:	Z1	4.000
Laufzeit	Z2	12 Monate = 2% Zinsen
Inzahlungnahme Altfahrzeug	Z5	Nein
Kaufpreis	Z7	24.000
Erwartetes Ergebnis / Erwartete Reaktion		1.700 Euro Monatsrate

Mit den bis hier beschriebenen Verfahren zur Testfallermittlung wird normalerweise der größte Teil der durchzuführenden Testfälle für den anforderungsbasierten Test erkannt und beschrieben, d.h. eine hohe Anforderungsabdeckung durch Testfälle wird erreicht. Anhand von Zustands- und Anwendungsfalldiagrammen können weitere fachliche bedingte Testszenarien entworfen werden. Diese Arten der Testfallermittlung werden in den beiden nächsten Abschnitten beschrieben.

4.1.4 Zustandsbasierte Testfallermittlung

Objekte und Systeme können unterschiedliche Zustände annehmen. In der Entwicklung von Realtime-Systemen (Ampelanlagen, Getränkeautomaten) werden Zustandsübergangsdiagramme (State Transition Diagram) „schon immer“ als Mittel zur Anforderungsbeschreibung von Zustandsänderungen eingesetzt.

Zustands-diagramme

Bei der Entwicklung von Web-Applikationen sind technische Zustände von Systemen in der Regel nicht relevant, sondern hier durchlaufen fachliche Objekte verschiedene Lebensabschnitte nach bestimmten Regeln. Die Status, die ein Objekt einnehmen kann, und die dazugehörenden Statusübergänge werden in Zustandsdiagrammen dargestellt. Zustandsdiagramme – alias Zustandsübergangsdiagramme – sind ein sehr mächtiges und für die Fachabteilung verständliches Beschreibungsmittel. Aus ihnen lassen sich recht einfach Testfälle für einen zustandsbasierten Test entwickeln.

zustandsbasierte Testfallermittlung

Die zustandsbasierte (zustandsbezogene) Testfallermittlung entwirft anhand von Zustandsdiagrammen Testfälle zur Überprüfung der Vollständigkeit, Richtigkeit und Erreichbarkeit der Zustände und Zustandsübergänge eines Testobjekts.

In unserem Beispiel durchläuft das Objekt <Finanzierungsanfrage> unterschiedliche Zustände (Abb. 4.1).

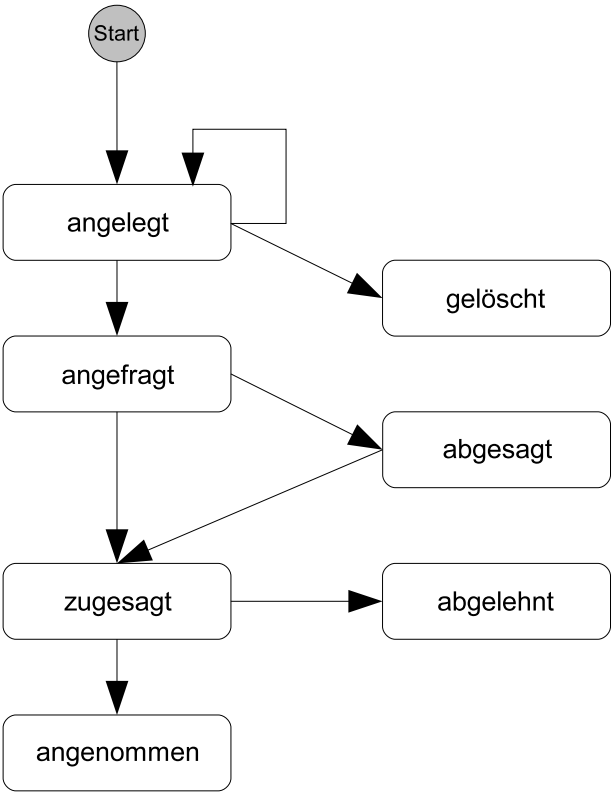


Abb. 4.1: Zustandsdiagramm Finanzierungsanfrage

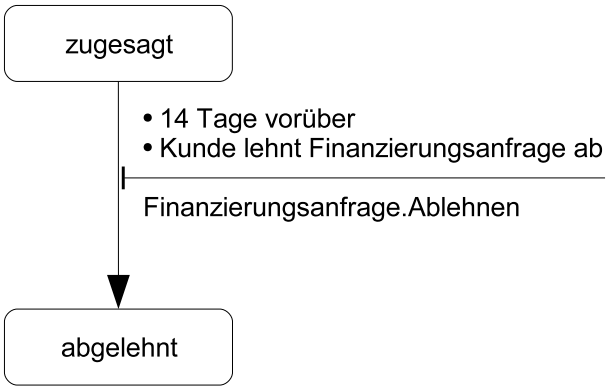
Ein Kunde kann mit dem Finanzierungsrechner eine Finanzierungsmöglichkeit berechnen und abspeichern (Zustand „angelegt“). Natürlich kann er die Berechnung vor dem Speichern verwerfen oder abbrechen, dann ist allerdings kein Zustand der Finanzierungsanfrage erreicht, der im System abgelegt werden müsste.

Zu einem späteren Zeitpunkt kann der Kunde eine angelegte Finanzierungsanfrage verändern (immer noch Zustand „angelegt“), löschen (Zustand „gelöscht“) oder eine konkrete Anfrage starten (Zustand „angefragt“). Vom Händler erhält er darauf eine verbindliche Zusage (Zustand „zugesagt“) oder eine Absage (Zustand „abgesagt“). Eine Absage kann vom Autohändler später doch noch in eine Zusage umgewandelt werden. Im positiven Fall hat der Kunde zwei Wochen Zeit, diese Zusage anzunehmen, was einen Vertragsabschluss zur Folge hat (Zustand „angenommen“). Lässt er die Zusage verfallen oder lehnt er sie explizit ab, führt das zum Zustand „abgelehnt“.

Der Zustand „gelöscht“ einer Finanzierungsanfrage ist virtuell, denn er besagt, dass das Objekt gelöscht wurde und nicht mehr in der Datenbank vorhanden ist. Alle anderen Zustände sind, wenn sie erreicht sind, in der Datenbank abgelegt und jederzeit abrufbar.

Die Abb. 4.1 stellt diese Anforderungen für eine konkrete Finanzierungsanfrage in einem Zustandsdiagramm dar. Das Diagramm ist aus Übersichtsgründen vereinfacht dargestellt. Zum einen fehlt der dem Start-Knoten entsprechende Ende-Knoten, weil alle finalen Zustände (abgesagt, abgelehnt, angenommen, gelöscht) mit ihm verbunden wären. Zum anderen sind keine Ereignisse und Funktionen (Methoden), die zur vollständigen Beschreibung eines Zustandsdiagramms gehören, abgebildet.

Abb. 4.2: Zustandsdiagramm mit Ereignissen



In Abb. 4.2 sind exemplarisch die Ereignisse und die durch die Ereignisse ausgelöste Methode dargestellt, die den Zustandsübergang

einer Finanzierungsanfrage von „zugesagt“ nach „abgelehnt“ herbeiführen.

In einem Zustandsdiagramm sind eine Vielzahl fachlicher Anforderungen enthalten, so auch in Abb. 4.1 und Abb. 4.2:

- Jeder Zustandsübergang wird von mindestens einem Ereignis herbeigeführt und jedes Ereignis muss von der Anwendung bearbeitet werden. Im Beispiel überführen die beiden Ereignisse „14 Tage vorüber“ und „Kunde lehnt Finanzierungsanfrage ab“ das konkrete Objekt <Finanzierungsanfrage> vom Zustand „zugesagt“ in den Zustand „abgelehnt“.
- Zu jedem Ereignis muss eine verarbeitende Funktion spezifiziert und realisiert sein. Im Beispiel werden die beiden Ereignisse „14 Tage vorüber“ und „Kunde lehnt Finanzierungsanfrage ab“ von der Funktion <Finanzierungsanfrage.Ablehnen> bearbeitet.
- Ein Zustandsdiagramm legt Reihenfolgen fest, in denen Ereignisse eintreten müssen bzw. eintreten dürfen, um verarbeitet werden zu können. Um zum Beispiel eine vom Händler zugesagte Finanzierung annehmen zu können, muss zuvor der Zustand „zugesagt“ erreicht worden sein.
- Nicht erlaubte Aufrufsequenzen von Funktionen einer Anwendung sind sofort ersichtlich. Im Beispiel darf eine Finanzierungsanfrage nur gelöscht werden, wenn sie im Zustand „angelegt“ ist. In jedem anderen Zustand darf die Funktion/Methode zum Löschen des Objektes nicht ausführbar bzw. für den Benutzer in der Anwendung erst gar nicht sichtbar sein.

Tipp

Ich habe sehr positive Erfahrungen mit dieser Spezifikationsmethode gemacht und kann sie aus Sicht des Qualitätssicherers nur empfehlen. In der Projektarbeit entbrennen über dieser Art der Darstellung von fachlichen Zusammenhängen sehr konstruktive Diskussionen, weil schnell Lücken in der Spezifikation aufgedeckt werden.

In unserem Beispiel drängen sich bei der Prüfung des Zustandsdiagramms folgende Fragen auf, die fachlich zu klären wären: Darf eine zugesagte Finanzierung geändert werden? Wann kann ein Finanzierungsangebot von wem storniert werden? Ist die Vertragsabwicklung der angenommenen Finanzierung in weiteren Zuständen dieses Objektes abzubilden (unterschrieben, aktiv, still gelegt, abgeschlossen) oder handelt es sich um ein neues Objekt <Vertrag> mit eigenen Zuständen?

10. Schritt: Zustandsbasierte Testszenarien entwerfen

Aus einem Zustandsdiagramm werden Testfälle abgeleitet, die aufgrund der erlaubten Reihenfolgen zu Testszenarien zusammengeführt werden. Die Menge der zustandsbasierten Testfälle zu einem Zustandsdiagramm müssen folgende Forderungen erfüllen:

- Jeder Zustand wird erreicht.
- Jedes Ereignis wird ausgelöst.
- Jeder erlaubte Zustandsübergang wird ausgeführt.
- Es wird nachgewiesen, dass jeder nicht erlaubte Zustandsübergang nicht ausgeführt werden kann.

Zustandsdiagramme können sich über mehrere Anwendungsfälle (s. Abschnitt 4.1.5) erstrecken und sogar komplette Geschäftsprozesse abbilden. Daher sind zustandsbasierte Testszenarien nicht nur für die Tests von Objekten im Klassentest (Kap. 8.1) und im Komponententest (Kap. 8.2) relevant, sondern auch für die Tests von Geschäftsprozessen im funktionalen Systemtest (Kap. 8.4).

4.1.5 Anwendungsfallbasierte Testfallermittlung

Eine Möglichkeit, Testfälle auf Basis eines objektorientierten Software-Entwurfs zu erstellen, ist die Analyse von Anwendungsfällen.

anwendungsfallbasierte Testfallermittlung

Die anwendungsfallbasierte Testfallermittlung entwirft anhand von Anwendungsfalldiagrammen Testszenarien zur Überprüfung der vollständigen und korrekten Umsetzung der spezifizierten Anwendungs- und Geschäftsvorfälle in einem Software-System.

Anwendungsfall

Ein Anwendungsfall (Use Case) beschreibt Interaktionen, die zwischen Akteuren und dem betrachteten System stattfinden, um ein bestimmtes fachliches Ziel zu erreichen. Ein Anwendungsfall wird durch einen Akteur angestoßen und führt zu einem konkreten Ergebnis für die handelnden Akteure.

Anwendungsfalldiagramm

Mehrere voneinander abhängige oder in Beziehung stehende Anwendungsfälle bilden einen Geschäftsprozess und werden in einem Anwendungsfalldiagramm dargestellt, deren einzelne Elemente im Rahmen der Anforderungsbeschreibung detailliert spezifiziert werden. Ein Anwendungsfalldiagramm (Use Case Diagram) ist eine

Diagrammart der UML[3] und zeigt die Zusammenhänge der Anwendungsfälle mit Akteuren, untereinander und zum Systemumfeld.

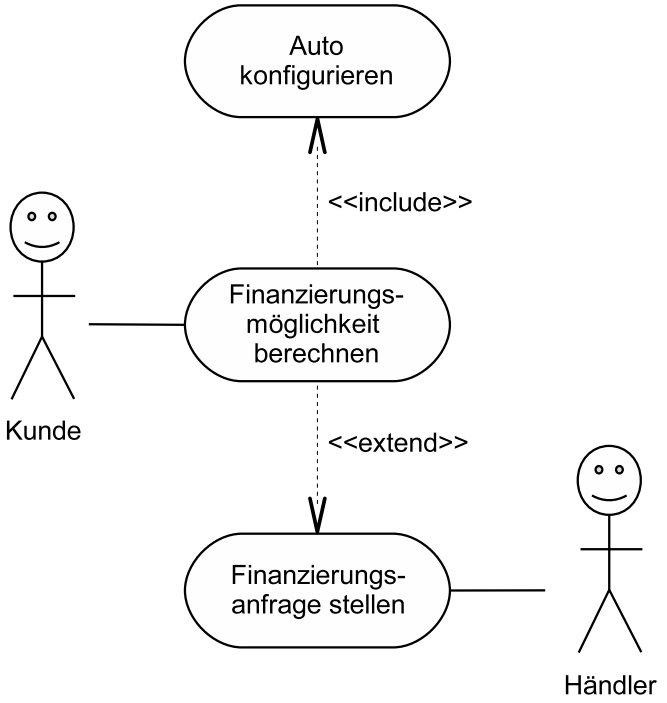


Abb. 4.3: Anwendungsfall-diagramm

Ein Anwendungsfalldiagramm zu unserem Finanzierungsrechner ist in Abb. 4.3 dargestellt. Darin werden die drei Anwendungsfälle „Auto konfigurieren", „Finanzierungsmöglichkeit berechnen" und „Finanzierungsanfrage stellen" mit ihren Beziehungen abgebildet. Beziehungen bestehen zu den beiden Akteuren „Kunde" und „Händler" sowie untereinander in einer <<include>>-Beziehung und einer <<extend>>-Beziehung. Eine <<include>>-Beziehung muss immer bestehen und eine <<extend>>-Beziehung ist optional.

11. Schritt: Anwendungsfallbasierte Testszenarien ergänzen

Auf Basis der Anwendungsfälle und ihrer Beziehungen können Testfälle entwickelt werden. Allerdings sind nicht alle Informationen, die für die Realisierung und die Testdurchführung notwendig sind, allein aus dem Anwendungsfalldiagramm ersichtlich. Details zu einem Anwendungsfall wie Ausgangsbedingungen, Bedingungen

[3] Unified Modeling Language (UML) ist eine standardisierte Sprache zum objektorientierten Software-Engineering ([URL: UML-OMG]).

für <<extend>>-Beziehungen oder Nachbedingungen sind in den fachlichen Spezifikationen zu entnehmen, die zu jedem Anwendungsfall beschrieben werden müssen.

Die anwendungsfallbasierten Testszenarien sind besonders für den Systemtest und den Abnahmetest interessant, weil sie die fachlichen Prozesse überprüfen und Systemzusammenhänge darstellen. Im Abschnitt 8.4.1 ist ein Beispiel eines anwendungsfallbasierten Testszenarios zum funktionalen Systemtest beschrieben.

4.1.6 Werkzeuge für die Testspezifikation

Probleme der Testdokumentation

Die Dokumentation von Testfällen und Testszenarien kann in einem Text- oder Tabellenkalkulationsprogramm erfolgen. Bei komplexen Anwendungen wird es für ein Projekt allerdings sehr schwer, die Testdokumentation einheitlich und aktuell zu halten. Besonders die Verknüpfungen von einer Äquivalenzklasse zum Testfall, vom Testfall zum Testszenario, vom Testszenario zum Testdatensatz und eventuell vom Testfall zu einem gefundenen Fehler sind manuell kaum abzubilden. Solche Referenzen werden aber für die Nachvollziehbarkeit der Tests und den Nachweis des Testfallabdeckungsgrades benötigt.

Weitere Probleme bei der manuellen Testdokumentation liegen darin, umfangreiche Entscheidungstabellen übersichtlich darzustellen, daraus Testszenarien zu generieren und alle Dokumente bei Änderungen konsistent zu halten.

Daher ist es sinnvoll, Testtools einzusetzen, die eine lückenlose Dokumentation ermöglichen und den Tester in seiner Arbeit unterstützen.

CaseMaker

Zum methodenunterstützten Testfallentwurf gibt es einige kommerzielle Werkzeuge. Eines davon ist CaseMaker, das die Pairwise-Methode unterstützt und unter bestimmten Voraussetzungen aus Zustandsdiagrammen Testfälle generieren kann ([URL: CaseMaker]).

ALLPAIRS

Ein Open Source Tool unter GPL 2.0[4], dass sich auf die Pairwise-Methode zur Reduzierung von Entscheidungstabellen konzentriert, ist ALLPAIRS von James Bach ([URL: ALLPAIRS]).

weitere Testspezifikationswerkzeuge

Weitere Tools zur Testspezifikation sind zum Beispiel bei [URL: TestToolsFAQ] in der Rubrik „Test Drivers and Test Suite Management Tools“ zu finden.

[4] GNU General Puplic License Version 2, Juni 1991

4.1.7 Qualitätsanforderungen zu den Blackbox-Verfahren

Anforderungsabdeckung

Eine Qualitätsanforderung an die Blackbox-Verfahren ist die Anforderungsabdeckung. Der Grad der Anforderungsabdeckung gibt an, wie viel Prozent der definierten Anforderungen durch Testfälle überprüft werden.

Eine Anforderungsabdeckung von 100% bedeutet, dass für jede Äquivalenzklasse, jeden Grenzwert und jede Bedingungskombination mindestens ein Testfall festgelegt ist. Für ein Zustandsdiagramm sind Testfälle nachzuweisen, die sicherstellen, dass jeder Zustand erreicht, jedes Ereignis ausgelöst, jeder erlaubte Zustandsübergang ausgeführt wird und jeder nicht erlaubte Zustandsübergang nicht möglich ist.

Testfallabdeckung

Der Testfallabdeckungsgrad ist ein Qualitätsmaß zur Testdurchführung. Er gibt an, wie viel Prozent der definierten Testfälle nachweislich im Test ausgeführt wurden.

Damit der Testaufwand eines Projektes im budgetierten Rahmen bleibt, sollten Testfallabdeckungsgrade pro Risikoklasse festgelegt werden. Dazu müssen alle Testobjekte einer Risikoklasse zugeordnet werden. Für jede Risikoklasse wird ein bestimmter, nachzuweisender Testfallabdeckungsgrad festgelegt, wie zum Beispiel[5]:

- Risikoklasse A: 100% Testfallabdeckung
- Risikoklasse B: 80% Testfallabdeckung
- Risikoklasse C: 50% Testfallabdeckung

Der Nachweis der Erfüllung eines geforderten Testfallabdeckungsgrades zu einem Testobjekt kann natürlich erst vorgelegt werden, nachdem die Tests mit den beschriebenen Testfällen durchgeführt wurden.

[5] Im Kapitel 5.3 sind in Tabelle 5.2 Risikoklassen für Programme definiert, die für das beschriebene Beispiel angewendet werden können.

4.1.8 Empfehlungen zu den Blackbox-Verfahren

Testfälle früh entwerfen

Die Entwicklung von Testfällen nach den Blackbox-Verfahren ist eine sehr effektive Qualitätssicherungsmaßnahme. Beim Testfallentwurf muss jedes Detail zur Eingabe, Verarbeitung und Ausgabe eines Testobjekts vollständig beschrieben werden, so dass fehlerhafte oder fehlende Anforderungen sofort offensichtlich werden. Daher muss es ein Ziel der Qualitätssicherung sein, Testfälle möglichst früh zu entwerfen.

Testfälle genehmigen lassen

Die mit den Blackbox-Verfahren entworfenen Testszenarien sollten einem Abnahme-Review[6] durch Vertreter der Fachabteilung und des Entwicklerteams unterzogen werden. Das hat zwei gute Gründe:

Erstens bestätigt die Fachabteilung mit der Abnahme der Testszenarien die fachliche Richtigkeit und Vollständigkeit der geplanten Tests und somit indirekt die der Anforderungskonzepte. Sind die Testfälle falsch, so sind entweder die Anforderungen nicht hinreichend beschrieben oder die Testfälle sind nicht richtig. In beiden Fällen müssen Verbesserungen vorgenommen werden.

Zweitens bestätigt das Entwicklerteam durch die Abnahme der Testszenarien, dass die Anforderungen realisierbar sind und die Testfälle der Realisierung nicht widersprechen. Aufgrund des gemeinsamen Verständnisses der Anforderungen und der geplanten Tests werden Fehler bei der Programmierung vermieden.

Testfälle bewerten

Um sicherzustellen, dass die kritischen Tests zuerst durchgeführt werden, sollten Testfälle und Testszenarien einer Risikoklasse zugeordnet werden. Von Fachleuten wird festgelegt, welche Testszenarien bei den Tests unbedingt durchgeführt werden müssen und welche bei Zeit- oder Budgetmangel – mit bekanntem Risiko – vernachlässigt werden können.

Testfälle mehrfach verwenden

Testfälle sollten bis zum Beginn der Programmierung beschrieben sein und den Entwicklern als konstruktive Qualitätssicherungsmaßnahme zur Verfügung gestellt werden. Auf diese Weise kann das Entwicklerteam in seiner Entwicklungsumgebung eine große Anzahl der Tests mit den Testfällen des Testteams durchführen. Die Entwickler benötigen ihrerseits weniger Zeit für den Entwurf von Testfällen und ersparen bei entsprechender Nachweisführung dem Testteam einige Tests (s. Kap. 8.1, Klassentest).

---

[6] Zur Durchführung von Reviews s. Kap. 7.1.

4.2 Whitebox-Verfahren

Bei der Ermittlung der Testfälle an Hand von Blackbox-Methoden bleiben Programminterna unberücksichtigt. Nicht so bei den Whitebox-Methoden, bei denen Programmstrukturen analysiert werden. Zudem stellen sich im Rahmen der Whitebox-Verfahren interessante Fragen nach der Qualität der durchgeführten Testfälle:

- Gibt es Programmteile, die beim Test nicht durchlaufen wurden?
- Gibt es Programmcode, der nie durchlaufen werden kann?

4.2.1 Testabdeckungsgrade

Testabdeckungsgrade

Testabdeckungsgrade – auch Testüberdeckungsgrade genannt – beantworten diese Fragen. Die Messung eines Testabdeckungsgrades (Logic Coverage Test, Code Coverage Test) stellt fest, welcher Code bzw. wie viel Prozent des Codes eines Programms mit den durchgeführten Tests durchlaufen wurde.

Beim Testen können verschiedene Testabdeckungsgrade mit unterschiedlicher Aussagekraft bestimmt werden. Am häufigsten werden die Anweisungs-, Zweig- und Bedingungsabdeckung als Testmetrik eingefordert, weil sie mit akzeptablem Aufwand nachweisbar sind.

Anweisungsabdeckung

Anweisungsabdeckung (C0, Statement Coverage):
Jede Anweisung wird mindestens einmal ausgeführt.

Zum Beispiel bedeutet eine Anweisungsabdeckung von 100%, dass alle ausführbaren Anweisungen beim Testen auch ausgeführt wurden. Wenn sich beim Test herausstellt, dass ein Teil des Codes nicht durchlaufen wurde, kann das zwei Ursachen haben. Entweder sind die Testfälle unzureichend, dann müssen zusätzliche definiert werden, oder der Programmcode ist falsch, so dass die betreffenden Statements nie erreicht werden können. In diesem Fall muss der Programmcode angepasst werden.

Zweigabdeckung

Zweigabdeckung (C1, Decision and Branch Coverage):
Jeder Programmzweig und jeder Sprung wird einmal ausgeführt. Jedes mögliche Ergebnis einer Entscheidung wird mindestens einmal herbeigeführt.

Einfache Bedingungs-abdeckung

Einfache Bedingungsabdeckung (C2, Branch Coverage):
Jedes Entscheidungskriterium einer Bedingung nimmt mindestens einmal den Wert „wahr“ und „falsch“ ein.

Die C2-Testbadeckung ist als alleiniges Qualitätssicherungsmaß nicht befriedigend, weil weder die Zweig- noch die Anweisungsabdeckung in diesem Maß enthalten sind.

Mehrfache Bedingungs-abdeckung

Mehrfache Bedingungsabdeckung (C3, Condition Coverage):
Alle möglichen Kombinationen der Entscheidungskriterien einer Bedingung werden durchlaufen.

Beispiel für Testabdeckungsgrade

Ein Beispiel soll die Bedeutung und die Unterschiede der Testabdeckungsgrade verdeutlichen. Dazu betrachten wir die folgende Anforderung:

Einem Kunden wird beim Kauf eines Autos ein Rabatt von 3% gewährt, wenn für ihn das Stammkundenkennzeichen gesetzt ist oder der Gesamtpreis für die Bestellung über 30.000 Euro liegt. Die Anforderung ist in Tabelle 4.10 in Pseudocode umgesetzt.

Tabelle 4.10: Pseudocode

Pseudocode
if (STAMMKUNDE = “JA“ or GESAMTPREIS > 30.000)
then
RABATT = 3%
endif

Beispiel: C0 – Anweisungsabdeckung
Die Forderung der Anweisungsabdeckung bedeutet, dass jedes ausführbare Statement mindestens einmal durchlaufen wird. Um das nachweisen zu können, legen wir den ersten Testdatensatz fest, der das Stammkundenkennzeichen mit „JA“ und einen Gesamtpreis von 10 Euro enthält:

Testdatensatz	1.
STAMMKUNDE	“JA“
GESAMTPREIS	10

Mit diesem Testdatensatz spielen wir Testtool, d.h. wir versehen diejenigen Anweisungen mit einem „X“, an denen wir auf Grund der Testdaten vorbeikommen (Tabelle 4.11).

Anweisungsabdeckung	1.
if (STAMMKUNDE = “JA“	
or	X
GESAMTPREIS > 30.000)	
then	
RABATT = 3%	X
endif	

Tabelle 4.11: Anweisungsabdeckung

Die 100%ige C0-Testabdeckung ist für unser Beispiel nachgewiesen, weil jedes ausführbare Statement einmal ausgeführt wurde.

Beispiel: C1 – Zweigabdeckung

Ein Problem ergibt sich für den Nachweis der C1-Testabdeckung. Wie kann von unserem Tool nachgewiesen werden, ob der „Nein“-Zweig der Entscheidung jemals durchlaufen wird? Ein „X“ im „Nein“-Zweig kann nicht gesetzt werden, weil er keine ausführbare Anweisung enthält. Hier wird deutlich, dass die Testabdeckung von der Art der Programmierung und dem Testtool abhängt. Es gibt zwei Lösungen für das Problem:

Entweder wird eine Programmiervorgabe erlassen, die „tote“ Zweige im Code verbietet, oder es wird ein intelligentes Werkzeug zur Messung der Testabdeckung eingesetzt, das beim Instrumentieren des Programms einen solchen Zweig erkennt und ihn automatisch ergänzt.

In unserem Beispiel entscheiden wir uns für die erste Lösung. Das heißt, die Anweisung CONTINUE[7], die „nichts tut“, wird in den Programmcode eingebaut (Tabelle 4.12).

Zweigabdeckung	1.
if (STAMMKUNDE = “JA“	
or	X
GESAMTPREIS > 30.000)	
then	
RABATT = 3%	X
else	
CONTINUE	
endif	

Tabelle 4.12: Zweigabdeckung (1)

[7] Was „alte“ COBOL-Kenner erfreuen wird.

Zurück zum ersten Testdatensatz. Mit ihm wird beim Test eine Zweigabdeckung von 50% erreicht, weil nur der „Ja“-Zweig des Codesegments durchlaufen wird (s. Tabelle 4.12). Für eine 100%-Zweigdeckung wird ein zweiter Testdatensatz benötigt:

Testdatensatz	1.	2.
STAMMKUNDE	“JA“	„NEIN“
GESAMTPREIS	10	10

Mit diesen beiden Testdatensätzen notiert unser Testtool eine Anweisungsabdeckung von 100% (Tabelle 4.13).

Tabelle 4.13: Zweigabdeckung (2)

Zweigabdeckung	1.	2.
if (STAMMKUNDE = “JA“		
or	X	X
GESAMTPREIS > 30.000)		
then		
RABATT = 3%	X	
else		
CONTINUE		X
endif		

Beispiel: C2 – einfache Bedingungsabdeckung

Soll für unser Miniprogramm die C2-Testabdeckung nachgewiesen werden, so muss jede Einzelentscheidung in der if-Abfrage mindestens einmal wahr und einmal falsch sein. Das erreichen die beiden folgenden Testdatensätze, wobei der erste aus den Testdaten für den Nachweis der Zweigabdeckung stammt:

Testdatensatz	1.	3.
STAMMKUNDE	„JA“	“NEIN“
GESAMTPREIS	10	44.444

Unser Testtool notiert mit diesen Daten hinter jedem Entscheidungskriterium ein „w“ für wahr oder ein „f“ für falsch (s. Tabelle 4.14). Wie in Tabelle 4.14 zu erkennen ist, wird der zweite Testdatensatz (STAMMKUNDE = „NEIN“, GESAMTPREIS = 10) zur einfachen Bedingungsabdeckung nicht benötigt.

Das zeigt, wie schwach dieses Testmaß ist, weil die C1-Testdeckung nicht in der C2-Testabdeckung enthalten ist. Die einfa-

che Bedingungsabdeckung sollte daher immer in Verbindung mit der Zweigabdeckung gefordert werden.

Einfache Bedingungsabdeckung	1.	3.
if (STAMMKUNDE = “JA“	w	f
or		
GESAMTPREIS > 30.000)	f	w
then		
RABATT = 3%	X	X
else		
CONTINUE		
endif		

Tabelle 4.14: einfache Bedingungs-abdeckung

Beispiel: C3 – mehrfache Bedingungsabdeckung

Wenn der Test für unser Beispielprogramm die mehrfache Bedingungsabdeckung C3 liefern soll, d.h. alle Kombinationen der Einzelentscheidungen der if-Abfrage getestet werden sollen, wird ein vierter Testdatensatz benötigt:

Testdatensatz	1.	2.	3.	**4.**
STAMMKUNDE	“JA“	„NEIN“	„NEIN“	**“JA“**
GESAMTPREIS	10	10	44.444	**44.444**

Der Testlauf mit diesen vier Testdatensätzen weist für unser Programm den C3-Testabdeckungsgrad nach, der die C0-, C1- und C2-Abdeckungsgrade einschließt (Tabelle 4.15).

Mehrfache Bedingungsabdeckung	1.	2.	3.	4.
if (STAMMKUNDE = “JA“	w	f	f	w
or				
GESAMTPREIS > 30.000)	f	f	w	w
then				
RABATT = 3%	X		X	X
else				
CONTINUE		X		
endif				

Tabelle 4.15: mehrfache Bedingungs-abdeckung

4.2.2 Weitere Testabdeckungsgrade

minimale Mehrfachbedingungsabdeckung

Bei komplexen Bedingungen kann es vorkommen, dass die elementaren Bedingungen von einander abhängen und nicht alle Kombinationen für den C3-Testabdeckungsgrad mit Testdaten erreicht werden können. In diesem Fall werden nur die elementaren Bedingungen in Testfällen berücksichtigt, die sich im Ergebnis des Gesamtausdruckes auswirken. Bei einer solchen Reduzierung der möglichen Bedingungskombinationen wird von einer minimalen Mehrfachbedingungsabdeckung gesprochen.

Pfadabdeckung und Schleifenabdeckung

Neben den beschriebenen Testabdeckungsmaßen gibt es noch weitere, wie zum Beispiel die Pfadabdeckung (Path Coverage) oder die Schleifenabdeckung (Loop Coverage). Diese sind aber aufgrund des hohen Testaufwandes nicht praktikabel und werden hier nicht weiter betrachtet.[8]

kontrollflussbasierter Test

Die bisher besprochenen, aus der „alten", nicht objektorientierten Programmierwelt bekannten Testabdeckungsgrade, begründen sich in den Kontrollflüssen eines Programms (Zweige, Bedingungen, Pfade,...). Auf die objektorientierten Programmiersprachen übertragen, sind diese Metriken nicht befriedigend, weil die Besonderheiten der Objektorientierung von ihnen nicht berücksichtigt werden.

Method Coverage

Die Testmaße C0 bis C3 können auf den Programm-Code jeder Methode einer Klasse angewendet werden. Mit dem Nachweis einer dieser Metriken ist auch jede Methode mindestens einmal ausgeführt worden (Method Coverage), was im Klassentest (Kap. 8.1) als Qualitätssicherungsnachweis gefordert wird.

Schwierig wird ein vollständiger Test von objektorientierten Software-Komponenten durch die Kapselung und Vererbung von Attributen und vielen kleinen[9] Methoden sowie die komplexen Kommunikationsmöglichkeiten unter den verschiedenen Objekten. Objektorientierte Abdeckungsgrade finden in der Praxis (noch?) geringe Anwendung, wie zum Beispiel:

Attribute Coverage

- Jedes Objektattribut wird mindestens einmal modifiziert (Attribute Coverage).

[8] Die minimale Mehrfachbedingungsabdeckung und die Pfadabdeckung sind zum Beispiel in [Spillner_2005] beschrieben.

[9] Klein im Vergleich mit den Code-Sequenzen prozessorientierter Programmiersprachen.

- Jede Nachricht wird mindestens einmal gesendet (Interface Coverage). *Interface Coverage*
- Jeder Parameter jeder Nachricht wird mindestens einmal modifiziert (Parameter Coverage). *Parameter Coverage*

4.2.3 Werkzeuge für die Testabdeckungsgradmessung

Code Coverage Tools

Der Nachweis von Testabdeckungsgraden ist ohne ein Code Coverage Tool, das es für jede gängige Programmiersprache gibt, nicht praktikabel. Kommerzielle Tools sind unter [URL: TestToolsFAQ] in der Rubrik „Test Coverage“ zu finden. Speziell für Java gibt es Open Source Tools unter [URL: TestToolsJava] in der Rubrik „Code Coverage“.

Nachweis der Testabdeckung in 4 Schritten

Der werkzeugunterstützte Nachweis einer Testabdeckung erfolgt in vier Schritten:

1. Schritt: Der Sourcecode wird beim Kompilieren vom Code Coverage Tool instrumentiert, d.h. er wird so erweitert, dass die ausgeführten Anweisungen, Entscheidungen und/oder Entscheidungskriterien protokolliert werden können.
2. Schritt: Alle nach den Blackbox-Verfahren definierten Testszenarien werden mit den instrumentierten Programmen ausgeführt, wobei die Testdurchläufe vom Code Coverage Tool protokolliert werden. Für die erste Messung des Testabdeckungsgrades werden zunächst einmal keine zusätzlichen Testfälle benötigt, denn wenn die bis dahin ermittelten Testfälle richtig und vollständig sind, wird mit ihnen die geforderte Testabdeckung eventuell schon erreicht.
3. Schritt: Sind alle Funktionstests durchgeführt, wird anhand des Testprotokolls überprüft, ob die gewünschte Testabdeckung erreicht wurde. Nehmen wir der Einfachheit halber C0, d.h. jedes ausführbare Statement sollte durchlaufen worden sein. Wurde eine Anweisung nicht durchlaufen, so kann es dafür drei Erklärungen geben:

   3.a) Die Testfälle sind nicht vollständig, denn ein neuer Testfall, der mit seinen Testdaten die fehlende Programmanweisung erreicht, kann gefunden werden. *unvollständige Testfälle*

   3.b) Es gibt überflüssigen Programmcode, weil kein Testfall und somit kein Testdatensatz gefunden werden kann, der die bisher nicht ausgeführte Anweisung erreicht. *überflüssiger Programmcode*

Programmierfehler

3.c) Der kritische Fall, dessen Eintreten den Aufwand für die Testabdeckungsgradmessung rechtfertigt, liegt in der dritten Möglichkeit. Es gibt einen Testfall, der das nicht ausgeführte Statement eigentlich hätte erreichen müssen. Hier liegt ein Programmierfehler vor, der in den vorhergehenden Tests nicht gefunden wurde.

4. Schritt: Anhand der Analyseergebnisse aus dem 3. Schritt werden Testfälle und Testdaten ergänzt und Programmfehler berichtigt. Anschließend werden die Tests wiederholt. Hier ist es von Vorteil, wenn die Tests mit Testrobotern automatisiert sind (s. Abschnitt 12.4.2).

4.2.4 Qualitätsanforderungen zu den Whitebox-Verfahren

Testabdeckungsgrade als Testendekriterien

Die im Qualitätssicherungsplan eines Projektes zu definierenden Testabdeckungsgrade beschreiben Qualitätsanforderungen an die durchgeführten Tests und stellen zugleich Testendekriterien dar. Zum Beispiel können folgende Qualitätsanforderungen für Programmmodule festgelegt werden, für die allerdings eine Risikoanalyse (Kap. 5) Voraussetzung ist:

- Für alle Module der Risikoklasse A ist eine C1-Testabdeckung von 100% nachzuweisen.
- Für alle Module der Risikoklasse B und C ist eine C1-Testabdeckung von 80% nachzuweisen.
- Sollte die 100%-Testabdeckung für ein Modul aus technischen Gründen nicht nachweisbar sein, sind die nicht durchlaufenen Programmteile zu dokumentieren und sowohl vom Projektleiter als auch vom Qualitätssicherungsverantwortlichen zu genehmigen.

↓

Exkurs

Eine Erfahrung zum C1-Testabdeckungsgrad

Die kompliziert klingende Formulierung zum letzten Punkt der Qualitätsanforderungen beruht auf einer Projekterfahrung:

In einem Projekt war laut vertraglich bindendem Qualitätssicherungsplan eine C1-Testabdeckung von 100% nachzuweisen. In jedes Programm mussten aber standardisierte Code-Blöcke zur Fehlerbearbeitung und zur Kommunikation mit dem verwendeten Transakti-

onsmonitor eingebunden werden. Diese Programmergänzungen bewirkten, dass je nach Umfang des neu programmierten Codes nur eine Testabdeckung zwischen 60% und 80% erreicht werden konnte. Der Grund lag darin, dass die Code-Blöcke mit „normalen“ Testdaten nicht durchlaufen wurden. Zudem sollten sie im Rahmen der anstehenden fachlichen Tests auch nicht mitgetestet werden, weil sie schon viele Jahre produktiv im Einsatz waren.

Die ursprüngliche Qualitätsanforderung, 100% Zweigabdeckung, war somit nicht haltbar. Der Qualitätssicherungsplan musste überarbeitet werden und erhielt den Zusatz: „Sollte aus technischen Gründen die 100%-Abdeckung für ein Modul nicht nachweisbar sein,...“.

Tipp

Moral von der Geschicht': Qualitätsanforderungen können von Rahmenbedingungen abhängig sein und müssen genau überprüft werden.

↑

4.2.5 Empfehlungen zu den Whitebox-Verfahren

Testabdeckung im Komponententest

Der Nachweis von Testabdeckungsgraden sollte in der Teststufe Komponententest durchgeführt werden, weil der Sourcecode bereitgestellt werden muss und eine enge Zusammenarbeit mit den Entwicklern erforderlich ist. Zudem sollten keine instrumentierten Programme, wie sie zum Nachweis der Testabdeckung benötigt werden, in den Integrationstest gegeben werden, weil die Instrumentierung anwendungsfremden Code erzeugt und Einfluss auf die Performanz haben kann.

Testabdeckung nur für kritische Module

Testabdeckungsgrade sollten nur für Module mit hoher Risikobewertung bestimmt werden, da der Aufwand für die Whitebox-Tests nicht zu unterschätzen ist und im Verhältnis zum Nutzern stehen sollte.

C1 und C2 sind praktikabel

Weil die Anweisungsabdeckung ein schwaches Testmaß ist und sich Werkzeuge mit der mehrfachen Bedingungsabdeckung schwer tun, ist der Nachweis der Zweigabdeckung in Verbindung mit der einfachen Bedingungsabdeckung eine praktikable Qualitätsanforderung. Manueller Aufwand sollte hierbei vermieden werden, d.h. ein Code Coverage Tool sollte eingesetzt werden.

4.3 Fehlererwartung

Neben den Blackbox- und Whitebox-Methoden zur systematischen Testfallermittlung darf beim Testen der gesunde Menschenverstand nicht vergessen werden.

Fehlererwartung

Der erprobte Tester weiß auf Grund seiner Erfahrungen um die Schwachstellen bei der Programmierung und beschreibt Testfälle für Situationen, bei denen mit einer gewissen Wahrscheinlichkeit Fehler gemacht werden (können). Drei Beispiele für intuitive Testfälle:

Berechnungen

Für unseren Finanzierungsrechner aus dem Kapitel 4.1 wäre zum Beispiel interessant zu testen, welches der größt mögliche Kaufpreis ist, der vom Autokonfigurator übergeben wird. Kann der Finanzierungsrechner diese Zahl korrekt darstellen und verarbeiten? Gibt es vielleicht Zahlenüberläufe oder Rundungsfehler?

Browser-Navigation

Ein meist erfolgreicher Fehlererwartungstestfall für browser-basierte Anwendungen ist das Aktualisieren des Browser-Fensters während der Bearbeitung einer Transaktion[GL]. Was passiert zum Beispiel, wenn beim Finanzierungsrechner statt der Funktion <Berechnen> die Refresh-Funktion des Browsers ausgeführt wird? Oft passiert es, dass ein Vorgang durch den Aktualisierungs-Button des Browsers unkontrolliert abgebrochen wird.

Zeitzonen

Interessant ist die Reaktion einer Web-Applikation, wenn sie von einem Client aufgerufen wird, der in einer anderen Zeitzone als der Web-Server arbeitet. „Verträgt" das System unterschiedliche Zeiteinstellungen?

Damit das Wissen der erfahrenen Tester der Allgemeinheit zur Verfügung gestellt werden kann, sollte von der Qualitätssicherung eine Liste der „beliebten" und bekannten Fehlerquellen erstellt und in Form einer Checkliste fortgeschrieben werden. Eine Checkliste zur Fehlererwartung enthält zum Beispiel folgende Punkte:

✓ Checkliste Fehlererwartung

- Versuche, Divisionen durch Null zu erzeugen.
- Versuche, Zahlenüberläufe zu erzeugen.
- Versuche, Rundungsfehler zu erzeugen.

- Drücke in beliebigen Situationen den Refresh-Button des Browsers.
- Führe in kritischen Situationen die Navigationsfunktionen des Browser aus (Vor- und Zurückblättern).
- Führe mehrmals hintereinander dieselbe Funktion aus, ohne dazwischen andere Aktionen durchzuführen.
- Rufe die Web-Anwendung zeitzonen- und länderübergreifend auf.

Anhand einer solchen Checkliste werden Testfälle beschrieben, welche die mit den Blackbox- und Whitebox-Methoden entworfenen ergänzen.

4.4 Zusammenfassung

- Blackbox- und Whitebox-Verfahren stellen Methoden zur systematischen Testfallerstellung bereit.
- Mit den Blackbox-Verfahren werden Testfälle aus den Anforderungen abgeleitet, ohne dass die Programmstrukturen des Testobjektes untersucht werden.
- Zu den Blackbox-Verfahren gehören die Äquivalenzklassenanalyse, die Grenzwertanalyse, die Ursache-Wirkungs-Analyse sowie die Analyse von Zustandsdiagrammen und Anwendungsfalldiagrammen.
- Mit der Ursache-Wirkungs-Analyse werden Testszenarien aus Kombinationen gültiger Äquivalenzklassen abgeleitet und in Entscheidungstabellen dargestellt.
- Mit der Pairwise-Methode werden komplexe Entscheidungstabellen systematisch reduziert.
- Die Testfallerstellung mit Hilfe von Blackbox-Methoden ist eine effiziente Qualitätssicherungsmaßnahme, denn sie setzt vollständige und detaillierte Anforderungsbeschreibungen voraus und deckt bei der Beschreibung der erwarteten Testergebnisse schnell Mängel in den Spezifikationen auf.
- Mit Whitebox-Verfahren werden Testfälle aus Programmstrukturen abgeleitet.
- Testabdeckungsgrade geben Auskunft über die Qualität der durchgeführten Tests. Die Methoden-, Anweisungs-, Zweig-

und Bedingungsabdeckung haben am meisten Bedeutung erlangt.

- Der Nachweis der Testabdeckungsgrade kann nur mit Code Coverage Tools effizient erbracht werden.
- Die Messung von Testabdeckungsgraden sollte nur für Komponenten mit einer hohen Risikoeinstufung vorgenommen werden, weil ihr Aufwand nicht unerheblich ist.
- Bei der intuitiven Methode der Fehlererwartung beschreiben erfahrene Tester Testfälle für Situationen, in denen erwartungsgemäß Fehler auftreten. Diese „beliebtesten“ Fehler werden in einer regelmäßig zu überarbeitenden Checkliste festgehalten.
- Testfälle und Testszenarien werden mit Testmanagementwerkzeugen dokumentiert, priorisiert und verwaltet.

5 Risikoanalyse

„Manchmal muss man einfach ein Risiko eingehen - und seine Fehler unterwegs korrigieren."

Lee Iacocca (*1924)

In termingedrängten Projekten wird die Vielfalt der Qualitätssicherungsaufgaben und der Aufwand der durchzuführenden Qualitätssicherungsmaßnahmen oft unterschätzt. Als Folge davon sind voreilige und unkontrollierte Freigaben von Software das Gebot der Stunde, Hauptsache Termine halten. Bisher sind aber alle mir bekannten, vom Management verordneten „Risikofreigaben" schief gelaufen, d.h. der vermeintlich eingesparte Testaufwand kam stets in Form von Produktionsfehlern und -ausfällen als Boomerang zurück.

Risikoanalysen sorgen dafür, dass die Freigabe einer Anwendung kein generelles Risiko darstellt, sondern die Risiken bei einer Freigabe bekannt und kalkulierbar sind.

5.1 Ziele der Risikoanalyse

Wenn man sich die technische Komplexität einer Web-Applikation und die sich daraus ergebenden Testvarianten vor Augen führt, wird deutlich, wie viel Fehler- und somit Risikopotential darin stecken. Um das Risiko einer Software-Freigabe einschätzen und die Balance zwischen Testaufwand und Kosten in einem Projekt halten zu können, müssen Risikoanalysen Bestandteil der Projektplanung sein und zielgerichtet durchgeführt werden.

Mit der Risikoanalyse werden Risiken transparent gemacht, geeignete Qualitätssicherungsmaßnahmen abgeleitet sowie Testfälle und Fehler priorisiert. Zudem stellt die Risikoanalyse sicher, dass die wichtigsten Tests zuerst durchgeführt werden, falls der Testaufwand aus Zeit- oder Budgetgründen reduziert werden muss.

5.2 Grundlagen der Risikoanalyse

Risiko

Ein Risiko ist die Möglichkeit des Eintretens eines Ereignisses mit negativen Auswirkungen. In der Mathematik wird das Risiko definiert als Eintrittswahrscheinlichkeit des negativen Ereignisses multipliziert mit der potentiellen Schadenshöhe:

Risiko = Eintrittswahrscheinlichkeit x Ausmaß

Risikoprioritätszahl – RPZ

Die Fehlermöglichkeits- und Einflussanalyse (FMEA[1] - Failure Mode and Effects Analysis) misst ein Risiko durch die Risikoprioritätszahl (RPZ), indem sie das Risiko mit der Entdeckungswahrscheinlichkeit bewertet:

RPZ = Eintrittswahrscheinlichkeit x Ausmaß x
Wahrscheinlichkeit der Nichtentdeckung

Jeder der drei Faktoren eines Risikos erhält in der FMEA eine Bewertung zwischen 1 und 10, so dass jedes Risiko eine RPZ zwischen 1 und 1000 besitzt. In Abhängigkeit der Höhe der RPZ werden Maßnahmen festgelegt, mit denen dem Risiko begegnet wird.

5.3 Risikoanalyse in der Software-Entwicklung

Projekt- und Produktrisiken

In einem Software-Entwicklungsprojekt müssen Projektrisiken (zum Beispiel die Beistellungen des Auftraggebers oder die Verfügbarkeit von Ressourcen) und Produktrisiken (zum Beispiel die Fehler in Programmen) bewertet werden. In diesem Buch liegt der Fokus auf dem Produktrisiko des Produktes „Web-Applikation“.

Die Bestimmung der Risikoprioritätszahl ist für Software-Produkte nicht einfach, denn die Eintrittswahrscheinlichkeit und die

[1] Die FMEA wird in der DIN 25448 als Ausfalleffektanalyse beschrieben ([DIN 25448]).

Wahrscheinlichkeit der Nichtentdeckung eines nicht bekannten Fehlers sind – wenn überhaupt – sehr schwer zu bestimmen.

ABER: Durch systematische und ausführliche Tests kann die Eintrittswahrscheinlichkeit eines negativen Ereignisses, was in unserem Kontext das Eintreten einer Fehlerwirkung bedeutet, verringert werden. Um im Rahmen der Testplanung die notwendigen Qualitätssicherungsmaßnahmen ergreifen und richtigen Entscheidungen treffen zu können, muss das Ausmaß einer möglicherweise eintretenden Fehlerwirkung abgeschätzt werden. Dazu hat sich in IT-Projekten die ABC-Analyse bewährt, die eine dreistufige Bewertungsskala statt einer zehnstufigen benutzt.

ABC-Risikoanalyse

Durch eine ABC-Risikoanalyse werden die zu bewertenden Objekte in die Risikoklassen (Risikostufen) A, B oder C eingeteilt. Diesen drei Risikoklassen werden Maßnahmen zugeordnet, die dem Risiko entgegenwirken sollen. Die zu bewertenden Objekte und zugeordneten Maßnahmen können unterschiedlicher Natur sein, wie folgende Beispiele zeigen:

Testobjekte

- Testobjekte werden einer Risikoklasse zugeordnet, um angemessene Qualitätssicherungsmaßnahmen und Testmetriken festlegen zu können. Kritische Komponenten mit hoher Risikoklasse unterliegen strengeren Maß(nahm)en als unkritische.

Testfälle

- Testfälle werden bewertet, um festzulegen, mit welcher Priorität sie zu testen sind. Die wichtigsten Testfälle, d.h. Testfälle, die Fehlerwirkungen mit der höchsten potentiellen Schadenshöhe aufdecken können, müssen zuerst durchgeführt werden.

Fehler

- Mängel und aufgedeckte Fehler werden priorisiert, um Freigabeentscheidungen treffen zu können. Unkritische Mängel sollten nie den Projektfortschritt oder eine Freigabestufe verhindern.

Die beiden folgenden Beispiele beschreiben Definitionen von Risikoklassen, wie sie im Qualitätssicherungsplan eines Projektes festgelegt werden können.

1. Beispiel: Risikoklassen für Mängel in Dokumenten

Wenn in einem Review[2] zur Abnahme einer Anforderungsspezifikation Mängel festgestellt werden, so müssen diese bewertet werden, um die weiteren Schritte im Projekt festlegen zu können. In Tabelle 5.1 sind Merkmale festgelegt, mit denen ein in einem Dokument gefundener Fehler einer Risikoklasse zugeordnet wird.

[2] siehe Kap. 7.1, Dokumententest

Tabelle 5.1: Risikoklassen für Dokumente

Risikoklasse	Risikomerkmal
A - Hoch	Falsche oder unvollständige Beschreibung
	Fehlende Funktionalität oder fehlender Algorithmus
B - Mittel	Formulierungen, die falsch interpretiert werden können
C - Gering	Schreib- und Formfehler
	Fehlende Erläuterungen

Für jede Risikoklasse werden Maßnahmen und Konsequenzen festgelegt:

- Mängel mit der Risikoklasse A und B verhindern die Freigabe eines Dokumentes und der darauf basierenden Arbeitsergebnisse. Mängel der Risikoklasse A und B müssen vor der Freigabe behoben werden.
- Wird bei einem Review ein Fehler der Klasse A festgestellt, so muss nach Überarbeitung des Dokumentes ein erneutes Review durch Experten stattfinden. Für B-Fehler ist die Prüfung durch den Qualitätssicherungsverantwortlichen hinreichend.
- Mängel der Risikoklasse C können bis zu einem im Prüfprotokoll festgelegten Termin beseitigt werden, der nach dem Freigabetermin liegen kann. Die Nachprüfung erfolgt durch den Qualitätssicherungsverantwortlichen.

Die in diesem Beispiel beschriebenen Risikoklassen und damit verbundenen Qualitätssicherungsmaßnahmen werden im Dokumententest angewendet, siehe Kap. 7.

2. Beispiel: Risikoklassen für Programme

Ein fehlerhaft arbeitendes Programm stellt ein Risiko dar.

Tabelle 5.2: Risikoklassen für Programme

Risikoklasse	Risikomerkmal / Fehlerwirkung
A - Hoch	Geldwerte Nachteile entstehen.
	Schadenansprüche können gestellt werden.
	Aufträge gehen verloren.
B - Mittel	Kunden werden verärgert.
	Interessenten werden nicht gewonnen.
	Relevante Informationen werden nicht geliefert.
C - Gering	Mängel und Schönheitsfehler, die nicht geschäftsrelevant sind.

In Tabelle 5.2 sind Risikoklassen festgelegt, die durch das Ausmaß einer potentiellen Fehlerwirkung bestimmt werden. Im Rahmen der Risikoanalyse muss jedes Programm einer Risikoklasse zugeordnet werden, bevor es von der Qualitätssicherung geprüft wird.

Bewerten wir drei Beispielprogramme anhand dieser Risikoklassen:

1. Der Finanzierungsrechner, der durch einen Fehler falsche Finanzierungsraten berechnet, ist in die Risikoklasse A einzuordnen, weil ein finanzieller Schaden entstehen kann.
2. Das Programm, dass automatisch Informationsmaterial per E-Mail versendet, wird mit der Klasse B bewertet. Denn wenn ein Interessent keine Informationsunterlagen erhält, so ist das nicht gut für das Geschäft, aber es entsteht kein direkter finanzieller Schaden.
3. Die Wochenstatistik für das Management wird der Risikoklasse C zugeordnet, da ein Fehler in der Statistik keine negative Außenwirkung hat.[3]

Für jede der drei definierten Risikoklassen werden in Tabelle 5.3 die notwendigen Qualitätssicherungsmaßnahmen festgelegt, wie zum Beispiel der Nachweis des Testfallabdeckungsgrades (Abschnitt. 4.1.6), des Testabdeckungsgrades (Abschnitte 4.2.1 und 4.2.2) oder die Durchführung von Code-Inspektionen (Kap. 10.1).

Risiko-klasse	Qualitätssicherungsmaßnahme/ Qualitätssicherungsmaß
A - Hoch	- Code-Inspektion durch Review-Sitzung
	- Checkliste „Code-Inspektion“[4] 100% positiv beantwortet
	- 100% Testfallabdeckung
	- 90% C1-Testabdeckung
B - Mittel	- Code-Inspektion durch schriftliche Stellungnahme
	- Checkliste „Code-Inspektion“ 100% positiv beantwortet
	- 100% Testfallabdeckung
	- 60% C1-Testabdeckung
C - Gering	- 80% Testfallabdeckung

Tabelle 5.3: Risikoklassen für Programme - Maßnahmen

[3] Die innenpolitische Auswirkung für das Projektteam wird hier nicht bewertet ☺

[4] Checkliste „Code-Inspektion“ siehe Kapitel 10.1.2

Die in Abhängigkeit von der Risikoklasse festgelegten Qualitätssicherungsmaßnahmen verringern das Risiko, dass eine Fehlerwirkung erst im produktiven Betrieb des Programms auftritt.

5.4 Werkzeuge für die Risikoanalyse

Werkzeuge können die Priorisierung von Testfällen unterstützen. Zum Beispiel kann mit CaseMaker ([URL: CaseMaker]) jedem Testfall eine Risikoklasse von 1 bis 10 zugeordnet werden.[5] Ein Testszenario erhält dann automatisch die höchste Risikoklasse aller ihm zugeordneten Testfälle. Für die Testplanung lässt sich auf diese Weise eine Übersicht erstellen, in der alle durchzuführenden Testszenarien nach ihrer Priorität geordnet aufgeführt sind und somit die Reihenfolge der Tests festgelegt ist.

Die Einschätzung des Risikos eines Testfalles kann natürlich nur vom Testdesigner und der Fachabteilung vorgenommen werden, nicht von einem Werkzeug.

5.5 Zusammenfassung

- Die Risikoanalyse ist ein wichtiges Instrument der Qualitätssicherung.
- Die ABC-Analyse vereinfacht die Risikoermittlung indem sie mit 3 Risikoklassen statt mit 10 arbeitet.
- Mit der Risikoanalyse werden Testobjekte, Testfälle und Fehler priorisiert, um Risiken von zeit- und kostengetriebenen Entscheidungen beurteilen und Qualitätssicherungsmaßnahmen risikobasiert planen zu können.
- Werkzeuge können die Priorisierung von Testfällen unterstützen und den Planungsprozess vereinfachen.

[5] In diesem Fall wird keine ABC-Analyse, sondern eine 10-stufige Bewertung der Risiken durchgeführt.

6 Checklisten

„Wenn dein einziges Werkzeug ein Hammer ist, neigst du dazu, in jedem Problem einen Nagel zu sehen.“

Abraham Maslow (1908 - 1970)

Ein effektives Hilfsmittel der konstruktiven und analytischen Qualitätssicherung sind wiederverwendbare Checklisten.

6.1 Ziele des Einsatzes von Checklisten

Mit dem Einsatz von standardisierten Checklisten werden mehrere Ziele verfolgt:

- Checklisten machen Dritten in kompakter Form Erfahrungen zugänglich, die von vielen Personen zusammengetragen wurden.
- Einheitliche Checklisten stellen sicher, dass bei Prüfungen keine Punkte vergessen werden.
- Prüfungen, die mit den selben Checklisten durchgeführt werden, sind vergleichbar.
- Durch Checklisten werden Fehler vermieden, wenn sie als konstruktive Qualitätssicherungsmaßnahme frühzeitig bekannt gemacht und bei der Produktentwicklung berücksichtigt werden.

- Der geforderte Erfüllungsgrad einer verbindlich vorgeschriebenen Checkliste stellt eine Anforderung dar, an der die Qualität eines zu prüfenden Ergebnisses gemessen werden kann.[1]

Nehmen wir als Beispiel eine Checkliste zur Code-Inspektion[2] von Programmen, die in einer bestimmten Programmiersprache geschrieben sind.

Eine solche Checkliste wird regelmäßig von den Entwicklern überarbeitet. Sie enthält die aktuellen Programmierstandards sowie die „Tipps & Tricks" der erfahrenen Programmierer. Unerfahrene Entwickler können sich schnell daran orientieren, sie lernen anhand der Checkliste schnell die Standards und Kniffe der betreffenden Programmiersprache kennen.

Weil die Checkliste allen Beteiligten vor Projektbeginn zur Verfügung gestellt wird, werden bestimmte Fehler in der Programmierung von vornherein vermieden.

Mit der Vorgabe, dass für jedes Programm der Risikoklasse A im Rahmen einer Code-Inspektion die Checkliste „Code-Inspektion" zu 100% erfüllt sein muss, stellt sie ein Qualitätsmaß dar.

6.2 Werkzeuge für die Checklistenverwaltung

Checklisten können in jedem Text- oder Tabellenkalkulationsprogramm beschrieben werden.

Reviewmanagement-Werkzeuge

Sollen Checklisten im umfangreichen Maße eingesetzt und die mit ihnen durchgeführten Prüfungen revisionssicher und auswertbar in einer Datenbank abgelegt werden, bieten sich kommerzielle Reviewmanagement-Werkzeuge an, wie zum Beispiel Q-Chess ([URL: QCHESS]).

---

[1] Der Erfüllungsgrad einer Checkliste wird im zweiten Teil des Buches, in dem die Testtypen beschrieben werden, oft als Maß für ein Qualitätssicherungsmerkmal angegeben.

[2] Im Kap. 10.1.2. ist eine ausführliche Checkliste zur Code-Inspektion abgebildet.

6.3 Empfehlungen zum Einsatz von Checklisten

Checklisten zentral verwalten

Um Checklisten effektiv und projektübergreifend einsetzen zu können, sollten sie zentral von einer Qualitätssicherungsinstanz, zum Beispiel dem IT-Qualitätssicherungsverantwortlichen des Unternehmens, verwaltet werden. Diese Instanz muss folgende Aufgaben wahrnehmen:

- Bedarfe für neue Checklisten ermitteln
- Neue Checklisten mit Fachleuten entwerfen
- Checklisten zum Einsatz freigeben
- Bestehende Checklisten regelmäßig überarbeiten
- Veraltete Checklisten aus dem Verkehr ziehen

Akzeptanz der Checklisten herstellen

Damit die Akzeptanz der Checklisten bei allen Beteiligten vorhanden und ihr Nutzen als konstruktives Hilfsmittel möglichst groß ist, sollte der Personenkreis, deren Arbeitsergebnisse mit den Checklisten geprüft werden, an der Erstellung der Checkfragen beteiligt werden. Zudem sollten die Checklisten, bevor sie zum Einsatz freigegeben werden, von Fachleuten geprüft werden.

„Ja/Nein"-Fragen stellen

Die Formulierung der Fragen einer Checkliste, die als Prüfungsgrundlage dient, ist sehr entscheidend. Die Fragen einer Checkliste sollten so formuliert sein, dass sie mit „ja" oder „nein" beantwortet werden können. Allerdings muss in einer Prüfung eine Frage auch mit der Antwort „nicht relevant" beantwortet werden dürfen. Denn das wesentliche Ziel des Einsatzes einer Checkliste ist der Nachweis, dass über alles nachgedacht wurde, und nicht, dass alle Checkpunkte unbedingt umgesetzt wurden.

Checklisten regelmäßig prüfen

Checklisten müssen regelmäßig überarbeitet werden, denn ihre Inhalte können veralten oder Prüfpunkte enthalten, die sich im Einsatz nicht bewährt haben, weil sie zum Beispiel nicht eindeutig sind oder stets mit „nicht relevant" beantwortet werden.

Hinweis

In diesem Sinne erheben die in diesem Buch aufgeführten Checklisten keinen Anspruch auf Vollständigkeit. Sie werden auch nicht hundertprozentig in jedes Projekt oder auf jedes Prüfobjekt passen. Sie sind aber sicherlich eine gute Basis für die Erstellung von eigenen Checklisten, weil sie kurz und knapp wesentliche Sachverhalte erfas-

sen und einen großen, in der Praxis erworbenen Erfahrungsschatz offen legen.

6.4 Zusammenfassung

- Checklisten unterstützen konstruktive und analytische Qualitätssicherungsmaßnahmen.
- Checkfragen sind so zu stellen, dass sie mit „ja“ oder „nein“ beantwortet werden können. Eine Checkfrage ist auch mit einem begründeten „nicht relevant“ positiv beantwortet.
- Checklisten sind nicht restriktiv einzusetzen, sondern dienen dem Nachweis, an alles gedacht zu haben.
- Eine Qualitätssicherungsinstanz sollte Checklisten an zentraler Stelle verwalten und dafür sorgen, dass sie regelmäßig von Experten überarbeitet werden.
- Mit Reviewmanagement-Werkzeugen können Checklisten und Prüfergebnisse zentral gepflegt, verwaltet und ausgewertet werden.

Teil II
Testtypen

7. **Prüfungen von Dokumenten**
8. **Tests zur Funktionalität**
9. **Tests zur Benutzbarkeit**
10. **Tests zur Änderbarkeit und Übertragbarkeit**
11. **Tests zur Effizienz und Zuverlässigkeit**

Der zweite Teil dieses Buches gibt mit der ausführlichen Beschreibung der Testtypen dem Tester Methoden, Beispiele, Tipps und Checklisten an die Hand, um die Tests zu einer Web-Anwendung effizient vorbereiten, durchführen und bewerten zu können.

Jeder Testtyp zeichnet sich dadurch aus, dass er ein bestimmtes Qualitätsmerkmal überprüft. Die Hauptqualitätsmerkmale von Software sind Funktionalität, Benutzbarkeit, Effizienz, Zuverlässigkeit, Änderbarkeit und Übertragbarkeit. Aus diesem Grund wird jedem Qualitätsmerkmal ein Kapitel gewidmet, in dem die Testtypen beschrieben werden, die dieses Qualitätsmerkmal überprüfen.

Eine Ausnahme ist das Kapitel zur Prüfung von Dokumenten, denn die darin beschriebenen Verfahren prüfen kein Qualitätsmerkmal der zu testenden Software, sondern Merkmale der Dokumente, die diese Software beschreiben. Diese Verfahren haben also „nur“ indirekten Einfluss auf die Qualitätsmerkmale der Software. Die Qualitätssicherung von Dokumenten ist aber sehr wichtig, weil durch sie in frühen Projektphasen konzeptionelle Fehler aufgedeckt werden. Daher beginnt der zweite Teil des Buches mit dem Kapitel über Prüfungen von Dokumenten.

7 Prüfungen von Dokumenten

„Wer A sagt, der muss nicht B sagen. Er kann auch erkennen, dass A falsch war.“

Bertold Brecht (1898 – 1956)

Dieses Kapitel behandelt die Qualitätssicherung von Dokumenten. Je nach Inhalt und angesprochener Zielgruppe eines zu prüfenden Dokumentes stehen bei einer Prüfung unterschiedliche Qualitätsmerkmale im Vordergrund. Zum Beispiel hat eine Funktionsbeschreibung die Funktionalität, ein systemtechnisches Konzept die Effizienz und ein Anwenderhandbuch die Benutzbarkeit im Fokus. Das bedeutet, dass der Dokumententest als Testtyp per se keinem bestimmten Software-Qualitätsmerkmal zugeordnet werden kann.

7.1 Dokumententest

Dokumententest

Der Dokumententest überprüft die formale und inhaltliche Vollständigkeit, Widerspruchsfreiheit und Richtigkeit der in einem Dokument beschriebenen Aussagen und Anforderungen.

In einem Software-Entwicklungsprojekt fallen prozessbezogene und produktbezogene Dokumente an, die einer Prüfung unterliegen können.

prozessbezogene Dokumente

Dokumente, die sich auf den Entwicklungsprozess beziehen, sind zum Beispiel Projektmanagementpläne, Statusberichte, Protokolle, Qualitätssicherungspläne und Testpläne, die vom Projekt- und Qualitätsmanagement erstellt werden.

produktbezogene Dokumente

Zu den produktbezogenen Dokumenten gehören fachliche und technische Anforderungen sowie Dokumentationen für den Benutzer.

Fachliche Anforderungsdokumente

Fachliche Anforderungsdokumente sind unter anderem Beschreibungen von Algorithmen und Anwendungsfällen, Datenmodelle, Entwürfe von Websites, Funktionsbeschreibungen, Geschäftsprozessmodelle, Objektmodelle, Online- und Druckformulare sowie Testspezifikationen.

Technische Anforderungsdokumente

Zu den technischen Anforderungsdokumenten gehören Datenbank-Design, Sicherheitskonzept, Netzwerk-Design, Rechnerarchitektur, Konzept zur Ausfallsicherheit und Systemressourcenplanung (Zeitverhalten, Speicherbedarf).

Benutzerdokumentation

Zur Inbetriebnahme einer Anwendung werden Benutzerhandbücher, Installationsanweisungen, Online-Hilfen und Schulungsunterlagen benötigt, die nicht ungeprüft freigegeben werden dürfen.

Diese exemplarische Aufzählung von Dokumenten lässt erahnen, wie aufwendig die Qualitätssicherung aller Dokumente in einem Web-Projekt ist.[1] Aber weil der Dokumententest sicher stellt, dass die fachlichen und technischen Anforderungen des Auftraggebers an eine Software richtig verstanden und realisiert werden und weil vollständig und interpretationsfrei beschriebene Anforderungen den Grundstein für ein erfolgreiches Projekt legen, darf der Dokumententest in keinem IT-Projekt vernachlässigt werden. Dokumententests werden in Form von Reviews durchgeführt.

Review

> Ein Review ist die geplante, systematische, kritische Prüfung von Arbeitsergebnissen.

[1] Was wohl mit ein Grund dafür ist, dass sie oftmals vernachlässigt wird.

Ein Review dient der frühzeitigen Erkennung von Problemen, Fehlern und Inkonsistenzen. Mit einem Review kann auch die Abnahmereife eines Produktes durch Fachleute festgestellt werden. Die Ergebnisse eines Reviews sind zu bewerten und zu protokollieren. Reviews können je nach Prüfziel, Prüfbedingungen und Prüfobjekt unterschiedliche Ausprägungen haben.

Im folgenden wird eine pragmatische Vorgehensweise zur Prüfung von Dokumenten beschrieben, mit der ich positive Erfahrungen in meinen Projekten gemacht habe. Detaillierte Abstufungen von Review-Arten werden hier nicht vorgenommen. Sie sind zum Beispiel in [Spillner_2005] nachzulesen.


Hinweis

Ein bewährtes Vorgehen bei der Qualitätssicherung von Dokumenten besteht darin, jedes fertiggestellte, prüfungsrelevante Dokument zuerst einer formalen Prüfung zu unterziehen. Anschließend wird es in einer Gruppensitzung (Review-Sitzung) durch mehrere Experten geprüft *oder* in einer schriftlichen Stellungnahme durch einen oder mehrere Experten begutachtet.

7.1.1 Formale Prüfung

Ein formal korrektes Dokument erspart bei der anschließenden Prüfung des fachlichen Inhalts wertvolle Zeit, denn Diskussionen über Formalitäten müssen in Review-Sitzungen nicht mehr geführt werden.[2]

dokumentenbezogene und inhaltsbezogene Prüfungen

Eine formale Prüfung kann dokumentenbezogen und/oder inhaltsbezogen sein. Dokumentenbezogen bedeutet, dass das Dokument selbst auf Einhaltung von Formalismen wie Layout und Gliederung geprüft wird. Inhaltsbezogen heißt, dass die Darstellung der Inhalte bestimmten Richtlinien und Standards unterliegt, wie zum Beispiel die Grafik eines Objektmodells oder die beschreibenden Eigenschaften eines Anwendungsfalls.

[2] Die Zeit der Fachabteilung ist immer knapp und teuer! Ich habe Reviews beiwohnen müssen, in denen es erst in der dritten Sitzung um Inhalte ging. Zuvor wurden nur endlos „Form und Farbe“ des Dokumentes diskutiert.

formale Dokumentenprüfung

Die formale Prüfung eines Dokumentes stellt die Einhaltung von Standards, Richtlinien und Methoden für das Dokument selbst und für die darin beschriebenen Inhalte sicher.

Für die formale Prüfung von Dokumenten eignen sich Checklisten, wie die beiden folgenden Beispiele zeigen.

Die dokumentenbezogene Checkliste „Dokument" legt die formalen Eigenschaften fest, die jedes Arbeitsergebnis, dass in Papierform erstellt wird, erfüllen muss.

✓ Checkliste Dokument

- Hat das Dokument eine eindeutige Identifikation?
- Ist ein Deckblatt vorhanden und ist es vollständig ausgefüllt?
- Ist der Autor genannt?
- Sind Vertraulichkeitsstufen und Urheberrechte ausgewiesen?
- Sind Seitenzahlen und Gesamtseiten auf jeder Seite angegeben?
- Entspricht das Layout den vorgegebenen Standards?
- Ist ein Inhaltsverzeichnis vorhanden?
- Stimmt das Inhaltsverzeichnis mit der Gliederung des Dokumentes überein?
- Existiert ein Abbildungsverzeichnis?
- Existiert ein Tabellenverzeichnis?
- Existiert ein Abkürzungsverzeichnis?
- Existiert ein Quellenverzeichnis?
- Sind alle Referenzdokumente aufgelistet?
- Sind Querverweise eindeutig und richtig?
- Ist das Dokument verständlich gegliedert, übersichtlich aufgebaut und leicht verständlich (Grafiken, Tabellen,...)?
- Ist der Inhalt widerspruchsfrei?
- Sind Begriffe eindeutig definiert und durchgängig verwendet?
- Sind alle nicht allgemein bekannten Begriffe und Abkürzungen definiert?
- Ist das Dokument redaktionell vollständig, d.h. fehlen keine Textstellen, Seiten, Abbildungen?
- Ist das Dokument frei von Rechtschreibfehlern und Grammatikfehlern?
- Ist das Dokument aktuell?

- Ist das Dokument stets mit aktuellem Inhalt reproduzierbar?
- Ist der Status (in Arbeit, erledigt, freigegeben,...) angegeben und richtig?
- Gibt es eine Änderungshistorie?
- Sind die Änderungen markiert?
- Hat das Dokument einen Versionsstand?
- Ist das letzte Speicherdatum des Dokumentes angegeben?

Das zweite Beispiel zur formalen Prüfung eines Dokumentes ist eine inhaltsbezogene Checkliste für Datenmodelle. Wenn in einem Projekt die Datenmodellierung als Software-Entwicklungsmethode vorgeschrieben wird, so muss das beschriebene Datenmodell methodischen Anforderungen genügen. Dabei ist es für die formale, inhaltsbezogene Prüfung unerheblich, ob das Datenmodell in einem CASE-Tool spezifiziert wurde oder mit einem Textverarbeitungsprogramm.

✓Checkliste Datenmodell

- Ist die Grafik des Datenmodells vorhanden?
- Hat jede Entität eine Beschreibung
- Sind für jede Entität Beispiele und ggf. auch Gegenbeispiel aufgeführt?
- Wird jede Entität von mindestens einer Funktion angelegt und benutzt?
- Ist jede Entität im Singular benannt?
- Ist begründet, wenn eine Entität nur ein Attribut hat?
- Hat jede Entität einen Primärschlüssel?
- Hat jedes Attribut eine Beschreibung?
- Ist für jedes Attribut angegeben, ob es ein Pflichtfeld ist?
- Hat jedes Attribut Datentyp und Länge?
- Ist jeder Attributname eindeutig?
- Wird jedes Attribut von mindestens einer Funktion angelegt und benutzt?
- Ist jede Beziehung aussagekräftig benannt?
- Hat jede Beziehung eine Beschreibung?
- Sind Integritätsregeln festgelegt?
- Ist die fachliche Beschreibung des Datenmodells korrekt?

- Sind die Namenskonventionen aller Objekte des Datenmodells eingehalten?
- Sind abgeleitete Attribute dokumentiert?
- Ist das Datenmodell in 3. Normalform?
- Sind bei abhängigen Schlüsseln die Beziehungen als Schlüssel definiert?
- Stimmen Schlüsseldefinitionen mit den Beziehungen überein?
- Sind Super-/Subtypen richtig dargestellt?

Wenn die formale Qualität eines Dokumentes sichergestellt ist, kann anschließend der fachliche Inhalt effizienter in einer Review-Sitzung oder einer schriftlichen Stellungnahme geprüft werden.

7.1.2 Review-Sitzung

Review-Sitzung

Eine Review-Sitzung ist die geplante, systematische, kritische Prüfung eines Arbeitsergebnisses durch Fachleute in einer Gruppensitzung.

Review-Regeln

Die Durchführung einer Review-Sitzung unterliegt Regeln:

1. Für die Durchführung des Reviews ist der Qualitätssicherungsmanager des Projektes verantwortlich.
2. Der Teilnehmerkreis eines Reviews besteht in der Regel aus 4 bis 6 Personen (Autor(en), Fachleute als Prüfer, Moderator)
3. Die zu prüfenden Unterlagen werden rechtzeitig versendet, d.h. 7 bis 10 Tage vor dem Review-Termin.
4. Für die Review-Sitzung müssen sich alle Teilnehmer vorbereiten, d.h. die Dokumente durcharbeiten und Fragen und Mängel vorab schriftlich notieren.
5. Da alle Teilnehmer vorbereitet sind, wird das Dokument in der Review-Sitzung nicht mehr vom Autor vorgestellt, sondern es werden in der Vorbereitung erkannte Probleme und auftretende Fragen angesprochen.

6. Fachliche Diskussionen sind vom Moderator zu unterbinden, für Problemlösungen werden Tätigkeiten im Protokoll aufgenommen.

Feststellungen treffen, nicht diskutieren

7. Eine Review-Sitzung dauert 2-4 Stunden.
8. Der Moderator führt das Review-Protokoll.
9. Die festgestellten Mängel werden bewertet.[3]
10. Am Ende einer Sitzung ist eine Entscheidung über das Prüfobjekt zu treffen:

    – Abnahme
    – Abnahme nach Mängelbeseitigung ohne Wiederholung des Reviews
    – Abnahme nach Mängelbeseitigung mit Wiederholung des Reviews
    – Ablehnung

11. Das Protokoll wird an alle Teilnehmer versendet und gilt als akzeptiert, wenn innerhalb eines festgelegten Zeitraumes (zum Beispiel einer Woche) keine Widersprüche eingereicht werden.
12. Im Review-Protokoll vermerkte Mängel sind in einer vom Projektleiter festgelegten Zeit zu beheben und von der Qualitätssicherung oder in einer neuen Review-Sitzung zu prüfen.

↓

Anmerkung zur 8. Regel

In der Literatur wird oft gefordert, dass der Protokollant weder der Moderator, noch ein Prüfer, noch der Autor sein darf. Im Projekt ist dieses Vorgehen aus Personalmangel fast nie realisierbar. Rollen innerhalb einer Review-Sitzung können meines Erachtens auch in Personalunion besetzt werden, ohne dass der Erfolg eines Reviews darunter leidet. Warum sollte der Moderator oder der Autor nicht auch das Protokoll führen, das ohnehin von allen Teilnehmern bestätigt werden muss?

↑

Reviews sind aufwendig

Wie sich aus den obigen Regeln erahnen lässt, sind Review-Sitzungen sehr aufwendig. Dazu ein Beispiel. Wenn 5 Personen ein Review über die Dauer von 4 Stunden durchführen, für das sie sich je 2,5 Stunden vorbereitet haben, wenn der Moderator für

[3] Siehe zum Beispiel Tabelle 5.1 im Kap. 5, Risikoanalyse.

die Einladung, das Protokoll und die Nacharbeiten 8 Stunden aufwendet und wenn der Autor die Mängel in 8 Stunden behebt, dann sind unterm Strich schnell 6 Personentage investiert. Eine eventuell notwendige Wiederholung des Reviews ist dabei nicht einmal berücksichtigt.

Projekt- und Qualitätssicherungsmanagement müssen unter Abwägung der bekannten Risiken entscheiden, für welche Dokumente ressourcenintensive Reviews notwendig sind.

7.1.3 Schriftliche Stellungnahme

Eine budget- und terminfreundliche Alternative zu einer Review-Sitzung ist die schriftliche Stellungnahme.

Schriftliche Stellungnahme

> Eine schriftliche Stellungnahme ist die zu protokollierende, kritische Durchsicht eines Arbeitsergebnisses durch eine oder mehrere Personen mit Fachkompetenz.

Die schriftliche Stellungnahme wird von einer oder mehreren Personen, die voneinander zeitlich und örtlich unabhängig prüfen, durchgeführt und eignet sich in folgenden Situationen:

- Die Prüfergruppe ist so groß, dass sie den Rahmen eines Reviews sprengen würde.
- Die Prüfer können örtlich und/oder zeitlich nicht zusammengeführt werden.
- Die einem Prüfobjekt zugeordnete Risikoklasse sieht „nur“ eine Vieraugenüberprüfung vor, d.h. es ist hinreichend, dass nur eine Person das Dokument prüft.

Wenn mehrere Prüfer an einer Stellungnahme beteiligt sind, müssen die Aktivitäten von einem Qualitätssicherungsverantwortlichen koordiniert werden:

1. Die Prüfer erhalten die zu prüfenden Unterlagen mit Nennung des Prüfzieles und des Rückgabedatums, das 1 bis 2 Wochen nach dem Versandtag liegt.
2. Jeder Prüfer verfasst ein Prüfprotokoll, das er zum Rückgabedatum abliefert.

3. Der Koordinator fasst alle Prüfprotokolle zusammen und entscheidet anhand der Ergebnisse über den Stand des Prüfobjekts (Abnahme, Abnahme nach Mängelbeseitigung, Wiederholung des Prüfung, Ablehnung).
4. Er erstellt ein Gesamtprotokoll mit allen festgestellten Mängeln, Kommentaren und Antworten sowie die getroffene Entscheidung zur Abnahme des Prüfobjektes.
5. Das Protokoll wird an alle Teilnehmer versendet und gilt als akzeptiert, wenn innerhalb einer Woche keine Widersprüche eingereicht werden.

Wenn es um fachlich wegweisende oder projektrelevante Entscheidungen geht, kann eine schriftliche Stellungnahme ein Review nicht ersetzen, denn schriftliche Stellungnahmen bergen die Gefahr, dass sie von den Prüfern nebenher, d.h. nicht gründlich durchgeführt werden. Bei einer Review-Sitzung dagegen wird schnell offensichtlich, wer seine Hausaufgaben gemacht hat.

Ein „gesunder“ Mix aus Reviews und schriftlichen Stellungnahmen zur Prüfung von Dokumenten ist daher jedem Projekt zu empfehlen.

7.2 Spezielle Dokumententests

Es gibt produktbezogene Dokumente, die auf Grund ihres speziellen Inhalts per se einem Qualitätsmerkmal zugeordnet werden können. Ihre Prüfungen finden in speziellen Dokumententests statt, die eigene Testtypen darstellen. Sie sind in den entsprechenden Kapiteln beschrieben und der Vollständigkeit halber hier aufgeführt.

Content-Test

Eine Website kann als eigenständiges Dokument angesehen werden, dessen Inhalt den Standards und rechtlichen Vorgaben des WWW genügen muss. Die Prüfung des Inhaltes einer Website wird im Content-Test mit den Methoden des Dokumententests durchgeführt. Der Content-Test überprüft als Schwerpunkt das Qualitätsmerkmal Benutzbarkeit eines Web-Auftritts und wird im Kapitel 9.1 beschrieben.

Code-Analysen

Code-Walkthroughs und Code-Inspektionen sind spezielle Ausprägungen eines Reviews. Weil sie programmiertechnisches Wissen von den Prüfern verlangen und gezielt die Qualitätsmerkmale Änderbarkeit und Übertragbarkeit prüfen, werden die-

se beiden Prüfungsarten im Kapitel 10.1, Code-Analysen, beschrieben.

7.3 Werkzeuge für den Dokumententest

CASE-Tools

In IT-Projekten sollte die Projektdokumentation mit CASE-Tools (Computer Aided Software Engineering) erfolgen, denn aus Sicht der Qualitätssicherung bieten sie folgende Vorteile:

- Der Einsatz eines CASE-Tools zwingt das Projektteam, nach vorgegebenen Methoden und Vorlagen zu arbeiten.
- Der Aufbau und die Darstellung der Arbeitsergebnisse ist von den einzelnen Autoren unabhängig.
- Die Dokumente werden zentral verwaltet.
- Die Versionsführung der Dokumente ist gewährleistet.
- Formale Prüfungen zur Vollständigkeit und Widerspruchsfreiheit der Dokumente sowie zur Einhaltung von Modellierungsregeln werden automatisch ausgeführt.

Rational Rose

Unter [URL: CASE-Tools] ist eine umfangreiche Liste von CASE-Tools aufgeführt. Eine bekanntes Werkzeug zur Darstellung von UML-Diagrammen ist zum Beispiel Rational Rose von IBM ([URL: Rational]).

7.4 Qualitätsanforderungen zum Dokumententest

Qualitätsmerkmale, die für jedes abnahmerelevante Dokument nachgewiesen werden müssen, sind formale Korrektheit und Vollständigkeit sowie inhaltliche Fehlerfreiheit, Widerspruchsfreiheit und Vollständigkeit. Externe Dokumente müssen zudem benutzerfreundlich sein.

Checklisten

Die formale, dokumentenbezogene Korrektheit und Vollständigkeit kann anhand der Checkliste „Dokument“ gemessen werden. 100% Abdeckung der Checkliste bedeutet, dass im Rahmen der Prüfung alle Fragen der Checkliste mit „ja“ oder „nicht relevant“ beantwortet wurden.

Für Anforderungsdokumente, die nach speziellen Methoden erstellt werden müssen, sollten eigene Checklisten zur inhaltsbezogenen Prüfung festgelegt werden. Dann kann zum Beispiel gefordert werden, dass für ein Datenmodell die Checkliste „Datenmodell“ zu 100% erfüllt sein muss.

Prüfprotokolle

Fachliche Fehlerfreiheit, Widerspruchsfreiheit und Vollständigkeit eines Dokuments werden anhand von Prüfprotokollen nachgewiesen, die bei der Durchführung von Reviews und schriftlichen Stellungnahmen erstellt werden. Die Qualitätsanforderung ist, dass jedes Dokument nachweislich in einem Prüfprotokoll vom Auftraggeber abgenommen wird.

Benutzerfreundlichkeit

Die Benutzerfreundlichkeit von Dokumenten mit Außenwirkung (Benutzerhandbücher, Installationsanweisungen,...) kann durch Testpersonen festgestellt werden. Als Qualitätsanforderung kann zum Beispiel festgelegt werden, dass ungeschulte Nutzer, die ein Dokument lesen, im Durchschnitt höchstens 4 Verständnisfragen pro 50 Seiten Text stellen dürfen.

7.5 Empfehlungen zum Dokumententest

Dokumententests in frühen Phasen

Der Dokumententest hat seine Hoch-Zeit in den frühen Phasen eines Projektes, wenn die meisten konzeptionellen Arbeitsergebnisse entstehen. Da Reviews und schriftliche Stellungnahmen sehr ressourcenaufwendig sind, müssen sie rechtzeitig zum Projektbeginn geplant und budgetiert werden. Eine Balance zwischen Termin, Budget und Qualität sollte dabei erreicht werden.

Checklisten für formale Prüfungen

Zu Beginn eines Projektes sollte für jeden zu erstellenden Dokumententyp, der auf die Einhaltung von Regeln und Standards geprüft werden muss, eine Checkliste für die formale Prüfung bereitgestellt werden. Einerseits als konstruktive Qualitätssicherungsmaßnahme, andererseits, um fachliche Prüfungen effizient durchführen zu können.

4-Augen-Prinzip

Ein ungeschriebenes Gesetz sollte in jedem Projekt sein, dass jedes Dokument, das für Dritte relevant ist, dem Vier-Augen-Prinzip unterliegt.

CASE-Tool einsetzen

Für umfangreichere Projekte empfiehlt sich der Einsatz eines CASE-Tools. Er ist unter der Vorrausetzung, dass die Mitarbeiter die Methoden und das Tool beherrschen, als konstruktive Qualitätssicherungsmaßnahem sehr wirkungsvoll, weil so formal korrekte, einheitliche und stets aktuell abrufbare Spezifikationsdokumente erzwungen werden.

7.6 Zusammenfassung

- Wenn Dokumente, an denen sich das Projektgeschehen und die Anwendungsentwicklung orientieren, unvollständig, nicht eindeutig oder gar fehlerhaft sind, wird es sehr teuer, die nicht rechtzeitig erkannten Mängel nachträglich zu beheben.
- Der Dokumententest ist keine Domäne der Qualitätssicherung von Web-Applikationen, sondern muss in jedem IT-Projekt eingeplant werden.
- Dokumententests können für projektbezogene und für produktbezogene Dokumente durchgeführt werden.
- Dokumententests werden in Form von Reviews oder schriftlichen Stellungnahmen durchgeführt, denen eine formale Prüfung voraus geht.
- Für die formale Prüfung von Dokumenten eignen sich Checklisten, die vor der Erstellung der Dokumente den Autoren als konstruktive Qualitätsmaßnahme bereitgestellt werden.
- Reviews und schriftliche Stellungnahmen unterliegen festen Regeln.
- Die Prüfergebnisse eines Dokumententests werden in einem Prüfprotokoll festgehalten.
- Der Einsatz eines CASE-Tools ist eine wichtige konstruktive Qualitätssicherungsmaßnahme.
- CASE-Tools können formale Prüfungen übernehmen.
- Content-Tests, Code-Walkthroughs und Code-Inspektionen werden mit den statischen Methoden des Dokumententests durchgeführt.
- Das Vier-Augen-Prinzip gilt für alle Dokumente, die an Dritte gegeben werden.

8 Tests zur Funktionalität

„Irren ist menschlich. Aber wenn man richtig Mist bauen will, braucht man einen Computer.“

Dan Rather, CBS – Fernsehreporter (*1931)

In diesem Kapitel werden die Testtypen beschrieben, die das Qualitätsmerkmal Funktionalität überprüfen. Sie stellen sicher, dass die von einer Web-Anwendung geforderten Funktionen angemessen, vollständig und korrekt realisiert worden sind, dass die einzelnen Komponenten zu einem fehlerfrei funktionierendem Gesamtsystem integriert werden können und dass die Sicherheit der Daten gewährleistet ist.

Im Folgenden werden zuerst die klassischen funktionalen Tests beschrieben, die in jedem Anwendungsentwicklungsprojekt durchgeführt werden müssen. Das sind die Testtypen Klassentest, Komponententest, Integrationstest und funktionaler Systemtest.[1]

Anschließend werden die Testtypen beschrieben, die beim Einsatz der entsprechenden Web-Technologie notwendig sind. Dabei handelt es sich um den Link-Test, Cookie-Test und Plugin-Test. Dazu kommt der Sicherheitstest mit seinen web-spezifischen Besonderheiten.

[1] Die Testtypen sind nicht mit den zum Teil gleichnamigen Teststufen zu verwechseln (s. Kap. 14).

8.1 Klassentest

Die erste Gelegenheit, die Funktionen eines Programms zu testen, hat der Entwickler. Er führt, bevor er seine Software an das Testteam zur Qualitätssicherung übergibt, einen Entwicklertest durch.

Entwicklertest

Der Entwicklertest ist der informelle Test einer Software-Komponente, der parallel zur Entwicklung vom Programmierer in seiner Entwicklungsumgebung durchgeführt wird. In diesem Buch wird der Begriff Entwicklertest für die entsprechende Teststufe benutzt (s. Kap. 14.1).[2]

Weil in der objektorientierten Welt des WWW die kleinste, sinnvoll vom Entwickler zu testende Software-Einheit die Klasse ist, reden wir hier vom Klassentest.

Klassentest

> Der Klassentest einer objektorientierten Anwendung umfasst den Klasseneinzeltest und den Klassenintegrationstest. Im Klasseneinzeltest wird die Funktionalität einer Klasse getestet. Im Klassenintegrationstest wird die Funktionalität einer Gruppe miteinander verbundener Klassen getestet.

8.1.1 Klasseneinzel- und Klassenintegrationstest

Klassen-einzeltest

Beim Klasseneinzeltest wird für jede Klasse eine eigene Testklasse kodiert, die möglichst alle Methoden aufruft und überprüft. Weil die Methoden einer Klasse gemeinsam Klassenattribute verwenden und voneinander abhängen können, ist es nicht sinnvoll, Methoden einzeln zu testen. Jeder Sonder- oder Fehlerfall (jede eingehende und jede ausgehende Exception) einer Methode oder eines Objekts wird als eigener Testfall implementiert.

Abb. 8.1: Klassen-einzeltest

Für den Entwickler bedeutet das, dass er sein Programm sukzessive anhand der Ein- und Ausgabeanforderungen entwickelt und testet.

---

[2] ☺ Testertest: Der Begriff „Entwicklertest“ bezeichnet den Personenkreis, der den betreffenden Test ausführt. Dementsprechend müssten eigentlich alle anderen Tests „Testertests“ genannt werden.

Er programmiert einige Funktionalitäten, beschreibt die dazugehörenden Testfälle und führt die Tests durch. Dabei prüft er die internen Programmstrukturen. Sind die Tests erfolgreich, werden Testfälle und Code so lange in kleinen Schritten weiterentwickelt, bis der komplette Code programmiert ist. Bei jedem neuen Schritt werden die bestehenden Testfälle wieder mitgetestet.

Klassenintegrationstest

Nachdem eine Klasse programmiert und getestet ist, wird sie mit fachlich und/oder technisch zusammengehörenden Klassen integriert. Im Klassenintegrationstest werden Vererbung und Nachrichtenaustausch zwischen dienstanbietenden Objekten auf dem Server und dienstnutzenden Objekten auf dem Client getestet.

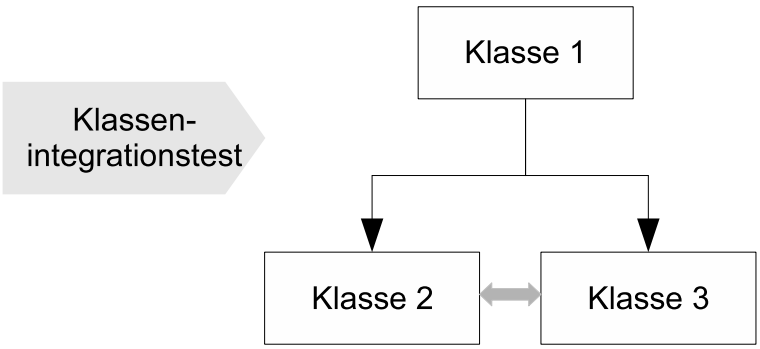


Abb. 8.2: Klassenintegrationstest

Test Driven Development

Spezifiziert der Entwickler *vor* jedem Codierungsschritt den zugehörenden Testfall und entwickelt anhand dessen den Code, so wird von testgetriebener Programmierung oder Test Driven Development gesprochen.[3]

8.1.2 Abgrenzung und Empfehlung zum Klassentest

Auch wenn der Klassentest vom Entwickler durchgeführt wird, sollte er wiederholbar sein und nachweislich dokumentiert werden. Die Grenzen zwischen dem vom Entwickler durchzuführenden Klassentest und dem anschließend vom Testteam durchzuführenden Komponententest (Kap. 8.2) verschwimmen bei der oben beschriebenen Vorgehensweise und lassen diesbezüglich einige Fragen aufkommen:

- Muss ein Entwickler Testnachweise erbringen? Dann wäre er ja Tester und nicht Entwickler!

[3] Literatur zur testgetriebenen Entwicklung: [Westphal_2005]

- Kann ein Tester den Klassentest, wie oben beschrieben, durchführen? Dann müsste er ja die Programmiersprache beherrschen und die notwendigen Testklassen programmieren.
- Welche Klassen werden wann zusammengefasst und als Komponenten an das Testteam übergeben? Wo also hört der Klassenintegrationstest auf und wo fängt der Komponententest an?

Antworten auf diese Fragen müssen im Rahmen der Testplanung festgeschrieben werden. Eine mögliche Vorgehensweise ist folgende:

- Das Testteam definiert parallel zur Spezifikationserstellung mit den Blackbox-Verfahren die Testfälle für den Komponententest und stellt sie dem Entwicklungsteam zur Verfügung.
- Der Entwickler führt parallel zur Programmierung Klassentests durch und verwendet dazu die vom Testteam bereitgestellten Testfälle.
- Die Klassentests werden mit geeigneten Testframeworks (s. Abschnitt 8.1.3) durchgeführt und dokumentiert.
- Die Integration funktional und technisch abhängiger Klassen wird vom Entwickler im Klassenintegrationstest geprüft. Das Endprodukt einer Klassenintegration ist eine im Testplan definierte Komponente.
- Eine Komponente wird nach dem Klassentest an das Testteam in den Komponententest übergeben. Das Testteam entscheidet, ob und welche Tests, die vom Entwickler durchgeführt worden sind, in der Testumgebung wiederholt werden müssen oder ob die Testnachweise des Entwicklers hinreichend sind.

Dieses Vorgehen bietet einige Vorteile. Das Entwicklerteam muss sich keine eigenen Testfälle ausdenken, Mängel in den Anforderungsspezifikationen und Testfällen werden frühzeitig entdeckt und ein Teil des Testaufwands wird in eine frühe Projektphase verlagert.

8.1.3 Werkzeuge für den Klassentest

Testframeworks

Programmierung und Test sind bei dem oben beschriebenen Vorgehen eng miteinander verbunden, denn mit jeder Erweiterung des Codes müssen alle bisher durchgeführten Tests wiederholt werden. Das ist ohne entsprechende Werkzeuge nicht durchführbar. Daher wer-

den Testframeworks für den Klassentest eingesetzt. Mit ihnen werden unter kontrollierten Bedingungen die Tests automatisch wiederholt. Programm- und Testdokumentation werden auf diese Weise stets aktuell fortgeschrieben. Zudem werden die Testergebnisse mit einem Testframework protokolliert und ausgewertet.

xUNITs

Testframeworks gibt es für fast jede Programmiersprache im Web-Umfeld. Eine der bekanntesten Frameworks ist das Java-Framework JUnit ([URL: junit01]). Eine deutsche Dokumentation mit Beispielen hat Franz Westphal in [URL: junit02] veröffentlicht.

Für die Open Source Skriptsprache PHP 5 wurde PHPUnit 2 (ebenfalls Open Source) entwickelt, das die Funktionalität von JUnit umfasst. Der Vorgänger PHPUnit für PHP 4 bildet dagegen nicht die komplette Funktionalität von JUnit ab ([Bergmann_2005], [URL: PHPUnit]). Weitere frei verfügbare Testframeworks gibt es für Visual Basic (vbUnit), C++ (CppUnit), .NET[GL] (NUnit) und Smalltalk (SUnit). Eine Aufstellung von Testframeworks ist in [URL: TestToolsFAQ] unter der Rubrik „Unit Test Tools“ aufgeführt.

Informationen

Eine umfassende Beschreibung zum Einsatz von Frameworks gibt das Buch von Paul Hamill [Hamill_2004].

8.1.4 Qualitätsanforderungen zum Klassentest

Methodenabdeckung

Wenn die Tests der Klassen parallel zur Programmierung und werkzeugunterstützt im Testframework durchgeführt werden, sind alle Testergebnisse nachweislich und ohne großen Aufwand für den Entwickler dokumentiert. Und weil zu jeder programmierten Methode eine zugehörende Testmethode kodiert ist, ist automatisch die Testmetrik der Methodenabdeckung zu 100% erfüllt, die für den Klassentest einer Web-Anwendung eingefordert werden sollte.

8.2 Komponententest

Hinweis

Im Folgenden wird der Testtyp Komponententest beschrieben, der die geforderte Funktionalität einer Software-Komponente sicherstellt und nicht mit der Teststufe Komponententest (Kap. 14.2) zu verwechseln ist.

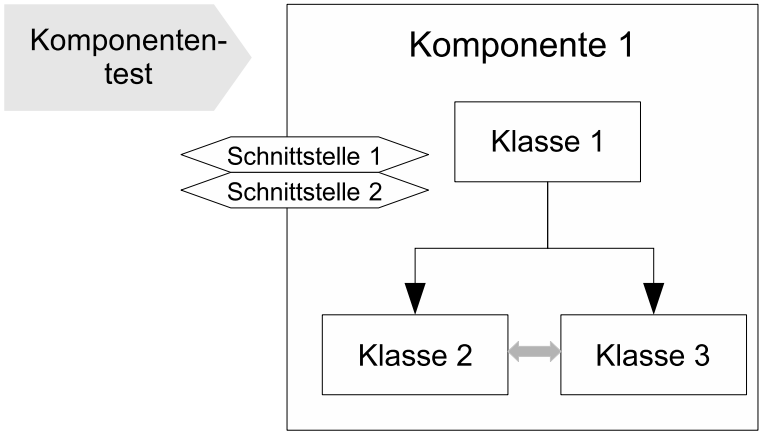
Komponente

Eine Komponente ist die kleinste Software-Einheit, für die eine Spezifikation verfügbar ist und die isoliert getestet werden kann.[4]

Komponententest

> Der Komponententest testet eine einzelne Software-Komponente mit dem Ziel, Fehler aufzudecken und die korrekte und vollständige Umsetzung der funktionalen Anforderungen an diese Software-Komponente nachzuweisen.

Abb. 8.3: Komponententest



Modultest, Unit-Test

In der nicht objektorientierten Welt sind Komponenten als Module und in der Objektorientierung als Units bekannt. Daher wird der Komponententest auch Modultest oder Unit-Test genannt.

↓

Beispiel einer Komponente

Betrachten wir als Beispiel den Finanzierungsrechner aus Kapitel 4.1. Der Finanzierungsrechner ist eine separat zu testende Software-Komponente, weil für ihn eine Spezifikation vorliegt, er eine funktionale Einheit bildet und nicht sinnvoll unterteilt werden kann.

Die Programmlogik des Finanzierungsrechners wird zum Teil auf dem Client und zum Teil auf dem Server durchgeführt (Abb. 8.4). Die Plausibilitätsprüfungen der Benutzereingaben werden auf dem Client vorgenommen und die Berechnung der Finanzierung auf dem Server. Für den Komponententest des Finanzierungsrechners wird die Schnittstelle zum Autokonfigurator durch ein Treiberprogramm ersetzt. Die Schnittstelle zur Verfügbarkeitsanfrage an die Datenbank übernimmt ein Platzhalter.

[4] Die kleinste zu testende Einheit ist für den Tester die Komponente und für den Entwickler die Klasse (s. Kap. 8.1, Klassentest).

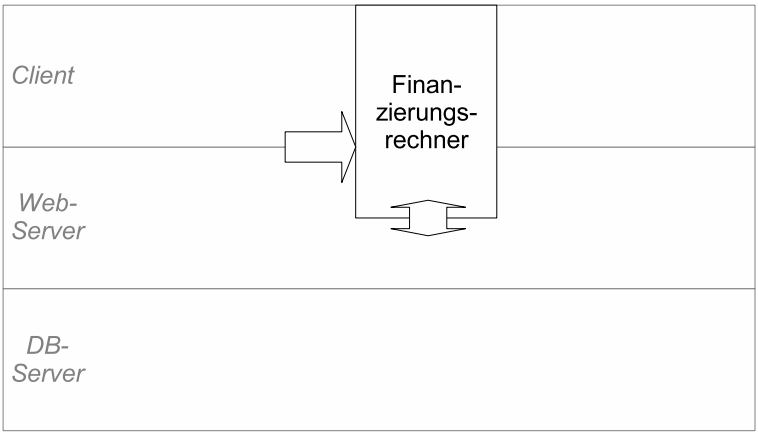


Abb. 8.4: Komponente Finanzierungsrechner

↑

8.2.1 Schritte des Komponententests

Die Funktionalität, die in einer Komponente realisiert ist, ist aus fachlicher Sicht in sich abgeschlossen und wird im Komponententest in mehreren Schritten getestet:

1. Blackbox-Tests: Im ersten Schritt werden die mit Blackbox-Verfahren (Kap. 4.1) entworfenen Testfälle in einem Blackbox-Test ausgeführt.
2. Whitebox-Tests: Nachdem die Testfälle des Blackbox-Tests erfolgreich durchgeführt worden sind, d.h. durch sie keine Fehlerwirkungen mehr aufgedeckt werden, werden Whitebox-Tests für die Software-Komponente durchgeführt. Dieser zweite Schritt ist nur notwendig, wenn in den Qualitätsanforderungen der Nachweis von Testabdeckungsgraden gefordert ist (Kap. 4.2).
3. Fehlererwartungstests: Bewährt hat sich als zusätzlicher Test der Fehlererwartungstest, der eine Software-Komponente auf Basis intuitiv erstellter Testfälle (Kap. 4.3) testet. Dieses „unmethodische“ Vorgehen wird auch Fehlererwartungstest oder Error Guessing Test genannt.

8.2.2 Werkzeuge für den Komponententest

Treiber und Platzhalter

Um die Schnittstellen einer Komponente mit Testdaten bedienen zu können, müssen ggf. Testtreiber und Platzhalterprogramme entwickelt werden. Testtreiber rufen die zu testende Komponente auf, Platzhalter werden von der zu testenden Komponente aufgerufen.

Capture Replay Tools

Für die Ausführung der Tests können Capture Replay Tools eingesetzt werden. Damit wird die Durchführung von Testszenarien aufgezeichnet und wieder abgespielt. Obwohl Capture Replay Tools ursprünglich für die Automatisierung von Regressionstest entwickelt worden sind, sind sie für den Komponententest sehr interessant. Mit ihnen können nicht nur Tests aufgezeichnet und wieder abgespielt, sondern ganze Testläufe mittels Skriptsprachen programmiert werden. (S. Kap. 12.4, Werkzeuge für die Testwiederholung.)

Code Coverage Tools

Testabdeckungsgrade werden, sofern gefordert, im Rahmen des Komponententests nachgewiesen. Dazu sind Code Coverage Tools notwendig, deren Arbeitsweise im Kap. 4.2.3. beschrieben ist.

8.2.3 Qualitätsanforderungen zum Komponententest

Die im Rahmen des Komponententests auszuführenden Testfälle werden mit den Blackbox- und Whitebox-Verfahren entworfen. Daher leiten sich die Qualitätsanforderungen zum Komponententest wie Testfallabdeckungsgrad und Testabdeckungsgrad von den Box-Verfahren ab. Sie sind ausführlich in den Abschnitten 4.1.6 und 4.2.4 beschrieben.

8.3 Integrationstest

Um eine Web-Applikation betreiben zu können, müssen die Systemkomponenten zu einem Gesamtsystem integriert werden. Dazu werden im Integrationstest die einzelnen Komponenten sukzessive zusammengeführt, bis in der letzten Stufe der Integration das gesamte System zur Verfügung steht und im Systemtest komplette Geschäftsvorfälle durchlaufen und getestet werden können.

Integrationstest

Der Integrationstest prüft das korrekte funktionale und technische Zusammenspiel von Hard- und Software-Komponenten, mit dem Ziel, Fehlerwirkungen in den Schnittstellen und der Kommunikation zwischen den Komponenten aufzudecken.

Als Voraussetzung für den Integrationstest müssen die zu integrierenden Komponenten zuvor im Komponententest geprüft worden sein und fehlerfrei funktionieren.

Für jede Abhängigkeit zwischen zwei Komponenten werden für den Integrationstest Testfälle definiert, die nachweisen, dass die Komponenten über ihre Schnittstellen korrekt kommunizieren, d.h. Daten austauschen und gemäß den Anforderungen richtig funktionieren. Zusätzliches Augenmerk liegt beim Integrationstest auf der Fehlerbehandlung von Ausnahmesituationen.

8.3.1 Strategien der Integration

Es gibt unterschiedliche Strategien, wie Komponenten zu einem Gesamtsystem integriert werden können.

Backbone

Bei der Backbone-Integration werden fertiggestellte Komponenten in ein zuvor erstelltes Programmskelett von Treibern und Platzhaltern in der zeitlichen Reihenfolge ihrer Fertigstellung eingeklinkt.

Ad-hoc

Werden die Komponenten in der zeitlichen Reihenfolge ihrer Fertigstellung zusammengefügt, ohne ein Programmskelett zur Verfügung zu haben, wird von Ad-hoc-Integration gesprochen. Die Tests können bei diesem Vorgehen zeitlich nicht optimiert durchgeführt werden, dafür entfällt der Aufwand für die Erstellung eines kompletten Programmskeletts. Allerdings müssen bei Bedarf auch Treiber und Platzhalter programmiert werden.

Top-down, Bottom-up

Integrationsverfahren wie die strenge Top-down- oder Bottom-up-Zusammenführung von Modulen sind in der flachen Programmhierarchie von Web-Anwendungen ungeeignet.

Big Bang

Ein „Big Bang“, der alle Komponenten gleichzeitig zusammenführt und testet, verbietet sich von selbst. Denn wie soll mit vertretbarem Aufwand der verursachende Fehlerzustand einer auftretenden Fehlerwirkung in der Komplexität einer Web-Applikation lokalisiert werden?

Anwendungsfallorientiert

Bewährt hat sich die Integration der einzelnen Komponenten anhand von Anwendungsfällen (Use Cases) und Geschäftsvorfällen. Auch für dieses Vorgehen müssen Treiber und Platzhalter bereitgestellt werden.

In der Praxis lassen sich die beschriebenen Integrationsstrategien nicht scharf trennen, sondern in der Regel werden Mischformen umgesetzt.

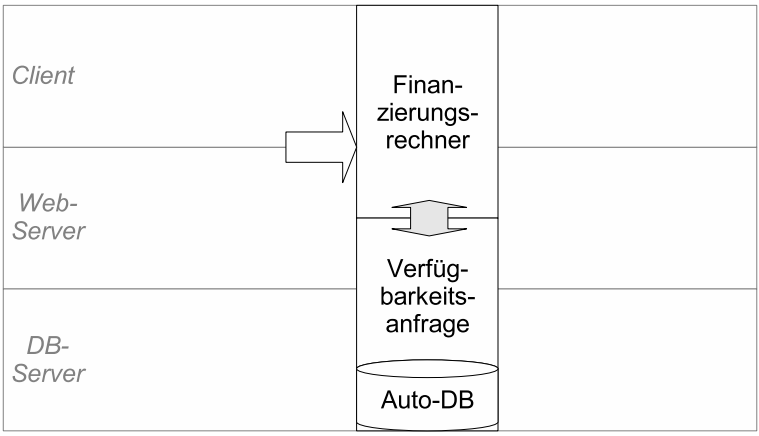
↓ *Beispiel zum Integrationstest*

Nehmen wir unser Beispiel des Finanzierungsrechners wieder auf.

Beispiel - 1. Integrationsstufe:

Die Komponente <Finanzierungsrechner> (Abb. 8.4) ist vom Testteam im Komponententest nach allen Regeln der Testkunst geprüft worden und wird in den Integrationstest gegeben, damit sie mit der Komponente <Verfügbarkeitsanfrage> zusammengeführt werden kann (Abb. 8.5).

Abb. 8.5: Integration Verfügbarkeitsanfrage



Wenn der Kunde mit dem Finanzierungsrechner eine Finanzierungsmöglichkeit berechnet hat und ein verbindliches Angebot haben möchte, muss eine Anfrage an die produktive Datenbank gestellt werden. Diese stellt fest, ob und ab wann das konfigurierte Fahrzeug zur Verfügung steht. Die Verfügbarkeitsanfrage enthält selbst keine Verarbeitungslogik.

In dieser Integrationsphase muss die Anbindung der Datenbank mit den dazu notwendigen Kommunikationskomponenten geprüft werden. In unserem Beispiel ist die Kommunikation der Systemkomponenten mit SOAP[5] realisiert. Die beteiligten Komponenten und die Kommunikationswege sind in Abb. 8.6 skizziert.

---

[5] SOAP ist die Abkürzung für Simple Object Access Protocol. SOAP definiert als offizieller W3C-Standard einen Nachrichtenaustausch zwischen zwei Partnern im Internet.

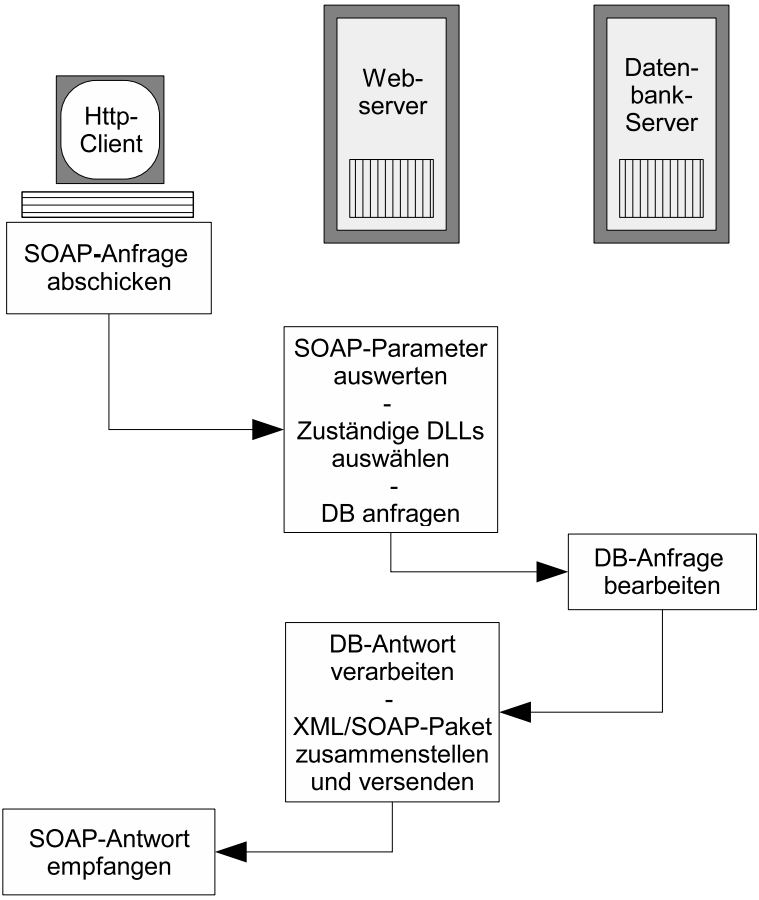


Abb. 8.6: SOAP-Anfrage

Der Benutzer stellt seine Anfrage über seinen Internetzugang (HTTP-Verbindung über Port 80), die über einen Web-Server an eine Datenbank weitergeleitet und von dort auf demselben Weg zurückgegeben wird.

Beim Integrationstest wird überprüft, ob die benötigten Systemkomponenten (Client, Web-Server, Datenbank-Server, Kommunikations-DLLs) zur Anfrage der Verfügbarkeit fehlerfrei zusammenarbeiten. Dazu müssen fachliche und technische Testfälle entworfen werden.

fachliche Testfälle

Aus fachlicher Sicht gibt es zwei Testfälle, die nach der Äquivalenzklassenanalyse (Kap. 4.1.1) gebildet werden:

1. Das konfigurierte Fahrzeug ist laut Datenbankanfrage verfügbar.
2. Das konfigurierte Fahrzeug ist nicht verfügbar.

technische Testfälle

Aus technischer Sicht müssen mehrere Situationen getestet werden. Tritt irgendwo auf dem Weg der Nachrichtenübermittlung ein Fehler auf, muss einerseits der Benutzer eine hilfreiche Meldung im Browser erhalten und anderseits der Systemadministrator benachrichtigt werden. Für jede Fehlersituation, die während des Betriebes auftreten kann, wird ein Testfall benötigt. Exemplarisch ist hier der Testfall beschrieben, der die Reaktion auf eine nicht bereitstehende DLL überprüft:

Testfall:
Verteiler-DLL auf Web-Server nicht verfügbar

Beschreibung:
Der Web-Server wertet die Anfrage des Web-Clients aus und ruft als Ergebnis die für den Datenbank-Zugriff zuständige DLL auf. Steht diese DLL dem Web-Server nicht zur Verfügung, muss diese Situation korrekt abgefangen und eine Meldung an den Benutzer gegeben werden.

Erwartete Ergebnisse:

1. Internal Server Error, HTTP-Fehler 500, Rückgabe einer SOAP-Message im Response Body mit der SOAP-Fault: „Leider liegt zur Zeit ein technisches Problem vor, bitte versuchen Sie es später noch ein Mal.“
2. Eintrag in eine Protokolldatei für den Systemadministrator: HTTP-Fehler 500 - Internal Server Error, inkl. SOAP-Message wie oben.

Weitere technische Fehlersituationen, die in Testfällen zu behandeln sind, lassen sich für unser Szenario schnell finden:

- Der Web-Server steht nicht bereit.
- Der Datenbank-Server steht nicht bereit.
- Die Auto-Datenbank steht nicht zur Verfügung.

Für unseren Finanzierungsrechner haben wir nun nach der Integration der Datenbankanfrage die in Abb. 8.5 dargestellte Integrationsstufe erreicht.

Beispiel - 2. Integrationsstufe:
Bisher wird für den Test ein Treibermodul eingesetzt, dass dem Finanzierungsrechner den Kaufpreis des Fahrzeugs übergibt. Im nächsten Schritt soll der Finanzierungsrechner mit dem Subsystem <Autokonfigurator> verbunden werden. Dieses ist schon als einzeln aufzurufende Funktion produktiv im Einsatz. Die nächste Testphase beweist, dass der Autokonfigurator den Finanzierungsrechner fehlerfrei aufruft und den Listenpreis des konfigurierten Fahrzeugs korrekt als Kaufpreis übergibt. Das System besteht nun aus den in Abb. 8.7 gezeigten Komponenten.

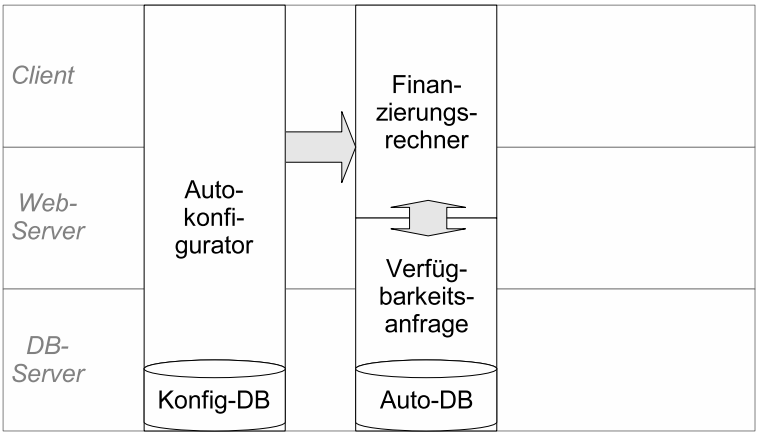


Abb. 8.7: Integration Autokonfigurator

Beispiel - 3. Integrationsstufe = Gesamtsystem:
Gehen wir davon aus, dass auch die Vertragsabwicklung in mehreren Integrationsphasen zu einem Subsystem herangereift ist, so kann mit der letzten Integrationsstufe das gesamte System im Zusammenspiel getestet werden.[6]

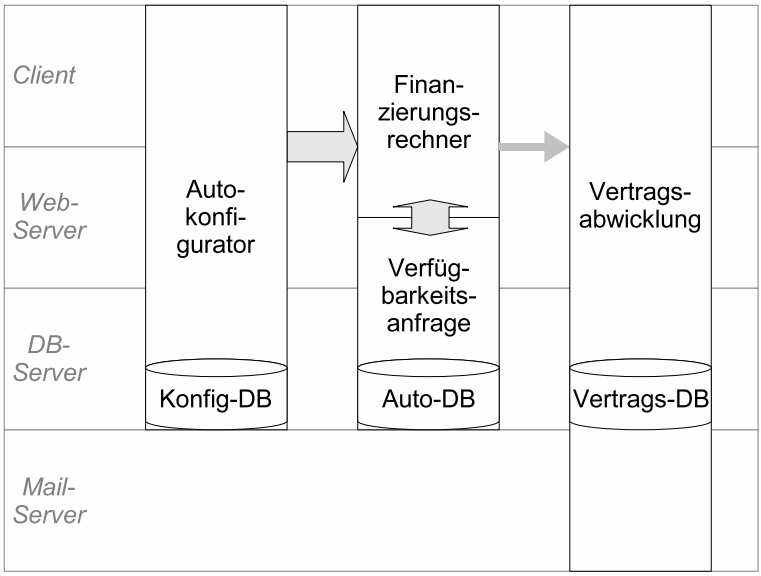
Der Finanzierungsrechner tauscht mit der Vertragsabwicklung keine Daten auf direktem Wege aus, sondern die Vertragsabwicklung liest einmal täglich die benötigten Informationen aus dem Datenhaltungssystem des Finanzierungsrechners aus[7]. Daher sind für die bisher beschriebenen Integrationstests keine Treiber oder Platzhalter für diese indirekte und zeitversetzte Kommunikation notwendig gewesen.

Wie die Abb. 8.8 zeigt, hat das Subsystem <Vertragsabwicklung> das Gesamtsystem um einen Mail-Server erweitert, weil damit ver-

[6] Der Übersichtlichkeit halber sind einige Komponenten wie Login-Prozedur, Firewall und Load-Balancer außen vor gelassen.
[7] Die Schnittstelle ist in Abb. 8.8 als schmaler Pfeil dargestellt.

bindliche Finanzierungsangebote per E-Mail an den Kunden versendet werden können.

Abb. 8.8: Gesamtsystem



Das Anwendungssystem steht nun mit getesteten Schnittstellen für den anschließenden Systemtest zu Verfügung.

↑

Dieses ausführliche Beispiel macht deutlich, wie sehr die einzelnen Schritte des Integrationstests von den zu integrierenden Komponenten und den verwendeten Web-Technologien abhängen. Für jede Integrationsstufe müssen fachliche und technische Testfälle entworfen werden.

8.3.2 Integration externer Komponenten

externe Komponenten

Der Integrationstest von Web-Anwendungen beschränkt sich nicht nur auf selbst entwickelte Komponenten. Auch in das System eingebundene Standardsoftware und für den Einsatz der Web-Applikation benötigte Komponenten müssen beim Integrationstest berücksichtigt werden. Komponenten können sowohl auf Client-Seite als auch auf Server-Seite notwendig sein. Das sind zum Beispiel für den Client:

- ActiveX[GL]-Controls

- ausführbare EXE-Dateien
- Java-Archive[GL]
- Dynamic Link Libraries (DLLs)
- Plugins[8]
- Web-Archive[GL]

Auf der Server-Seite können Komponenten wie Firewall und Proxy-Server[GL] eingesetzt werden.

8.3.3 Werkzeuge für den Integrationstest

Treiber und Platzhalter

Bei der Planung der Tests muss berücksichtigt werden, dass je nach Integrationsstrategie Testtreiber oder Platzhalter notwendig sind und programmiert werden müssen. Ggf. können sie schon im Komponententest eingesetzt und für den Integrationstest übernommen werden.

Monitore und Systemprotokolle

Der Testschwerpunkt liegt beim Integrationstest auf der Kommunikation der Schnittstellen. Daher ist es sehr hilfreich, den Datenverkehr auf den Schnittstellen aufzuzeichnen, sei es durch sogenannte Monitore (s. Abschnitt 11.1.5) oder eigens programmierte Protokollausgaben, die im produktiven Betrieb abgeschaltet werden können.

Capture Replay Tools

Für den Integrationstest können Capture Replay Tools zur Testautomatisierung (Abschnitt 12.4.2) eingesetzt werden. Oftmals können aber technische Ausnahmesituationen effizienter manuell als automatisiert getestet werden. Daher sollte der Einsatz von Capture Replay Tools im Integrationstest situationsabhängig erfolgen.

8.3.4 Qualitätsanforderungen zum Integrationstest

Die Qualitätsanforderung an den Integrationstest ist die 100-prozentige Erfüllung aller für die einzelnen Integrationsstufen definierten Testfälle, die zuvor von Fachleuten entworfen, geprüft und priorisiert wurden. Mit diesen nach Blackbox- und Whitebox-Verfahren entworfenen Testfällen werden die funktionalen Anforderungen an die Schnittstellen überprüft.

[8] Dem Test von Plugins ist das Kapitel 8.7 gewidmet.

8.3.5 Empfehlungen zum Integrationstest

nur getestete Komponenten integrieren

Für die einzelnen Schritte des Integrationstests dürfen nur von der Qualitätssicherung freigegebene Komponenten in eine nächst höhere Integrationsteststufe übernommen werden, denn wenn beim Integrationstest Fehlerwirkungen auftreten, sollten diese eindeutig der Schnittstellenfunktionalität zugeordnet werden können.

Technologieexperten einsetzen

Der Integrationstest ist ein typischer Greybox-Test, denn wie am Beispiel der oben beschriebenen SOAP-Kommunikation zu sehen ist, ist technisches Detailwissen für die Integration web-basierter Anwendungen notwendig. Daher sollten Spezialisten für die eingesetzten Technologien beim Integrationstest mitarbeiten und die systemtechnischen Testfälle entwerfen und priorisieren.

8.4 Funktionaler Systemtest

Hinweis

Der funktionale Systemtest ist in diesem Kapitel als Testtyp beschrieben. Unter dem umfassenderen Begriff Systemtest wird eine Teststufe verstanden, die neben dem funktionalen Systemtest weitere Testtypen umfasst (s. Kap. 14.4).

Ist ein Subsystem fertig getestet, so wird es als eigenständige Komponente in einen übergeordneten Integrationstest übernommen. Dieses Verfahren wird so lange angewendet, bis in der letzten Stufe das Gesamtsystem zusammengeführt ist und für den funktionalen Systemtest in einer produktionsnahen Testumgebung zur Verfügung gestellt wird.

Funktionaler Systemtest

Der funktionale Systemtest testet das Gesamtsystem gegen die funktionalen Anforderungen.

8.4.1 Testszenarien zum funktionalen Systemtest

Der funktionale Systemtest überprüft das Qualitätsmerkmal Funktionalität. Dazu werden Testfälle und Testszenarien nach den Methoden der zustands- und anwendungsfallbasierten Testfallermittlung (Abschnitte 4.1.4 und 4.1.5.) entworfen, mit denen die in der Anforderungsspezifikation beschriebenen Anwendungs- und Geschäftsvorfälle systematisch getestet werden. Diese Testszenarien zum Sys-

temtest stellen sicher, dass alle Systemschnittstellen und alle Funktionen über das gesamte System hinweg korrekt arbeiten und den fachlichen Anforderungen entsprechen.

↓

Beispiel Testszenario Systemtest

Das Testszenario, das den einfachsten Geschäftsvorfall der Finanzierungsanfrage abbildet und sich durch das gesamte Anwendungssystem bewegt, besteht aus folgenden Anwendungsfällen:

1. Der Kunde konfiguriert ein Auto mit dem Autokonfigurator.
2. Der Kunde berechnet eine Finanzierungsmöglichkeit im Finanzierungsrechner.
3. Der Kunde stellt zur Finanzierungsmöglichkeit eine Finanzierungsanfrage, die nach einer positiv beantworteten Verfügbarkeitsanfrage gespeichert wird.
4. Die Vertragsabwicklung erstellt anhand der Finanzierungsanfrage ein verbindliches Angebot und versendet es per E-Mail an den Kunden.

Für den funktionalen Systemtest müssen weitere Testszenarien entworfen werden, welche die komplexeren Geschäftsvorfälle abbilden. Sie beinhalten zum Beispiel, dass mehrere Finanzierungsmöglichkeiten gespeichert werden und später eine davon für ein Angebot ausgewählt wird, dass eine Verfügbarkeitsanfrage nicht positiv beschieden wird und dass ein Angebot per Post versendet wird.

↑

8.4.2 Werkzeuge und Qualitätsanforderungen zum funktionalen Systemtest

Capture Replay Tools, Monitore

Zum funktionalen Systemtest kommen, bis auf die nicht mehr benötigten Treiber und Platzhalter, die gleichen Werkzeuge wie beim Integrationstest zum Einsatz, d.h. Capture Replay Tools und Monitore.

Testfallabdeckung

Zum funktionalen Systemtest wird der Nachweis eines festgelegten Testfallabdeckungsgrades gefordert, wie zum Beispiel eine Testfallabdeckung der Systemtestszenarien von 80%.

Anforderungsabdeckung

Die beschriebenen Systemtestszenarien wiederum decken zu 100% die spezifizierten Anwendungsfälle ab.

8.5 Link-Test

Hyperlinks[GL] (kurz Links) sind ein wichtiges Steuerungsinstrument von Web-Anwendungen. Websites beinhalten interne und externe Links, die nicht ins Leere laufen dürfen. Neben der Funktionalität der Links muss gewährleistet sein, dass keine unerlaubten Link-Typen benutzt werden.

Link-Test

Der Link-Test überprüft die Rechtmäßigkeit und Fehlerfreiheit von internen und externen Links.

8.5.1 Link-Typen

Der Link-Test stellt sicher, dass jeder auf der Website angebrachte Link sein Ziel findet und dass keine „Gemeinheiten" in der Website versteckt sind, denn Deep Links, Inline Links, Framing und Metatags mit Namen und Produkten von Mitbewerbern sind verboten.

Deep Link

Ein Deep Link verweist direkt auf eine fremde, untergeordnete Web-Seite. Dabei wird die Homepage der verlinkten Seite umgangen. Wird der Benutzer nicht darauf hingewiesen, entsteht für ihn der Eindruck, dass der angezeigte Inhalt zur verweisenden Website gehört. Ein Deep Link kann somit Urheberrechte verletzen, denn das Recht der körperlichen Verwertung (Vervielfältigung, Verbreitung, Ausstellung) liegt ausschließlich beim Verfasser.

Inline Link

Ein Inline Link stellt eine Grafik, die einen anderen Ursprung als die Website selbst hat, als Teil dieser Website dar. Wird der Benutzer nicht darauf hingewiesen, dass sich diese Grafik nicht auf dem gleichen Web-Server wie die Website befindet, können die Urheberrechte verletzt werden.

Framing

Wie das Setzen von Inline Links ist auch das Framing ohne Erlaubnis des Betreibers der verlinkten Seite rechtlich unzulässig. Beim Framing wird eine fremde Web-Seite oder gar eine komplette Website in einem Frame der eigenen Website angezeigt.

Metatag

Metatags enthalten Angaben im Quelltext eines HTML-Dokuments, die der Benutzer nicht sieht, die aber Web-Servern und Suchmaschinen Auskunft über die Website liefern. Solche Metatags dürfen nicht auf Namen und Produkte von Mitbewerbern verweisen, da sonst das Markenrecht verletzt wird. So könnten auf diesem indirekten Wege durch Suchmaschinen Benutzer irregeführt und auf Webseiten umgeleitet, sozusagen „gelinkt" werden.

Mit diesen Erkenntnissen lässt sich eine Checkliste zum Link-Test zusammenstellen:

✓Checkliste Link-Test

- Werden keine Deep Links verwendet?
- Werden keine Inline Links verwendet?
- Wird kein Framing eingesetzt?
- Finden alle internen Links Ihr Ziel?
- Werden interne Links im gleichen Fenster geöffnet?
- Ist jeder externe Link als solcher gekennzeichnet?
- Finden alle externen Links Ihr Ziel?
- Werden externe Links in einem neuen Fenster geöffnet?
- Funktionieren alle Links, die eine E-Mail generieren (Mailto-Links)?
- Funktionieren alle Links, die ein Plugin aufrufen (z. B. einen Mediaplayer oder einen PDF-Reader)?
- Ist hinter jedem Link, der zu einem Dokument führt, der Dokumenttyp und die Dateigröße angegeben?
- Wenn eine Grafik mit einem Link versehen ist, ist der Link für den Benutzer erkenntlich – insbesondere, wenn die Darstellung von Grafiken in den Browser-Einstellungen unterbunden ist?
- Sind Grafiken mit mehreren Links (Imagemaps) erkennbar und sind die einzelnen Links eindeutig zu unterscheiden?
- Werden keine Namen und Produkte von fremden Unternehmen oder Mitwettbewerbern in Metatags verwendet?

8.5.2 Werkzeuge für den Link-Test

Link Checker

Mit dem Einsatz von Link Checkern lässt sich die Funktionalität von Links ohne großen Aufwand feststellen. Im Internet gibt es zahlreiche kostenlose Möglichkeiten zur Überprüfung von Links, wie zum Beispiel den frei verfügbaren Link Checker des W3C [URL: W3C-checklink] oder das einfach zu bedienende Freeware-Tool Xenu's Link Sleuth ([URL: XenuLink]).

Weitere Tools sind unter [URL: TestToolsSQA] in der Rubrik „Link Checkers“ zu finden.

8.5.3 Qualitätsanforderungen zum Link-Test

Die Qualitätsanforderungen an den Link-Test bestehen aus dem Erfüllungsgrad der Checkliste „Link-Test“ zu 100%. Zum Nachweis müssen die rechtliche Unbedenklichkeit der Links und Metatags in einem Review-Protokoll und die Funktionalität der Links in einem – evt. automatisch erstellten – Testprotokoll bestätigt werden.

8.5.4 Empfehlungen zum Link-Test

Link-Test mit Content-Test verbinden

Die Prüfung auf nicht erlaubte Link-Typen und verbotene Metatags hat starken Bezug zum Content-Test (Kap. 9.1), der die Rechtskonformität eines Web-Angebotes sicherstellt. Daher sollte die Checkliste „Link-Test“ schon beim Content-Test zu Rate gezogen werden. Der abschließende Link-Test kann allerdings erst nach der Fertigstellung der Web-Site erfolgen.

Link-Tests im laufenden Betrieb

Die Überprüfung der Links muss besonders in Hinsicht auf externe Links auch während des laufenden Betriebes regelmäßig erfolgen. Diese Tätigkeit kann ohne großen Aufwand mit Link Checkern automatisiert werden.

8.6 Cookie-Test

Cookies[GL] erlauben einem Web-Server mit Hilfe des Browsers Informationen auf dem Rechner des Nutzers in einer Textdatei abzulegen. Das kann zum Beispiel bei einem Online-Shop notwendig sein, der wissen muss, welche Artikel ein Benutzer bereits in den Warenkorb gelegt hat.

Cookie-Test

Der Cookie-Test überprüft die korrekte fachliche und technische Funktionalität von eingesetzten Cookies.

8.6.1 Überprüfung von Cookies

Die Ausführung einer Funktion, die Cookies benötigt, darf durch unerwartete Einstellungen des Browsers nicht verhindert werden.

Sind zum Beispiel Cookies auf dem Client gesperrt, muss der Benutzer darauf hingewiesen werden.

Das kann dadurch geschehen, dass der Benutzer erst im Fehlerfall die Nachricht erhält, dass seine Browser-Einstellungen das Anlegen von Cookies nicht zulassen. Eine andere Möglichkeit ist die aktive Ansprache des Benutzers, indem er mit dem Aufruf der entsprechenden Funktion einen Hinweise bekommt, dass Cookies benötigt werden und ob seine Browser-Einstellungen den Einsatz von Cookies zulassen. Diese zweite Lösung hat den Vorteil, dass der Benutzer auf jeden Fall die Information erhält, dass Cookies eingesetzt werden und ggf. Maßnahmen ergreifen kann.

Der Cookie-Test prüft also nicht nur die fachliche und technische Funktionalität eines Cookies, sondern auch die Erfüllung der Informationspflicht gegenüber dem Benutzer:

✓ *Checkliste Cookie-Test*

Funktion

- Werden die Inhalte des Cookies von der Anwendung korrekt gespeichert und wieder ausgelesen?
- Wird das Cookie nach der Sitzung korrekt gelöscht oder besitzt das Cookie ein Verfallsdatum?
- Arbeitet die Anwendung korrekt, wenn der Nutzer das Cookie löscht, insbesondere während der Sitzung?
- Funktioniert das Cookie, wenn die Sitzung unterbrochen und wieder aufgenommen wird?
- Wird der Benutzer bei dynamischer IP-Adresse richtig identifiziert?

Unterrichtung

- Wird der Benutzer darauf hingewiesen, wenn er in seinen Browser-Einstellungen Cookies unterbunden hat?
- Wird der Benutzer unterrichtet, wenn Cookies verwendet werden?
- Findet die Unterrichtung statt, bevor die Cookies gesetzt werden?
- Wird darauf hingewiesen, wie die Cookies verwendet werden?
- Ist festgelegt, wie lange die Website verlassen werden darf, ohne dass Informationen verloren gehen, und wird der Benutzer ggf. beim Verlassen der Website darüber informiert?

Zu jedem Prüfpunkt dieser Checkliste müssen Testfälle beschrieben und getestet werden.

8.6.2 Werkzeuge für den Cookie-Test

Cookie Viewer

Unter dem Stichwort „cookie viewer“ sind im Internet Werkzeuge zu finden, mit denen die Inhalte von Cookies überprüft und die Cookies gelöscht werden können.

Diese Möglichkeiten bieten auch einige Browser. Zum Beispiel können mit der Funktion „Cookies anzeigen“ des Browsers Firefox unter anderem Herkunft, Inhalt und Gültigkeitsdatum eines ausgewählten Cookies angezeigt werden (s. Abb. 8.9).

Abb. 8.9: Cookie

8.6.3 Qualitätsanforderungen und Empfehlungen zum Cookie-Test

Die Qualitätsanforderung an den Cookie-Test ist der Nachweis, dass alle zum Cookie-Test definierten Testfälle erfolgreich getestet wurden (100% Testfallabdeckung).

Bei der Beschreibung der Testfälle ist darauf zu achten, dass sie alle Punkte der obigen Checkliste abdecken. Neben diesen Standardtestfällen müssen auch die anwendungsspezifischen Anforderungen an die Cookies durch fachliche Testfälle abgedeckt werden.

Cookies werden von einzelnen Komponenten einer Website gesetzt und gelesen, daher sollten Cookie-Tests in der Testsstufe Komponententest durchgeführt werden.

8.7 Plugin-Test

Plugin

In der Regel hat jeder Internet-Benutzer Plugins installiert, die zusätzlich zum Browser eingesetzt werden. Ein Plugin ist ein Programmmodul, das in eine bestehende Anwendung integriert werden kann und deren Funktionsumfang erweitert. Es ist in der Regel ohne die dafür vorgesehene Anwendung nicht funktionsfähig. Ein Plugin ist vom verwendeten Betriebssystem abhängig und muss separat von der Anwendung auf dem Client installiert werden.

Ein bekanntes Beispiel ist das Adobe Reader Plugin, das für verschiedene Browser installiert werden kann. Es wird benötigt, um Dokumente lesen zu können, die im PDF[GL]-Format zur Verfügung gestellt werden.

Für Plugins müssen spezielle Tests durchgeführt werden.

Plugin-Test

Der Plugin-Test überprüft eine Web-Anwendung auf fehlerfreies Verhalten und korrekte Benutzerführung in Bezug auf benötigte Plugins.

8.7.1 Szenarien zum Plugin-Test

Beim Plugin-Test werden zwei generelle Situationen geprüft. Zum einen kann ein Plugin auf dem Benutzer-Client fehlen und zum anderen kann eines zu viel installiert sein.

fehlendes Plugin

Falls ein von einer Anwendungskomponente benötigtes Plugin nicht auf dem Client installiert ist, muss der Benutzer einen entsprechenden Hinweis erhalten und auf der Website die Möglichkeit erhalten, das Plugin aus dem Internet zu laden.

unverträgliches Plugin

Falls auf dem Rechner des Benutzers eine Software-Komponente installiert ist, die mit der neuen Web-Anwendung unverträglich ist, muss diese Situation automatisch erkannt werden und vom Benutzer behoben werden können. Sonst kann es passieren, dass die Anwendung nicht oder nicht korrekt funktioniert, wie die folgende Testgeschichte eines Plugins belegt.

↓

Szenarien eines Plugin-Tests

Eine wahre Plugin-Testgeschichte:

Eine Web-Applikation ist in Java realisiert und benötigt für die Ausführung von Java-Applets[GL], die mit Swing[GL]-Klassen arbeiten, das aktuelle Java-Plugin in der Version 1.4.1.

Eigentlich soll nur die Funktion einer Web-Anwendung getestet werden, die zu ihrer Ausführung das besagte Java-Plugin benötigt. Auf dem Rechner des Testers ist – was dieser aber noch nicht weiß – die ältere Version 1.3 des benötigten Plugins installiert.

Szenario Plugin-Test - Teil 1

Der Test beginnt. Eine wichtige Funktion der Web-Anwendung kann nicht aufgerufen werden. Es stellt sich heraus, dass die Anwendung nicht die Existenz der Version 1.4.1 des benötigten Plugins auf dem Client-PC überprüft, sondern nur das Vorhandensein irgend eines Java-Plugins dieses Herstellers. Die ältere Version 1.3 auf dem Testrechner, auf welche die Web-Anwendung zugreift, lässt aber keine Swing-Klassen zu.

Szenario Plugin-Test - Teil 2

Der Fehler wird behoben, d.h. beim Aufruf der Web-Anwendung wird nun geprüft, ob die richtige Version des benötigten Plugins installiert ist. Ist das nicht der Fall, wird der Benutzer darauf hingewiesen, dass er ein neues Plugin mindestens in der Version 1.4.1 installieren muss. Es wird auch der richtige Download-Link im Rahmen des Hinweises zur Verfügung gestellt. Die Version 1.3.1 ist inzwischen vom Testrechner deinstalliert. So weit, so gut.

Szenario Plugin-Test - Teil 3

Der anschließende Test deckt das nächste Problem auf: Das Plugin kann nicht installiert werden, weil der Tester sich nicht mit Administratorrechten auf seinem Rechner angemeldet hat. Da der verzweifelte Benutzer nicht auf seine fehlenden Rechte hingewiesen wird, weiß er nicht, warum sich das Plugin nicht installieren lässt. Das ist zwar kein Fehler der Web-Applikation, sondern ein Versäumnis des Plugin-Herstellers, aber die Wirkung ist fatal.

Szenario Plugin-Test - Teil 4

Nachdem dieses Problem erkannt ist und der Benutzer einen Hinweis zu den benötigten Rechten erhält, meldet sich der Tester mit Administratorrechten an seinem Rechner an. Das neue Plugin wird erfolgreich installiert. Die Anwendung funktioniert aber trotzdem nicht richtig. Der Grund liegt im Betriebssystem. Das greift nun auf ein zweites Plugin eines anderen Software-Herstellers zu, das bereits früher auf dem Rechner installiert ist. Die Situation tritt allerdings erst auf, seitdem die Version 1.3.1 deinstalliert worden ist – versteht sich. Erst nachdem das unverträgliche Plugin in den Systemeinstellungen deaktiviert ist, kann die eigentliche Funktion getestet werden.

↑

Wie die Geschichte[9] zeigt, ist es für den Plugin-Test wichtig zu wissen, welche Plugins von Benutzern oft eingesetzt werden, welche Plugins ein Browser-Typ unter welchem Betriebssystem benötigt

[9] Die kleine Testgeschichte zum Java-Plugin ist noch nicht zu Ende. Sie wird im Kapitel 12.2, Regressionstest, fortgesetzt.

und welche Unverträglichkeiten mit der zu testenden Web-Anwendung auftreten können.

Zum Beispiel zeigen Statistiken zu einer bestimmten Website, dass im Jahr 2005 über 95 Prozent aller Besucher, die mit dem Microsoft Internet Explorer arbeiteten, die Microsoft Virtual Machine installiert hatten, um Java-Applikationen benutzen zu können. Netscape-User setzten zu fast 40 Prozent das „Java Plugin 1.4.1 for Netscape Navigator“ ein. Um nachzuweisen, dass sich diese beiden Plugins im Zusammenspiel mit der zu testenden Web-Applikation „vertragen“, wenn sie gleichzeitig auf einem Rechner installiert sind, müssen entsprechende Tests durchgeführt werden.

Die oben beschriebenen Situationen führen zu einer Checkliste „Plugin-Test“:

✓ Checkliste Plugin-Test

- Ist auf der Website angegeben, für welche Plugins in welcher Version die Fehlerfreiheit der Anwendung garantiert wird?
- Hat der Benutzer die Möglichkeit, die benötigten Plugins direkt von der Website aus dem Internet zu laden?
- Sind die Installationsprozesse der benötigten Plugins getestet?
- Wird der Benutzer auf die Rechte hingewiesen, die er zur Installation eines Plugins benötigt?
- Liegen alle notwendigen Zertifikate vor und können sie installiert werden (zum Beispiel für „Cut & Paste“-Funktionen aus einem Java-Applet heraus)?
- Ist die Funktionalität der benötigten Plugins in Kombination mit möglicherweise bereits installierten, alten Plugin-Versionen des Herstellers getestet?
- Ist die Funktionalität der benötigten Plugins in Kombination mit möglicherweise bereits installierten, neueren Plugin-Versionen des Herstellers getestet?
- Ist die Funktionalität der benötigten Plugins in Kombination mit möglicherweise bereits installierten Plugins anderer Hersteller sichergestellt?
- Sind mögliche Unverträglichkeiten mit anderen Software-Komponenten (keine Plugins) geprüft und Gegenmaßnahmen beschrieben?

↓ Ein Beispiel für Abhängigkeitsphänomene

Zum letzten Punkt der Checkliste ein begründendes Beispiel aus einem Projekt: Ein gängiger PDF-Reader konnte nicht geöffnet werden, solange eine bestimmte DLL auf dem Client-PC installiert war.

Diese DLL wurde von einem Programm benötigt, das von der zu testenden Web-Anwendung unabhängig war – eigentlich.

Die Ursache lag in der eingesetzten Betriebssystemversion und konnte erklärt, aber nicht behoben werden.

↑

8.7.2 Werkzeuge für den Plugin-Test

User Tracking Tools

Für die Vorbereitung und Planung der Plugin-Tests können User Tracking Tools (s. Abschnitt 9.4.6) wichtige Informationen liefern. Sie erstellen zum Beispiel Übersichten, die zeigen, welche Plugins am häufigsten von den Besuchern einer Website verwendet werden.

Capture Replay Tools

Die Testszenarien zum Plugin-Test können mit Capture Replay Tools (s. Abschnitt 12.4.2) automatisiert werden. Allerdings sollte für solche technischen Tests, die auch einfach manuell durchgeführt werden können und relativ selten wiederholt werden müssen, im konkreten Fall der Nutzen einer Automatisierung hinterfragt werden.

8.7.3 Qualitätsanforderungen zum Plugin-Test

Das Hauptaugenmerk des Plugin-Tests liegt auf der Interoperabilität[GL], die ein Teilmerkmal des Qualitätsmerkmals Funktionalität ist. Die Interoperabilität eines zu testenden Plugins wird durch die 100-prozentige Abdeckung der Testfälle nachgewiesen, die auf Basis der obigen Checkliste entworfen werden.

8.7.4 Empfehlungen zum Plugin-Test

Der Plugin-Test wird schon mit dem Komponententest beginnen, sofern Komponenten der Anwendung Plugins aktiv nutzen. Während des laufenden Betriebes des Systems muss der Plugin-Test mit überarbeiteten Testszenarien immer dann wiederholt werden, wenn sich die Systemlandschaft verändert[10].

Experten für Plugin-Test

Der Plugin-Test sollte von Testern mit entsprechendem technischen Wissen durchgeführt werden. Der Test ist sehr wichtig, denn wenn das komplexe, technische Zusammenspiel von Software-Kom-

---

[10] ...wie das im Kap. 12 fortgeführte Beispiel zum Plugin-Test unterstreicht

ponenten, Browsern und Betriebssystemen nicht funktioniert, ist im schlimmsten Fall die gesamte Web-Anwendung unbrauchbar.

SAGA-Plugins

Eine Positiv-Liste von standardisierten Plugins (SAGA[11]-Plugins) stellt das Bundesministerium des Inneren unter [URL: Plugin-SAGA] zur Verfügung. Darin sind Plugins aufgeführt, deren stabiles Laufverhalten getestet wurde und bei denen keine Verstöße gegen deutsche Datenschutzgesetze festgestellt werden konnten. Diese Plugins können ohne Bedenken von einer Web-Anwendung aufgerufen werden.

Informationen

Informationen zu Java-Plugins stehen unter [URL: Plugin-Java]. Aktuelle Listen der verfügbaren und am häufigsten eingesetzten Plugins für Mozilla-Produkte wie den Firefox-Browser sind in [URL: PluginDoc] zu finden.

8.8 Sicherheitstest

Sicherheit

Sicherheit einer Software beschreibt als Teilmerkmal des Qualitätsmerkmales Funktionalität die Fähigkeit, nicht autorisierte Zugriffe, die vorsätzlich oder versehentlich auf Programme und Daten erfolgen, zu verhindern.

Je nach Aufgabenstellung, die eine Anwendung zu erfüllen hat, werden unterschiedliche Sicherheitsanforderungen an sie gestellt. Reine Informationsangebote müssen vor Manipulationen der angebotenen Inhalte sicher sein. Anwendungen, die sensible Daten verarbeiten, müssen vor fremden Zugriffen geschützt werden. Geschäftliche Transaktionen[GL] unterliegen noch schärferen Sicherheitsvorkehrungen, um Missbrauch zu verhindern.

Sicherheitstest

> Der Sicherheitstest prüft Eignung, Korrektheit, Unumgänglichkeit und Wirksamkeit der in einem System eingesetzten Sicherheitsvorkehrungen.

Sicherheitsvorkehrungen

Zu den Sicherheitsvorkehrungen eines Anwendungssystems zählen zum einen die speziell für die Anwendung programmierten Sicherheitsfunktionen und zum anderen die systemtechnischen Sicherheitsmechanismen.

[11] Standards und Architekturen für E-Government-Anwendungen

8.8.1 Schritte des Sicherheitstests

Der Sicherheitstest geht in drei Schritten vor:

1. Eignung der Sicherheitsvorkehrungen überprüfen
2. Systemschwachstellen identifizieren
3. Penetrationstest durchführen

1. Schritt: Eignung der Sicherheitsvorkehrungen überprüfen

Die fachlichen und technischen Sicherheitsanforderungen an das System müssen zu Beginn des Sicherheitstests vollständig definiert sein. Sicherheitsexperten prüfen die Sicherheitsanforderungen in einem Review und stellen fest, ob die notwendigen Sicherheitsfunktionen und Sicherheitsmechanismen bereitgestellt werden und ob sie angemessen sind.

2. Schritt: Systemschwachstellen identifizieren

Wenn die Sicherheitsvorkehrungen als geeignet eingestuft worden sind, werden anschließend potentielle Schwachstellen des Systems identifiziert, die ermöglichen könnten, dass die Sicherheitsvorkehrungen umgangen oder außer Kraft gesetzt werden.

3. Schritt: Penetrationstest durchführen

Penetrationstest

> Der Penetrationstest beweist oder widerlegt die Existenz der zuvor identifizierten Sicherheitsschwachstellen, indem gezielte Angriffe auf das Anwendungssystem simuliert werden.

Der Tester schlüpft beim Penetrationstest in die Rolle eines potentiellen Angreifers und versucht mit dessen Mitteln die Sicherheitsvorkehrungen zu umgehen. Das geschieht zum einen durch Manipulationen der Web-Anwendung und zum anderen durch gezielte Angriffe auf den Web-Server.

Der Tester wird zum Beispiel versuchen,

- in der Web-Applikation unerwartete Eingabekombinationen zu erzeugen,
- Reihenfolgen von Funktionen zu ändern,
- nicht erlaubte Funktionen auszuführen,

- Funktionen und Funktionsabfolgen zielgerichtet zu unterbrechen,
- an die Session-ID eines Benutzers zu gelangen, um dann mit dessen Rechten auf die Web-Anwendung zugreifen zu können (Session-Hijacking),
- den Passwortschutz zu umgehen,
- Passwörter durch Ausprobieren aller Möglichkeiten zu ermitteln (Brute Force Angriff).

Angriffe auf den Web-Server werden durchgeführt, um festzustellen, ob sie von den Sicherheitsvorkehrungen erkannt und abgewehrt werden:

DoS-Angriff

- Mit einem automatisierten Denial of Service (DoS) Angriff wird versucht, durch eine hohe Anzahl von Verbindungsanfragen dem Web-Server die Ressourcen zu entziehen und ihn für reguläre Anfragen zu blockieren. Wird der DoS-Angriff von vielen Computern gleichzeitig durchgeführt, spricht man von einem Distributed Denial of Service Angriff (DDoS).

SQL-Injektion

- Mit einer SQL-Injektion wird versucht, die Kontrolle über die Datenbank oder den Datenbank-Server zu erhalten. Dazu werden die Parameter einer Datenbankanfrage mit SQL-Steuerzeichen versehen, mit denen Datenbankbefehle in das System geschleust werden. Auf diese Weise können nicht zugängliche Daten ausgelesen oder verändert werden.

MitM-Angriff

- Beim Man-in-the-Middle-Angriff (MitM) wird versucht, die Verbindung einer gesicherten Verschlüsselung auszuspionieren. Dazu schaltet sich der Übeltäter in die Kommunikation zwischen Benutzer und Web-Server und manipuliert die ausgetauschten Informationen.

Zur Planung und Durchführung des Sicherheitstests dient die folgende Checkliste. Sie enthält sowohl Fragen, die bei der Schwachstellenanalyse berücksichtigt werden müssen, als auch Punkte, die anhand von Testfällen getestet werden müssen.

✓Checkliste Sicherheitstest

Sicherheitsanforderungen

- Sind die Anforderungen an die systemtechnische Sicherheit festgelegt?
- Ist die systemtechnische Sicherheit vertraglich geregelt?

- Sind die Anforderungen an die anwendungsspezifischen Sicherheitsfunktionen festgelegt?
- Sind die Sicherheitsanforderungen vollständig beschrieben?
- Sind die Sicherheitsanforderungen in den Geschäftsprozessen berücksichtigt?
- Wurden die Sicherheitsanforderungen nachweisbar von Fachleuten geprüft?
- Liegt eine Schwachstellenanalyse vor, in der fachliche und technische Sicherheitsaspekte betrachtet werden?
- Wurden die erkannten Schwachstellen bewertet und bei Bedarf beseitigt?
- Wurde die Schwachstellenanalyse nachweisbar von Fachleuten geprüft?
- Sind die Sicherheitsvorkehrungen für den laufenden Betrieb eingeplant?

Hardware / Sicherheits-Software

- Werden korrekt arbeitende Proxy-Server[GL] eingesetzt?
- Meldet der Relay-Host[GL] Sicherheitsverletzungen korrekt an den Administrator?
- Ist vor dem Relay-Host nur unbedingt notwendige Software installiert (zum Beispiel auf Web- und FTP-Server)?
- Sind nur unbedingt notwendige Daemons[GL] installiert?
- Sind die neuesten Versionen der Daemons installiert?
- Wird X-Windows[GL]-Verkehr verhindert?
- Sind alle bekannten Security-Patches installiert?
- Werden Checksummenverfahren angewendet?
- Kann die Firewall nicht von innen umgangen werden?

Verschlüsselung

- Werden bei Transaktionen Verschlüsselungsverfahren eingesetzt?
- Werden gängige Verschlüsselungsverfahren verwendet?
- Werden die zur Ver- und Entschlüsselung genutzten Schlüssel passwortgeschützt oder verschlüsselt gespeichert (Überschlüsselung)?
- Ist der Überschlüssel geschützt?

Passwortschutz

- Sind Mitgliederbereiche passwortgeschützt?
- Haben Passwörter mindestens eine Länge von acht Zeichen?
- Werden Sonderzeichen und Ziffern in den Passwörtern erzwungen?
- Werden Passwortdateien und -felder verschlüsselt gespeichert?
- Können alte Passwörter nicht mehr aktiviert werden?
- Wird der Zugang nach mehreren falschen Passworteingaben gesperrt?
- Kann der Passwortschutz nicht deaktiviert werden?
- Kann die Passwortprüfroutine nicht unterbrochen werden?
- Kann die Passworteingabe nicht umgangen werden?
- Wird eine regelmäßige Änderung des Passwortes beim Benutzer eingefordert?

Prüfungen Benutzerführung

- Werden falsche Eingaben sinnvoll abgefangen?
- Werden sicherheitsrelevante Eingaben Client- und Server-seitig geprüft?
- Bleiben bei unerwarteten Browser-Navigationen keine Transaktionen offen?
- Können Transaktionen nach einem Systemfehler nicht wieder aufgenommen werden?
- Hat der Benutzer die Möglichkeit, sich abzumelden?

Session Handling

- Sind Session-Ids eindeutig, zufällig generiert und für den Benutzer nicht sichtbar?
- Ist eine Session eindeutig einem Benutzer zugeordnet?
- Beendet ein Logout alle Sessions?
- Bleiben nach Beendigung der Session keine sensiblen Daten im Cache?
- Wird die Session automatisch beendet, wenn in einem vorgegebenen Zeitraum keine Eingabe durch den Benutzer erfolgt?
- Wird der Benutzer über den Zeitraum der automatischen Beendigung der Session informiert?

Protokollierung

- Werden Protokolldateien geschrieben?
- Enthalten die Protokolldateien keine sensiblen Daten?
- Werden die zu protokollierenden Aktivitäten (Anmeldungen, Abmeldungen, Transaktionen, Datenübertragungen,...) lückenlos aufgezeichnet?
- Haben Unbefugte keinen Zugriff auf die Protokolldateien?
- Können Administratoren die Protokollierung nicht unbemerkt manipulieren oder deaktivieren?

Zugriffsrechte

- Können Zugriffsrechte (auf Server, Datenbanken, Firewall, Dateien,...) nur von berechtigten Personen vergeben und geändert werden?
- Werden alte Zugriffsrechte gesichert und können sie zurückgespielt werden?
- Ist sichergestellt, dass Änderungen von Daten (zum Beispiel in der Datenbank) ausschließlich von autorisierten Personen vorgenommen werden können?
- Können Zugriffsrechte nicht über das Web administriert werden?

Penetrationstest

- Wurde ein Penetrationstest von Spezialisten durchgeführt?
- Wurden alle potentiellen Schwachstellen getestet und ggf. beseitigt?
- Wurden Angriffe auf den/die Server erkannt und abgewehrt?
  - Brute Force Angriff
  - Denial of Service Angriff (DoS)
  - Man-in-the-Middle-Angriff (MitM)
  - SQL-Injektion

Informationen

Wichtige Informationsquellen zum Thema Sicherheit liefern das Open Web Application Security Project (OWASP) und die Landesbeauftragten für den Datenschutz in Deutschland.

Das Open Web Application Security Project hat einen Guide zur Sicherheit im Web ([URL: OWASP]) und eine Checkliste zum Penetrationstest ([URL: OWASP-CL]) veröffentlicht.

✓ *Checkliste Penetrationstest (extern)*

Der Landesbeauftragte für den Datenschutz Niedersachsen stellt zum Thema Datenschutz/Datensicherheit eine Firewall-Checkliste zur Verfügung ([URL: LFDN]). Darin werden datenschutzrechtliche Anforderungen, technische und organisatorische Maßnahmen gegen Angriffe auf das Firewall-System und aus dem Internet auf das gesicherte Netz sowie Sicherungsmaßnahmen zur Paketfilterung behandelt.

✓ *Checkliste Firewall (extern)*

8.8.2 Werkzeuge für den Sicherheitstest

Security Test Tools

Zur Unterstützung des Penetrationstests werden Sicherheitskonfigurations- und Protokollierungswerkzeuge eingesetzt. Sie überprüfen die Systemkonfiguration und suchen nach Schwachstellen, wie zum Beispiel fehlenden Passwortabfragen oder offenen Ports, die einen unerlaubten Zugang zum Web-Server ermöglichen. Eine Liste von kommerziellen Werkzeugen zur Web-Sicherheit ist unter [URL: TestToolsSQA] in der Rubrik „Web Site Security Test Tools“ zu finden. Das U.S. Department of Energy veröffentlicht unter [URL: SecurityTools] ebenfalls eine Liste von Sicherheitswerkzeugen, worin auch das – wahrscheinlich bekannteste – Security Tool SATAN zu finden ist.

SATAN

SATAN (Security Administrator Tool for Analyzing Networks) wurde von Dan Farmer und Wietse Venema zur Analyse von Netzwerken geschrieben. SATAN ist ein kostenloses Werkzeug, das bekannte Sicherheitslücken aufspürt, dokumentiert und in einem Tutorial Gegenmaßnahmen zu jeder aufgespürten Schwachstelle vorschlägt.

8.8.3 Qualitätsanforderungen zum Sicherheitstest

Der Erfüllungsgrad von 100% der Checklisten „Sicherheitstest“, „Penetrationstest“ und „Firewall“ ist die erste Qualitätsanforderung an den Sicherheitstest.

Die zweite Anforderung ist, dass keine Person, insbesondere kein beauftragter Hacker, in das System eindringen konnte. Für diese Anforderung müssen in der Testspezifikation konkrete Testendekriterien festgelegt werden, wie zum Beispiel Dauer und Intensität der Hackerangriffe.

8.8.4 Empfehlungen zum Sicherheitstest

Sicherheitstests auch in der Betriebsphase

Der Sicherheitstest sollte schon in frühen Projektphasen mit der Analyse der Sicherheitsvorkehrungen beginnen, während der Penetrationstest in der Teststufe Abnahmetest und/oder im Pilotbetrieb in der produktiven Systemumgebung stattfindet. Die Überprüfung der Sicherheit darf während der gesamten produktiven Phase des Anwendungssystems nicht aus den Augen verloren werden, weil auf Grund der sich permanent verändernden Internettechnologien neue Sicherheitslücken entstehen können.

„legale Hacker" engagieren

Kritische Web-Applikationen müssen einem intensiven Sicherheitstest unterliegen, der entsprechend aufwendig und teuer ist. Weil für die Durchführung von Sicherheitstests Spezialistenwissen notwendig ist, kann es effektiver sein, „legale Hacker" zu engagieren als die Tests mit eigenen Kräften durchzuführen.

Bei weniger kritischen Anwendungen ist der Einsatz der oben abgebildeten Checkliste „Sicherheitstest" sehr hilfreich und kostengünstig.

Informationen

Weiterführende Informationen zum Thema Sicherheit liefert neben den bisher genannten Quellen auch das Bundesamt für Sicherheit in der Informationstechnik in [URL: BSI].

8.9 Zusammenfassung

- Zu den Testtypen, die als Schwerpunkt das Qualitätsmerkmal Funktionalität prüfen, gehören Klassentest, Komponententest, Integrationstest, Systemtest, Link-Test, Cookie-Test, Plugin-Test und Sicherheitstest.
- Der Klassentest wird vom Entwickler parallel zur Programmierung durchgeführt.
- Mit den entsprechenden Testwerkzeugen (Frameworks) werden Klassentests automatisiert und dokumentiert.
- Testfälle zum Komponententest können schon vom Entwickler im Klassentest durchgeführt werden.
- Nach dem Klassentest werden die Software-Komponenten an das Testteam in den Komponententest übergeben.
- Geprüfte Komponenten werden im Integrationstest zu Subsystemen zusammengeführt und getestet. Die Integration aller Sub-

systeme und Komponenten, die in mehreren Stufen erfolgen kann, bilden das Gesamtsystem.

- Der funktionale Systemtest testet die funktionalen Anforderungen des Gesamtsystems.
- Komponenten-, Integrations- und funktionaler Systemtest werden vom Testteam durchgeführt.
- Link-Test, Cookie-Test und Plugin-Test werden, falls die entsprechenden Technologien eingesetzt werden, vom Testteam durchgeführt.
- Der Link-Test prüft neben der Funktionalität die Ordnungsmäßigkeit der gesetzten Links.
- Der Plugin-Test stellt neben der Funktionalität benötigter Plugins auch die Verträglichkeit der Web-Anwendung mit bereits installierten Plugins sicher.
- Die Sicherheit eines Anwendungssystems wird von Sicherheitsexperten im Penetrationstest überprüft.
- Bei „brisanten“ Web-Anwendungen sollten professionelle „Hacker“ versuchen, Sicherheitslücken aufzudecken.

... und vor dem Sicherheitstest alle Daten sichern!

9 Tests zur Benutzbarkeit

„Nutzlose Objekte und nutzlose Subjekte lassen sich in drei Kategorien einteilen: Die einen arbeiten nicht, die anderen brechen zusammen und der Rest verschwindet einfach."

Russell Baker (*1925),

Die Benutzbarkeit ist für die Akzeptanz einer Web-Anwendung elementar. Wenn der Benutzer eine Website nicht erreichen, nicht mit seinem Standard-Browser ansehen, die Benutzerführung nicht verstehen oder seine gewünschten Informationen nicht abrufen kann, dann wechselt er zum Mitbewerber. Eine Web-Anwendung zeichnet sich auch dadurch aus, dass sie intuitiv bedient und schnell erlernt werden kann.

Dieses Kapitel beschreibt die Testtypen Content-, Oberflächen-, Browser-, Usability-, Zugänglichkeits- und Auffindbarkeitstest, die das Qualitätsmerkmal Benutzbarkeit mit den Teilmerkmalen Bedienbarkeit, Erlernbarkeit und Verständlichkeit sowie den web-spezifischen Merkmalen Auffindbarkeit, Barrierefreiheit und Rechtskonformität prüfen.

9.1 Content-Test

Es wird darauf hingewiesen, dass die folgenden Ausführungen zum Web-Recht und zur Gesetzesgebung nicht rechtsverbindlich sind und, ebenso wie die abgebildeten Checklisten, keinen Anspruch auf Vollständigkeit erheben.

✐ *Hinweis*

Der Content-Test ist eine besondere Ausprägung des Dokumententests (s. Kap. 7.2), der für Web-Auftritte von besonderer Bedeutung ist. Er wird in Form von Reviews oder schriftlichen Stellungnahmen anhand spezieller Checklisten durchgeführt.

Eine Website ist ein Dokument, das sich einem weltweiten Benutzerkreis darbietet. Im Inhalt einer Website (Content) müssen unabhängig von fachlichen und technischen Funktionalitäten die besonderen Belange des WWW berücksichtigt werden. Dazu gehören die Erwartungen, die Benutzer an den Inhalt einer Website haben, die Aufklärungspflicht des Anbieters sowie die Rechtmäßigkeit des Informations- und Dienstleistungsangebotes.

Content-Test

Der Content-Test stellt die Erfüllung der Benutzererwartungen und der Aufklärungspflicht sowie die Rechtskonformität einer Website sicher.

9.1.1 Erfüllung der Benutzererwartungen

Benutzererwartungen

Der Benutzer einer Website will sofort erkennen, ob die angebotenen Inhalte für ihn von Nutzen und aktuell sind. Er will mit der Anwendung intuitiv umgehen können, die gesuchten Informationen schnell erhalten und problemlos auf sie zugreifen können. Er hat konkrete Vorstellungen, welche Inhalte eine Website bieten muss. Dazu gehören:

- Detaillierte Informationen zu Produkten und Leistungen
- Kontaktdaten von Ansprechpartnern
- Aktuelle Unternehmensnachrichten
- Preise
- Tipps und Frequently Asked Questions (FAQ)
- Bestellmöglichkeiten

Auf Basis dieser Erwartungen und der von der ISO 9241 Teil 10[1] geforderten Qualitätsmerkmale Lernförderlichkeit und Individualisierbarkeit ergibt sich die Checkliste „Web-Content“:

[1] s. Kap. 2.1

✓ *Checkliste Web-Content*

Information

- Ist der Zweck der Website bzw. jedes Teilbereiches der Website deutlich erkennbar (kommerziell, informativ, unterhaltend)?
- Wird der Inhalt des Web-Angebotes seinem Zweck gerecht?
- Ist der persönliche Nutzen der Website für den Benutzer offensichtlich?
- Wird für jede Zielgruppe ein Mehrwert angeboten?
- Findet jede Zielgruppe schnell und ohne viel Aufwand ihre gewünschten Informationen?
- Erhält der Nutzer detaillierte Produkt- und Leistungsinformationen?
- Ist gewährleistet, dass die Inhalte der Website stets aktuell sind?
- Wird angezeigt, wann die Inhalte der Website zuletzt aktualisiert wurden?
- Sind die fachlichen Inhalte von Fachleuten geprüft und freigegeben?
- Ist eine Suchfunktion vorhanden?
- Gibt es folgende Rubriken:
  - FAQ?
  - Glossar?
  - Impressum?
  - Index?
  - News?
  - Tipps?

- Gibt es aktuelle Unternehmensnachrichten und Newsletter?
- Ist eine Bestellung von Informationsmaterial möglich?
- Sind Preise angegeben?
- Wird der Nutzer über den Verbleib seiner Eingabedaten informiert?
- Werden die Quellen zu den angebotenen Informationen genannt?
- Ist der Aufklärungspflicht genüge getan (entsprechend der Checkliste „Aufklärungspflicht“[2])?

[2] s. Kap. 9.1.3

Kommunikation

- Sind die Kontaktdaten der Ansprechpartner angegeben?
- Sind für jede Nutzer-/Zielgruppe persönliche Ansprechpartner benannt?
- Ist die komplette Adresse des Anbieters angegeben?
- Gibt es einen Lageplan und eine Anfahrtsskizze?
- Ist eine E-Mail-Kontaktadresse angegeben?
- Werden Kommunikationsmöglichkeiten (Forum, Formulare, Gästebuch) bereitgestellt?
- Wird eine Hotline angeboten und ist deutlich erkennbar, unter welchen Bedingungen sie zu erreichen ist (Zeit, Preis, Medien)?
- Wird der Nutzer bei der Durchführung von Geschäftsprozessen über die nächsten Schritte informiert?

Internationalität

- Wird die Website in Englisch angeboten?
- Werden die wichtigsten Informationen in weiteren Sprachen angeboten?
- Werden Informationen in der jeweiligen Sprache der Zielgruppe angeboten?
- Entsprechen Maß- und Zeitangaben internationalen Standards?

Lernförderlichkeit

- Gibt es innerhalb der Hilfefunktion ein Stichwortverzeichnis?
- Gibt es innerhalb der Hilfefunktion eine Gliederung?
- Gibt es eine kontextsensitive Hilfe?
- Sind alle beschriebenen Hilfe- und Fehlertexte verständlich?
- Ist eine Guided Tour durch die Web-Applikation vorgesehen?
- Ist die Guided Tour für die Zielgruppe verständlich?
- Können Testbuchungen/Testbestellungen vorgenommen werden?

Individualisierbarkeit - Persönliche Einstellungen

- Kann der Benutzer sich seine Inhalte individuell zusammenstellen, in seinem Profil speichern und wieder aufnehmen?

- Kann der Benutzer die Inhalte für seinen persönlichen Newsletter zusammenstellen?
- Kann der Newsletter abbestellt werden?

Die Checkliste ist für Websites relevant, die einen hohen Informationsanteil besitzen und neue Benutzer binden möchten. Sie hat im Gegensatz zu den folgenden Themen der Aufklärungspflicht und der Gesetzeskonformität keinen verbindlichen Charakter.

9.1.2 Einhaltung der Gesetze

Rechtskonformität

Eine Web-Anwendung muss den Anforderungen des Gesetzgebers genügen. Die Gesetze, die in einen Web-Auftritt beachtet werden müssen, sind in der Checkliste „Web-Recht“[3] zusammengefasst.

✓ Checkliste Web-Recht

Im Web zu beachtende deutsche Gesetze

- Bundesdatenschutzgesetz (BDSG)
- Gesetz über die Verbreitung jugendgefährdender Schriften und Medieninhalte (GjS)
- Gesetz zum elektronischen Geschäftsverkehr (EGG)
- Gesetz zum Schutze der Jugend in der Öffentlichkeit (JÖSchG)
- Jugendmedienschutzstaatsvertrag (JMStV)
- Jugendschutzgesetz (JuSchG)
- Markengesetz (MarkenG)
- Mediendienste-Staatsvertrag (MDStV)[GL]
- Signaturgesetz (SigG)
- Teledienstedatenschutzgesetz (TDDSG)
- Teledienstegesetz (TDG)[GL]
- Telekommunikations-Datenschutzverordnung (TDSV)[GL]
- Telekommunikationsgesetz (TKG)[GL]
- Urheberrechtsgesetz (UrhG)

[3] Die ausführliche Darlegung der in der Checkliste aufgeführten Gesetze würde den Rahmen des Buches sprengen. Die Liste ist als Grundlage für die Rechtsexperten gedacht, die bei der Veröffentlichung eines Web-Angebotes die Einhaltung der Gesetzte und Vorschriften überprüfen und nachweisen müssen.

- Zugangskontrolldiensteschutz-Gesetz (ZKDSG)

Sonstige rechtliche Bedingungen

- Werden EU-Rechte eingehalten?
- Werden internationale Rechte eingehalten?
- Ist das Wettbewerbsrecht eingehalten?
- Ist das Presserecht eingehalten?
- Ist die Signaturverordnung (SigV) eingehalten?
- Sind Preise nach den Vorgaben der Preisangabenverordnung (PAngV) angegeben?
- Liegen alle Copyrights für Audio-Dateien, Bilder, Karten, Markenzeichen, Texte und Videos vor?
- Sind alle Links rechtlich geprüft (s. Checkliste „Link-Test“[4])?

Informationen

Informationen und Gesetzestexte zu den aufgeführten Gesetzen sind auf den Websites der Bundesministerien [URL: GesetzeBMJ], [URL: GesetzeBMWI] und unter [URL: GesetzeIID] zu finden.

9.1.3 Erfüllung der Aufklärungspflicht

Der Betreiber einer Website hat dem Benutzer gegenüber eine Aufklärungspflicht. Zudem muss ein Web-Auftritt rechtlich und für den Benutzer erkennbar abgesichert sein.

Wichtige Punkte zur Aufklärungspflicht von Website-Anbietern, die sich aus den oben aufgeführten Gesetzen ergeben, sind in der folgenden Checkliste aufgeführt:

✓ Checkliste Aufklärungspflicht

Allgemein

- Ist ein Impressum bzw. eine Anbieterkennzeichnung vorhanden?
- Sind die Autoren der Website angegeben?
- Ist der Vertretungsberechtigte genannt, wenn der Diensteanbieter eine juristische Person ist?
- Sind die Daten für eine schnelle Kontaktaufnahme offensichtlich (wie zum Beispiel E-Mail, Telefon, Fax)?
- Ist die Umsatzsteueridentifikationsnummer angegeben?

[4] s. Abschnitt 8.5.1

- Ist die entsprechende Registernummer angegeben (wie zum Beispiel des Genossenschafts-, des Handels- oder Vereinsregisters)?
- Wird für zulassungspflichtige Dienste die zuständige Aufsichtsbehörde genannt?
- Existiert eine Abstandsklausel zu „verlinkten" Inhalten?
- Existiert eine Erklärung über die dargestellten Marken und Logos?
- Existiert der Haftungsausschluss bzgl. Aktualität, Korrektheit, Vollständigkeit und Qualität der bereitgestellten Informationen?
- Ist eine Absicherung gegenüber Downloads vorgenommen?
- Haben alle Seiten Copyright-Vermerke?

Datenschutz

- Gibt es eine Datenschutzerklärung?
- Gibt es Hinweise zur Speicherung und Löschung von personenbezogenen Daten?
- Sind Hinweise beim Setzen von Cookies vorgesehen?[5]

Verträge

- Sind Nutzungs- und Vertragsbedingungen angegeben und offensichtlich zugänglich?
- Sind die allgemeinen Geschäftsbedingungen (AGB) angegeben und offensichtlich zugänglich?
- Sind Preise bei jedem Bestellposten mit Mehrwertsteuer und mit allen zusätzlich anfallenden Preisen angegeben?
- Sind anfallende Versandkosten offensichtlich aufgeführt?
- Sind anfallende Lieferkosten offensichtlich aufgeführt?
- Sind Widerrufsinformation und Rückgabefrist offensichtlich angegeben?

↓

Beispiele zur Aufklärungspflicht

Zwei Beispiele zur Aufklärungspflicht sind die Abstandsklausel zu externen Links und die Download-Absicherung.

---

[5] siehe auch Kap. 8.6, Cookie-Test

1. Beispiel - Abstandsklausel:

„Haftungshinweis: Trotz sorgfältiger inhaltlicher Kontrolle übernehmen wir keine Haftung für die Inhalte externer Links. Für den Inhalt der verlinkten Web-Seiten sind ausschließlich die Betreiber der verlinkten Web-Seiten verantwortlich."

2. Beispiel - Download-Absicherung:

„Das Herunterladen der Dateien erfolgt auf eigene Gefahr. Wir übernehmen keinerlei Haftung für Schäden, die direkt oder indirekt durch die Benutzung dieser Dateien entstehen. Dies gilt insbesondere dann, wenn diese Dateien für strafbare Handlungen eingesetzt wurden."

↑

9.1.4 Qualitätsanforderungen zum Content-Test

Die Qualitätsanforderungen an den Content einer Website werden durch den Erfüllungsgrad der entsprechenden Checklisten definiert.

Für den Nachweis der Rechtskonformität muss eine 100-prozentige Abdeckung der rechtlichen Fragen erreicht werden. D.h. 100% der Fragen der Checklisten „Aufklärungspflicht" und „Web-Recht" werden positiv beantwortet bzw. von einem juristisch ausgebildeten Prüfer als nicht relevant begründet.

Die Checkliste „Web-Content" muss nicht so restriktiv gehandhabt werden. Zum Beispiel kann mit einer Abdeckung von 90% der Fragen die geforderte Qualität erreicht sein.

9.1.5 Empfehlungen zum Content-Test

Impressums-Assistenten

Im Internet gibt es Assistenten, die bei der Erstellung eines gesetzeskonformen Web-Impressums Unterstützung leisten, wie zum Beispiel bei [URL: Certiorina].

Rechts- und Marketing-Experten

Die Fragen zu den Web-Gesetzen und zum Web-Recht sind nicht immer einfach und nicht ohne rechtliches Hintergrundwissen zu beantworten. Daher muss die Durchführung des Content-Tests für diese Themen durch Rechtsexperten erfolgen. Da sich die Rechtsgrundlage permanent ändern kann, ist die Aktualität der Checklisten zu gewährleisten. Fachleute müssen sie regelmäßig vor jedem Einsatz in einem Projekt überprüfen und neu „justieren".

Prototyp prüfen

Der Content-Test sollte frühzeitig durchgeführt werden, wenn die Website als Design-Entwurf oder besser als Prototyp vorliegt, denn die Ergebnisse der Prüfung können entscheidenden Einfluss auf das Design haben.

finaler Content-Test zur Freigabe

Nach Fertigstellung der Web-Applikation und kurz vor ihrer Produktivstellung muss abschließend sichergestellt werden, dass die im Content-Test geprüften Inhalte nachträglich nicht verändert wurden. Daher empfiehlt sich ein wiederholter Check des Contents vor der Freigabe der Anwendung.

9.2 Oberflächentest

Eine Anwendung muss sich dem Benutzer in allen Bereichen gleich darstellen. Standardfunktionalitäten wie das Navigieren und das Drucken müssen einheitlich sein und funktionieren. Diese Anforderungen an eine Software werden in der Regel nicht explizit vom Auftraggeber spezifiziert, sondern als selbstverständlich vorausgesetzt. Zu prüfen sind sie dennoch im Oberflächentest.

Oberflächentest

Der Oberflächentest überprüft die Einhaltung von Dialogrichtlinien und die Korrektheit von Standardfunktionalitäten auf der Benutzeroberfläche einer Anwendung.

9.2.1 Dialogrichtlinien und Standardfunktionalitäten

Der Oberflächentest stellt sicher, dass sich eine Web-Applikation bei unterschiedlichen technischen Voraussetzungen korrekt darstellt und der Nutzer zielgerichtet und fehlerfrei durch die Anwendung navigieren kann. Zudem werden Design-Standards überprüft, die für jede Web-Anwendung in einer Organisation gelten sollten und unabhängig von der fachlichen Funktionalität sind. Diese Trennung von den fachlichen Funktionstests hat zwei Vorteile. Die Testfälle sind auf andere Web-Applikationen – zumindest innerhalb eines Unternehmens – übertragbar und der Oberflächentest kann von Personen durchgeführt werden, die sich nicht in die fachlichen Funktionen der Anwendung einarbeiten müssen.

Der Oberflächentest ist ein dynamischer Test und nicht mit dem statischen Content-Test (Kap. 9.1) gleichzusetzen, bei dem die Vollständigkeit und die Konformität der darzustellenden Web-Inhalte ohne Ausführung von Testfällen geprüft werden.

Der Oberflächentest ist den Tests zur Benutzbarkeit zugeordnet und nicht den Tests zur Funktionalität (Kap. 8), weil mit ihm nicht die fachlich angeforderten Funktionen getestet werden, sondern die Benutzbarkeit der Anwendung. Die Testfälle zum Oberflächentest werden anhand der Fragen der folgenden Checkliste entworfen, welche die Forderungen der ISO 9241 Teil 10[6] – Grundsätze der Dialoggestaltung – berücksichtigt:

✓ *Checkliste Oberflächentest*

Aufgabenangemessenheit

- Sind alle für die Ausführung einer Aufgabe benötigten Menüpunkte sichtbar und aktiviert?
- Sind alle für die Ausführung einer Aufgabe nicht benötigten Menüpunkte deaktiviert oder nicht sichtbar?
- Werden keine aufgabenfremden, unnötigen Eingaben gefordert (z. B. irrelevante, persönliche Angaben zu einer Bestellung)?
- Werden nicht benötigte Eingabefelder kontextsensitiv verdeckt?
- Werden Eingabeobjekte je nach Anforderung korrekt gesperrt bzw. entsperrt?

Erwartungskonformität

- Erscheinen alle wiederkehrenden Elemente auf jeder Seite an derselben Stelle in derselben Darstellung (Firmenlogo, Link zum Impressum, Steuerelemente,...)?
- Führt das Firmenlogo stets zur Homepage?
- Ist der Link zum Impressum am Ende jeder Seite vorhanden?
- Haben alle gleichen Elemente die selbe Darstellung (Farbe und Größe von Buttons, Menüpunkte, Textformate,...)
- Werden Texte und Oberflächenelemente auf allen Seiten gleich dargestellt?
- Sind Begriffe auf allen Seiten identisch, z.B. ist der "Warenkorb" immer und in allen Zusammenhängen der "Warenkorb"?
- Sind unterstrichene Wörter stets Links?
- Werden gängige Icons verwendet und sind sie aussagekräftig in Bezug auf ihre Verwendung?
- Sind Icons mit erklärenden Texten versehen?

---

[6] s. Kap. 2.1

Fehlertoleranz

- Werden Plausibilitätsprüfungen so bald wie möglich durchgeführt?
- Werden Eingaben möglichst vor dem Abschicken einer Seite überprüft?
- Steht der Cursor bei falscher Eingabe auf dem entsprechenden Feld?
- Ist die Prüfreihenfolge der Eingaben logisch und korrekt (Felder, Feldkombinationen)?
- Erfolgt die Fehlerprüfung in allen Windows/Dialogen gleich, entweder Fensterweise oder Eingabefeldweise?

Selbstbeschreibungsfähigkeit

- Wird dem Benutzer auf jeder Web-Seite angezeigt, wo er sich befindet?
- Wird der Status der Bearbeitung am unteren Bildrand korrekt angezeigt?
- Wird nach einer Suche die Anzahl der Treffer angezeigt?
- Wird bei Verarbeitungen, die von kurzer Dauer sind, stets die Sanduhr angezeigt und bei länger dauernden der Fortschrittsbalken?
- Sind die Pflichtfelder von den optionalen Feldern optisch zu unterscheiden?
- Werden besuchte Links als solche dargestellt?

Steuerbarkeit

- Können Tabellen spaltenweise sortiert werden?
- Kann für Suchvorgänge die Anzahl der anzuzeigenden Treffer pro Seite festgelegt werden?
- Gibt es Filtermöglichkeiten für große Treffermengen?
- Können Downloads unterbrochen werden?
- Kann das „Intro“ auf der Startseite übersprungen werden?
- Gibt es bei zeitintensiven Verarbeitungen eine Abbruchmöglichkeit und funktioniert dieser Abbruch?
- Sind Video- und Audiosequenzen abschaltbar?
- Können umfangreiche Eingaben gespeichert und wieder aufgenommen werden?

- Kann automatisch ein Bookmark auf die Startseite gesetzt werden?
- Werden schnell ladbare Alternativen für komplexe Seiten und Inhalte angeboten?

Navigation

- Sind Navigationselemente (Buttons, Menüpunkte) eindeutig, übersichtlich, verständlich und auffindbar?
- Entsprechen die Navigationselemente der inhaltlichen Gliederung (verzweigt z.B. jeder Menüpunkt zum richtigen Inhalt)?
- Funktionieren alle Navigationselemente (Buttons, Menüpunkte, Hyperlinks) richtig?
- Sind Navigations- und Präsentationsbereich getrennt?
- Ist die Navigationsleiste auf jeder Seite gleich und sichtbar?
- Existiert eine vollständige und funktionstüchtige Sitemap[GL]?
- Ist die Navigation der Web-Anwendung von der Browser-Navigation unabhängig?
- Werden horizontale Scroll-Balken vermieden?
- Steht in der Titelleiste des Browsers immer der Name der aktuellen Seite?
- Weiß der Benutzer zu jedem Zeitpunkt, an welcher Stelle er sich befindet?
- Kann über die Browser-Navigation vor- und zurückgeblättert werden, ohne dass Datenverluste oder Orientierungsprobleme entstehen?
- Existiert auf jeder Seite ein Link zur
  - nächst höheren Hierarchiestufe?
  - Hilfefunktion?
  - Homepage?

- Ist jede Funktion mit Tabulator-Tasten erreichbar?
- Erfolgt die Cursor-Steuerung mit den Tasten <TAB> und <SHIFT+TAB> in der richtigen Reihenfolge?
- Werden Shortcuts (Tastaturkurzbefehle) für wichtige Funktionen bereitgestellt und funktionieren diese?

Darstellbarkeit

- Werden die Seiten bei allen möglichen Bildschirmauflösungen korrekt und vollständig angezeigt?
- Sind die Inhalte insbesondere auch auf Bildschirmen mit kleiner Auflösung nutzbar?
- Ist die Anwendung bei allen möglichen Bildschirmauflösungen funktionstüchtig?
- Ist der Kontrast von Texten und Grafiken hinreichend für eine Schwarz/Weiß- und 256-Farb-Darstellung?
- Wurden durchgängig Cascading Style Sheets[GL] (CSS) zur Darstellung der Web-Inhalte verwendet?
- Sind Lesbarkeit und Handhabbarkeit bei Veränderung der Schriftart und Schriftgröße in den Browser-Einstellungen gewährleistet?
- Gibt es deutlich lesbare, alternative Texte für grafische Navigationselemente?
- Sind alle Links mit einer eindeutigen Beschreibung des Zielobjektes versehen?
- Sind Links in Grafiken deutlich erkennbar?
- Sind ungewöhnliche Links mit Hinweisen versehen (z.B. fremde Sprache, extreme Dateigröße, ungewöhnliches Format)?

Initialisierung

- Sind beim Start der Anwendung/einer Seite alle Defaultwerte richtig gesetzt?
- Sind alle Ausrichtungen korrekt?
- Sind Sortierungen richtig?
- Sind alle Eingabefelder sinnvoll vorbelegt (z. B. Datumsfelder)?
- Sind Radiobuttons richtig initialisiert?
- Werden Bilder und Grafiken mit einer Vorschaugrafik geladen?

Standardfunktionalitäten

- Funktionieren die Suchfunktionen korrekt?
- Funktionieren die Sortierfunktionen korrekt?
- Arbeitet die Funktion „Zurücksetzen“ stets richtig?
- Läuft die Guided Tour korrekt ab?

- Arbeiten die Druckfunktionen des Browsers mit der Anwendung korrekt zusammen?
- Arbeiten die Druckfunktionen innerhalb der Anwendung korrekt?
- Gibt es für den Druck der Web-Seiten ein extra CSS-Druck-Layout?
- Gibt es eine Druckvorschau?
- Können alle Inhalte einwandfrei gedruckt werden?
- Gehen keine Inhalte verloren, wenn Seiten für den Hochkant-Ausdruck vorgesehen sind?
- Gehen keine Inhalte verloren, wenn Seiten für den Quer-Ausdruck vorgesehen sind?

Tipp

Die Checkliste zum Oberflächentest kann mit wenigen „Handgriffen" so abgeändert werden, dass sie für Client/Server-Anwendungen einsetzbar ist. Es müssen hierzu „nur" die web-spezifischen Fragen herausgenommen oder entsprechend angepasst werden.

Der Test einer Web-Anwendung bei unterschiedlichen Browser-Einstellungen wird im folgenden Kapitel 9.3 behandelt. Daher kommen in der Checkliste „Oberflächentest" keine Fragen zu diesem Themenkomplex vor.

9.2.2 Werkzeuge für den Oberflächentest

Web Accessibility Toolbar

In der Web Accessibility Toolbar (WAT, [URL: AbITools]) können Browser-Einstellungen für Tests direkt vorgenommen werden, wie zum Beispiel die Auswahl unterschiedlicher Bildschirmauflösungen (Abb. 9.1). Das Werkzeug ist allerdings nur für den nicht kommerziellen Gebrauch kostenlos.

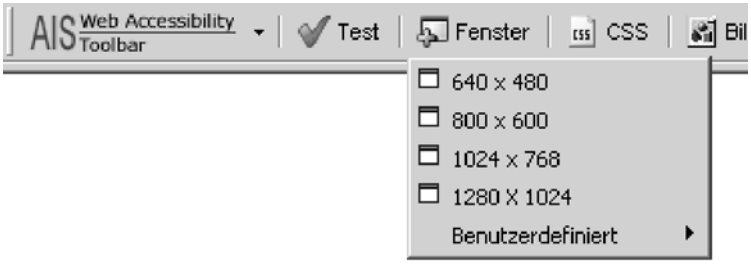


Abb. 9.1: WAT Bildschirmauflösung

CSS-Prüfung mit „verrückten“ Schriftarten

In der Checkliste zum Oberflächentest wird die Verwendung von Cascading Style Sheets (CSS) gefordert. Diese kann einfach überprüft werden, indem im Browser eine „verrückte“ Schriftart eingestellt wird, wie zum Beispiel „Geotype TT“. So wird sofort deutlich, welche Elemente nicht mit CSS-Anweisungen formatiert sind und welche Bereiche der Website bei Veränderung der Browser-Einstellungen empfindlich reagieren, d.h. unter Umständen nicht mehr korrekt dargestellt werden.

Die Abb. 9.2 zeigt, dass die abgebildete Web-Seite nicht mit CSS formatiert ist, weil die Auswahl der Schriftart „Geotype TT“ in den Browser-Einstellungen deutlichen Einfluss auf die Darstellung der Inhalte hat – sie ist unleserlich. Mit dem Einsatz von CSS würde sich die im Browser ausgewählte Schriftart nicht auf die Lesbarkeit der Seite auswirken.

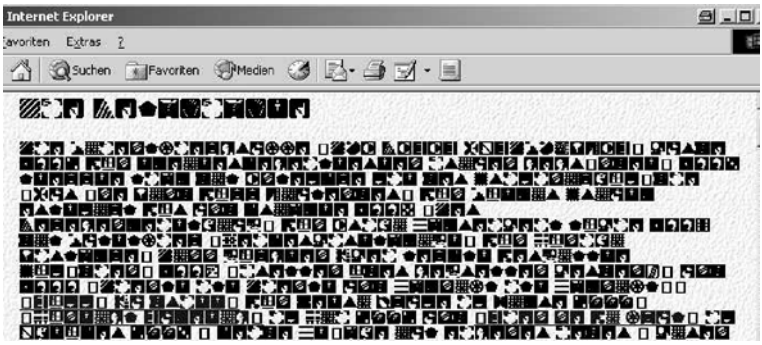


Abb. 9.2: Schriftart Geotype TT

Capture Replay Tools

Der Oberflächentest kann anhand der Checkliste „Oberflächentest“ manuell durchgeführt werden. Mit einem Capture Replay Tool können einige der Checkpunkte zum Oberflächentest automatisch überprüft werden, denn ein solches Werkzeug kann zum Beispiel die Einhaltung vorgegebener Standards (Farbgebung, Schriftart,...), die Existenz von Objekten (Buttons, Menüpunkte, Navigationshilfen,...) und die Eigenschaften von Objekten (Größe, Position,...) erkennen.

Wenn dem Projekt ein Capture Replay Tool zur Verfügung steht und die vereinbarten Standards für mehrere Web-Anwendungen getestet werden müssen, lohnt sich der Aufwand, standardisierte Skripts zur Oberflächenprüfung zu entwickeln und die Tests zu automatisieren. Dafür werden allerdings Testspezialisten benötigt. Im Testplan ist festzulegen, welche Prüfungen automatisiert werden sollen. Die entsprechenden Testfälle werden dann in der Checkliste zum Oberflächentest gestrichen. Folgende Punkte sind geeignete Kandidaten, die mit einem Capture Replay Tool automatisch überprüft werden können:

- Funktionieren alle Navigationselemente (Buttons, Menüpunkte, Hyperlinks) richtig?
- Ist die Navigationsleiste auf jeder Seite sichtbar?
- Gibt es alternative Texte für grafische Navigationselemente?
- Kann von jeder Seite die Homepage aufgerufen werden?
- Erscheinen alle wiederkehrenden Elemente auf jeder Seite an derselben Stelle in derselben Darstellung (Steuerelemente, Firmenlogo, Link zum Impressum,...)?
- Sind unterstrichene Wörter stets Links?
- Haben alle gleichen Elemente die selbe Darstellung (Farbe und Größe von Buttons, Menüpunkte, Textformate,...)
- Werden Texte und Oberflächenelemente auf allen Seiten gleich dargestellt?
- Wird der Bearbeitungsstatus am unteren Bildrand korrekt angezeigt?
- Sind alle Defaultwerte beim Laden einer Web-Seite/eines Formulars richtig gesetzt?
- Arbeitet die Funktion „Zurücksetzen“ stets korrekt?

9.2.3 Qualitätsanforderungen zum Oberflächentest

Die Qualitätsmerkmale Aufgabenangemessenheit, Erwartungskonformität, Fehlertoleranz, Lernförderlichkeit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Funktionalität und Benutzbarkeit sind in der Checkliste „Oberflächentest“ mit konkreten Prüfpunkten versehen, so dass die Qualitätsanforderungen an den Oberflächentest durch den vorzugebenden Erfüllungsgrad der Checkliste festgelegt werden.

9.2.4 Empfehlungen zum Oberflächentest

Oberflächentest nicht durch Spezialisten

Der Oberflächentest kann von Testern ohne fachliche oder technische Spezialkenntnisse durchgeführt werden. Daher ist es sinnvoll, den Oberflächentest von den komplexeren Tests, die nur von „teuren“ Spezialisten durchgeführt werden können, zu trennen.

kompletter Oberflächentest mit Standard-Browser

Um den Aufwand des Oberflächentests gering zu halten, sollte dieser nur einmal komplett durchgeführt werden, nämlich mit dem Browser-Typ, den die Anwender am häufigsten einsetzen. Für die

anderen Browser-Typen kann der Oberflächentest eingeschränkt durchgeführt werden, mehr dazu im folgenden Kapitel 9.3.

9.3 Browser-Test

Browser von unterschiedlichen Herstellern reagieren nicht identisch. Das Erscheinungsbild und das Verhalten einer Website können sich von Browser-Typ zu Browser-Typ ändern. Daher muss eine web-basierte Anwendung mit verschiedenen Browsern in unterschiedlichen Versionen getestet werden.

Browser-Test

Der Browser-Test überprüft die korrekte Darstellung und Bedienbarkeit einer Web-Anwendung mit unterschiedlichen Browser-Konstellationen.

Es stellt sich die Frage, welcher Browser in welcher Version getestet werden muss. Die Antwort hängt von der Reichweite der Web-Anwendung ab. Ist sie nur einem eingeschränkten Nutzerkreis im Intranet zugänglich, so ist der eingesetzte Browser bekannt oder sogar verbindlich vorgeschrieben. In diesem Fall muss die Anwendung nur für diesen Browser getestet werden. Ist eine Website im Internet frei zugänglich, so muss der Test auf alle gängigen Browser erweitert werden.

9.3.1 Auswahl der zu testenden Browser

Um die Auswahl der Browser, mit denen eine Web-Anwendung getestet werden soll, treffen zu können, empfiehlt sich eine Risikoanalyse. Dazu müssen entsprechende Statistiken vorliegen und Risikoklassen definiert werden.

Statistiken erstellen

Statistiken, die im Internet wie zum Beispiel unter [URL: Browser-Statistik] zu finden sind oder die für die eigene Website mit entsprechenden Werkzeugen erzeugt werden, geben darüber Auskunft, welche Browser aktuell im WWW benutzt werden. Nicht nur der Browser allein kann das Verhalten einer Web-Anwendung beeinflussen, sondern auch die Kombination mit dem Betriebssystem, unter dem er eingesetzt wird. Daher sollten zumindest bei geschäftsrelevanten

Web-Anwendungen die verwendeten Betriebssysteme statistisch erfasst und beim Test berücksichtigt werden.

Risikoklassen definieren

Damit eine risikobasierte Auswahl der Browser getroffen werden kann, müssen Bewertungskriterien, Risikoklassen und die daraus resultierenden Maßnahmen nach den Methoden der Risikoanalyse festgelegt werden (s. Kap. 5).

Browser/Betriebssystem-Risikomatrix erstellen

Anhand der statistischen Zahlen und der definierten Risikoklassen wird eine Browser/Betriebssystem-Risikomatrix erstellt. Darin wird bewertet, wie wichtig der Test einer Browser-Version in Zusammenhang mit dem verwendeten Betriebssystem ist. Aufgrund dieser Bewertung werden die Browser für den Test ausgewählt und die entsprechenden Maßnahmen geplant.

↓

Beispiel zur Browser-Auswahl

Das folgende Beispiel zeigt die Bereitstellung einer Browser/Betriebssystem-Risikomatrix.

Die drei Tabellen 9.1, 9.2 und 9.3 sind der Browser-Statistik einer dem Autor bekannten Website entnommen. Sie zeigen, welche Betriebsysteme, Browser-Typen und Browser-Versionen die Besucher der untersuchten Website eingesetzt haben.

Tabelle 9.1: Betriebssysteme

Betriebssystem	Anteil in %
Windows XP	42,4
Windows 2000	38,8
Windows NT	7,0
Windows 98	3,1
Mac OS	1,8
Windows ME	1,2
Windows 95	0,3
Windows 2003	< 0,1
Linux 2.x	< 0,1
Unbekannt	5,0

Browser-Typ	Anteil in %
Microsoft Internet Explorer	83,6
Netscape	6,3
Mozilla	5,1
Mozilla Firefox[7]	2,4
Opera	1,2
sonstige	unter 1,0

Tabelle 9.2: Browser-Typen

Browser-Version	Anteil in %
Microsoft Internet Explorer 6.x	69,9
Microsoft Internet Explorer 5.x	13,7
Mozilla 1.x	5,1
Netscape 7.x	4,1
Mozilla Firefox 0.x	1,4
Opera 7.x	1,2
Mozilla Firefox 1.x	1,0
Netscape 4.x	1,2
Netscape 3.x	1,0
sonstige	unter 1,0

Tabelle 9.3: Browser-Versionen

In Tabelle 9.4 sind Risikoklassen zu den auftretenden Browser/Betriebssystem-Kombinationen und die daraus abzuleitenden Maßnahmen aufgeführt. Anhand der Risikodefinitionen und der statistischen Zahlen wird mit Tabelle 9.5 eine Browser/Betriebssystem-Risikomatrix[8] für den Browser-Test erstellt.

In der in Tabelle 9.5 abgebildeten Risikomatrix sind nur Werte aus den Statistiken mit mehr als 3% Anteil berücksichtigt. Eine Ausnahme ist der Firefox-Browser, der immer mehr Anhänger findet und daher in die Risikomatrix aufgenommen wird.

Tipp

An dieser Entscheidung – sie wurde im Juli 2005 getroffen – wird deutlich, dass Trends und absehbare technische Veränderungen bei der Testplanung berücksichtigt werden müssen und der Browser-Test regelmäßig wiederholt werden muss. Die Internet-Landschaft verändert sich permanent.

[7] Der geringe Anteil von Firefox-Benutzern lässt sich damit erklären, dass die Statistik für eine geschäftliche Website und im ersten Quartal 2005 erhoben wurde.

[8] Aus Platzgründen ist die Matrix nicht vollständig abgebildet.

Tabelle 9.4: Risikoklassen Browser-Test

Browser/Betriebssystem-Kombination	Risikoklasse	Maßnahmen
Häufig	A	Test der Browser-Einstellungen – Checkliste „Browser-Einstellungen“[9]; Kompletter Oberflächentest – Checkliste „Oberflächentest“[10]
Gelegentlich	B	Test der Browser-Einstellungen – Checkliste „Browser-Einstellungen“; Stichproben aus dem Oberflächentest – Checkliste „Stichprobe Browser-Test“[11]
Selten	C	Sichttest ohne Nachweispflicht

Tabelle 9.5: Browser/Betriebssystem-Risikomatrix

Browser		Betriebssystem			
Browser-Typ	Version	Windows XP	Windows 2000	Windows NT	Windows 98
MS Internet Explorer	6.0	A	A	C	C
	5.5	A	A	C	C
	5.1	B	B	C	C
Mozilla	1.8	A	A	B	B
	1.7.5	C	C	C	C
Firefox	1.0	A	A	B	C
	0.9	B	B	C	C
Netscape	7.2	B	B	C	C
	7.1	B	B	C	C

↑

Der Browser-Test muss regelmäßig während des laufenden Betriebes wiederholt werden, weil sich die Systemlandschaft stetig verändert und somit Regressionstests notwendig werden (s. Kap. 12.2). Für jede Testwiederholung muss die Browser/Betriebssystem-Risikomatrix überarbeitet werden. Dazu ist es notwendig, dass stets aktuelle Statistiken vorliegen, die aufzeigen, mit welchen Browsern und welchen Betriebssystemen die Nutzer auf die Website zugreifen.

[9] S. Kap. 9.3.3
[10] S. Kap. 9.2.1
[11] S. Kap. 9.3.4

9.3.2
Tests mit alten Browser-Versionen

Alte Browser-Versionen sind noch oft im Einsatz und wie die folgenden Abbildungen zeigen, werden zum Beispiel Cascading Style Sheets nicht von allen alten Browser-Versionen richtig interpretiert.

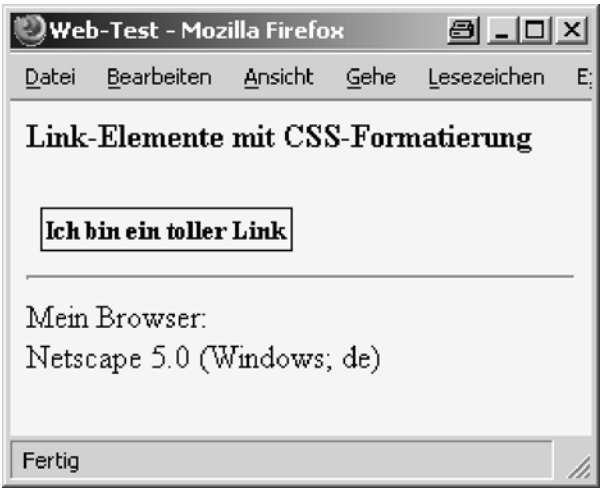


Abb. 9.3: CSS-formatierter Link im neuen Browser

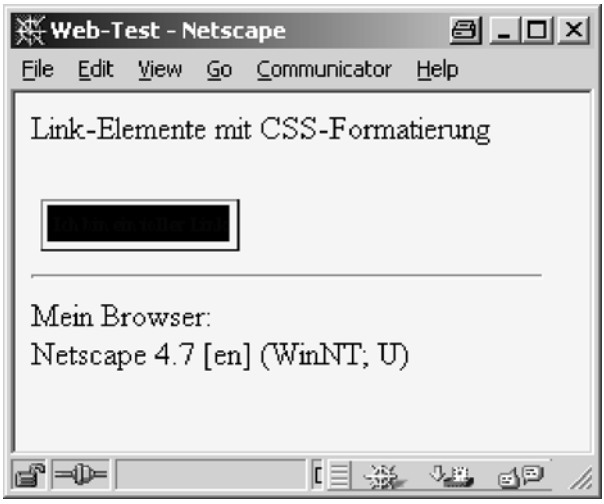


Abb. 9.4: CSS-formatierter Link im alten Browser

Der Link-Text, der durch einen Browser neuerer Version korrekt dargestellt wird (Abb. 9.3), erscheint in einer älteren Browser-Version nur als schwarzer Balken (Abb. 9.4).

Das Beispiel zeigt, dass bei der Testplanung je nach Zielgruppe der Anwendung auch alte Browser-Versionen berücksichtigt und ggf. in der Risikomatrix aufgenommen werden müssen.

9.3.3 Test der Browser-Einstellungen

Unerwünschte Browser-Einstellungen können eine Web-Applikation bis hin zur Funktionsuntüchtigkeit beeinflussen. Daher ist unter Einbeziehung der Risikoklassen einer Browser/Betriebssystem-Risikomatrix (Tabelle 9.5) die Reaktion der Anwendung auf unterschiedliche Browser-Einstellungen zu testen.

Eine Erfahrung

Die folgende persönliche Erfahrung zeigt, wie wichtig Tests zu den Browser-Einstellungen sind, um die Benutzbarkeit und Erreichbarkeit einer Web-Applikation sicherzustellen:

Meine Hausbank wollte – vermeintlich „nur" – den Aufruf des bislang einwandfrei funktionierenden Programms zum Online-Banking ändern. Ab dem angekündigten Umstellungstermin war aber kein Online-Banking mehr möglich. Die Vermutung lag nahe, dass der besagte Termin verschoben worden war, bis sich zufällig herausstellte, dass für die Online-Banking-Funktion PopUp-Windows zugelassen sein mussten. Diese wurden aber aufgrund meiner Browser-Einstellungen für die Website der Bank unterdrückt. Leider hatte die Anwendung für diesen Fall keinen Hinweis für den Benutzer vorgesehen...

Die Testfälle, die für den Browser-Test entworfen werden müssen, werden aus der folgenden Checkliste „Browser-Einstellungen" abgeleitet:

✓ Checkliste Browser-Einstellungen

- Arbeitet und erscheint die Anwendung korrekt, wenn in den Browser-Einstellungen ein Element nicht oder nur beschränkt erlaubt ist?
- Wird dem Benutzer ein entsprechender Hinweis gegeben, wenn ein Element benötigt wird, aber nicht verfügbar ist?
- Wurden die Funktionen und Reaktionen der Anwendung mit den folgenden Einstellungen überprüft?
  - Cookies[GL] sind nicht erlaubt
  - Cookies werden nur von der Original-Website akzeptiert

- Grafiken werden nicht geladen
- Nur Grafiken vom Ursprungs-Server werden geladen
- Animierte Grafiken werden nicht wiederholt
- Animierte Grafiken werden nur einmal wiederholt
- PopUp-Fenster werden blockiert
- ActiveX[GL] Steuerelemente sind deaktiviert
- .NET-Komponenten[GL] sind deaktiviert
- JavaScript ist deaktiviert
- Java ist deaktiviert

9.3.4 Stichprobentest der Oberfläche

Um den Aufwand eines Browser-Tests zu reduzieren, wird der komplette Oberflächentest nur für die am häufigsten eingesetzten Browser-Versionen, wie im vorherigen Abschnitt beschrieben, durchgeführt. Für die seltener installierten Browser-Versionen werden aus der Checkliste zum Oberflächentest die wichtigsten Testfälle ausgewählt und in Stichproben geprüft:

✓Checkliste Stichprobe Browser-Test

Allgemeine Darstellung

- Sind Lesbarkeit und Handhabbarkeit bei Veränderung der Schriftart und Schriftgröße in den Browser-Einstellungen gewährleistet?
- Haben alle gleichen Elemente die selbe Darstellung (Farbe und Größe von Buttons, Menüpunkte, Textformate, Breite von Listboxen,...)?
- Werden alle Elemente im Browser richtig dargestellt?
- Wird die Anwendung bei allen Bildschirmauflösungen richtig dargestellt?
- Werden die verwendeten Cascading Style Sheets richtig interpretiert?

Navigation

- Sind alle Navigationselemente (Buttons, Menüpunkte, Hyperlinks) auffindbar?
- Funktionieren alle Navigationselemente (Buttons, Menüpunkte,) richtig?

- Werden die alternativen Texte für grafische Navigationselemente angezeigt?
- Ist die Navigationsleiste auf jeder Seite sichtbar?
- Erfolgt die Cursor-Steuerung mit <TAB> und <SHIFT+TAB> in der richtigen Reihenfolge?

Druck

- Arbeitet die Druckfunktion des Browsers korrekt?
- Arbeiten die Druckfunktionen innerhalb der Anwendung korrekt?

9.3.5 Werkzeuge für den Browser-Test

Browser-Archiv

Wer für seine Tests alte Browser-Versionen sucht, wird unter [URL: Browser-Archiv] fündig.

User Tracking Tools

Im Internet sind zahlreiche Anbieter von User Tracking Tools zu finden. Diese Werkzeuge liefern neben Auswertungen zum Benutzerverhalten viele Statistiken, wie zum Beispiel von den Besuchern einer Website eingesetzte Browser-Versionen, Betriebssysteme und Plugins. Die im Abschnitt 9.3.1 aufgeführte Browser-Statistik wurde mit einem User Tracking Tool erzeugt.

Capture Replay Tools

Der Browser-Test muss bei Erscheinen neuer Versionen von Browsern und Betriebssystemen wiederholt werden, um sicherzustellen, dass die Web-Anwendung benutzbar bleibt. Daher lohnt es sich, den Browser-Test mit einem Capture Replay Tool (s. Abschnitt 12.4.2) zu automatisieren. Mit „guten“ Tools können Tests mit einem Browser-Typ aufgezeichnet und mit einem anderen Browser-Typ wieder abgespielt werden.

Selenium

Zum Nachweis der Browser-Kompatibilität einer Web-Applikation ist zum Beispiel das Open Source Tool Selenium ([URL: Selenium]) gut geeignet, weil die Testskripte unter mehreren Betriebssystemen und Browsern eingesetzt werden können.[12]

[12] Selenium siehe auch Abschnitt 13.4.2

9.3.6 Qualitätsanforderungen zum Browser-Test

Die Qualitätsanforderungen zur Benutzbarkeit und Erreichbarkeit leiten sich für den Browser-Test aus der erstellten Risikomatrix und den vorgegebenen Checklisten ab, wie zum Beispiel:

- Alle in der Risikomatrix mit Priorität „A“ bewerteten Einträge sind bezogen auf die Checklisten „Browser-Einstellungen“ und „Oberflächentest“ zu 100% fehlerfrei getestet.
- Alle in der Risikomatrix mit Priorität „B“ bewerteten Einträge sind bezogen auf die Checklisten „Browser-Einstellungen“ und „Browser-Stichprobentest“ zu 100% fehlerfrei getestet.
- Alle in der Risikomatrix mit Priorität „C“ bewerteten Einträge sind in Stichproben aus der Checkliste „Browser-Stichprobentest“ geprüft, d.h. es wurden mindestens 50% zufällig ausgewählte Fragen geprüft und alle davon positiv bewertet.

9.3.7 Empfehlungen zum Browser-Test

fremdsprachige Browser-Versionen testen

In den Anforderungen an die Web-Anwendung muss festgelegt werden, welche fremdsprachigen Browser-Versionen für den Web-Auftritt relevant sind. Sie dürfen beim Browser-Test nicht vergessen werden.

Browser-Statistiken beauftragen

Die Bereitstellung von Browser-Statistiken für eine Website kann im Internet beauftragt werden. Anbieter sind zum Beispiel unter dem Suchbegriff „Browseranalyse“ zu finden (siehe auch User Tracking Tools im Abschnitt 9.4.6).

Release Notes lesen

Eine Web-Anwendung muss nicht unbedingt für jedes Release und für jede Version eines Browsers erneut getestet werden. Der Testdesigner sollte sich nach den Unterschieden erkundigen und die Release Notes lesen. Erst dann entscheidet er die notwendigen Testmaßnahmen. Das gleiche gilt für neue Versionen von Betriebssystemen.

Technologien abwägen

Das Risiko, dass Darstellungen oder Funktionen einer Web-Anwendung empfindlich gegenüber Browsern sind, hängt auch von den eingesetzten Web-Techniken ab. Ein geringes Risiko besteht zum Beispiel bei Verwendung von statischen HTML-Seiten und JavaScripting, ein hohes beim Einsatz von Java-Applets oder von ActiveX[GL]. Der Einsatz von bestimmten Technologien sollte daher mit

Blick auf mögliche Risiken und den entstehenden Testaufwand genau überlegt werden.

Browser-Test automatisieren

Der Browser-Test sollte in der Teststufe Komponententest für den meist eingesetzten Browser durchgeführt werden. Für alle weiteren Browser-Versionen muss er mit der Teststufe Systemtest beendet werden. Beim Erscheinen neuer Browser- und Betriebssystem-Versionen muss der Browser-Test während der Betriebsphase wiederholt werden, was für eine Automatisierung der Tests spricht.

9.4 Usability-Test

Wenn eine Web-Anwendung der gesamten Web-Gemeinde zur Verfügung gestellt wird, sind die Nutzer unbekannt. Sie sprechen verschiedene Sprachen, haben keine Schulung für die Anwendung der Software erhalten, haben kein Handbuch gelesen, treffen mit unterschiedlichen Intentionen auf die Website. Trotzdem sollen sich alle Benutzer in der Anwendung zurechtfinden und sie optimal für ihre Zwecke nutzen können.

Usability-Test

Der Usability-Test überprüft in Abhängigkeit der zu erreichenden Zielgruppen die Gebrauchstauglichkeit (Usability)[13] einer Web-Applikation.

Das Problem beim Nachweis der Gebrauchstauglichkeit eines Software-Systems ist, dass die Bewertung einer Anwendung subjektiv ist. Als Konsequenz daraus sollten Repräsentanten aus unterschiedlichen Nutzergruppen die Anwendung in einem Usability-Labor ausprobieren und anschließend eine Bewertung abgeben.

Als Ergänzung oder Alternative zum Usability-Labor können Befragungen der einzelnen Nutzergruppen mit Fragebögen durchgeführt werden.

9.4.1 Usability-Labor

Nutzergruppen identifizieren

Für den Usability-Test müssen die Nutzergruppen der zu testenden Anwendung identifiziert und beschrieben werden. Das können zum Beispiel Gelegenheits-Surfer, Interessenten oder Kunden sein.

---

[13] s. Kap. 2.1, Normen und Qualitätsmerkmale

- Kunden wollen Geschäfte über die Web-Applikation abwickeln. Sie wollen ihre Transaktionen[GL] reibungslos, sicher und fehlerfrei durchführen.
- Interessenten steuern, eventuell über eine Suchmaschine, die Website gezielt an, um für sie relevante Informationen zu erhalten.
- Gelegenheits-Surfer stoßen mehr oder weniger zufällig auf die Website. Hier entscheiden die ersten Sekunden, ob der Besucher von ihren Inhalten gefesselt wird.

Aufgaben festlegen

Für jede Nutzergruppe werden anschließend konkrete Aufgaben festgelegt, die später mit der Web-Anwendung gelöst werden sollen, wie zum Beispiel drei Aufgaben zur Autofinanzierung:

Aufgabe 1:
Die Testpersonen der Zielgruppe Kunden sollen für ihr Wunschauto eine Finanzierungsberechnung durchführen und das Ergebnis ausdrucken.

Aufgabe 2:
Interessenten sollen herausfinden, welche unterschiedlichen Finanzierungsmöglichkeiten angeboten werden.

Aufgabe 3:
Die einfache Aufgabe für Informationssuchende ist, allgemeine Informationsunterlagen für eine Autofinanzierung anzufordern, die auf dem Postweg zugesendet werden.

Benutzerverhalten im Usability-Labor beobachten

Zu jeder Nutzergruppe werden mehrere neutrale Personen ausgewählt und aufgefordert, die gestellten Aufgaben in einem Usability-Labor zu lösen. Während der Sitzung werden die Testpersonen beobachtet und ihre Aktionen werden eventuell mit einer Kamera aufgezeichnet. Klickpfade und Kennzahlen, wie benötigte Zeit pro Aufgabe, Anzahl der Klicks pro Aufgabe oder Verweildauer auf einem Dokument, werden protokolliert. Die Kommentare der Testnutzer werden ebenfalls festgehalten.

9.4.2 Befragung

Fragebogen und/oder Interview

Nach der Sitzung im Usability-Labor werden die Testpersonen nach Ihren persönlichen Erfahrungen und Eindrücken befragt. Die Befragung kann alternativ oder ergänzend durch einen Fragebogen und in einem Interview erfolgen. Sie besteht aus zwei Teilen.

Zuerst werden die Vorkenntnisse und das allgemeine Nutzungsverhalten erfragt, um die Benutzer in ihren Bewertungen einordnen zu können. Beispiele für allgemeine Fragen zum Nutzungsverhalten sind in Tabelle A.1 im Anhang A aufgeführt.

Bewertung durch Benutzer

Im zweiten Teil einer Befragung füllen die Testpersonen einen Bewertungsbogen zur Benutzerführung, zur Struktur, zum Design und zum Inhalt der Web-Anwendung aus. Ein Beispiel eines Bewertungsbogens, in dem die Qualitätsmerkmale der ISO 9241-10[14] im Fokus stehen, ist in Tabelle A.2 im Anhang A abgebildet.

Die Testpersonen füllen den Bewertungsbogen auf Basis ihrer persönlichen Erfahrungen aus, die sie beim Lösen der gestellten Aufgaben mit der Anwendung gemacht haben. Das Bewertungsmaß ist auf einer Bewertungsskala mit einer geraden Anzahl von Antwortmöglichkeiten, die eine positive oder negative Entscheidung zu jeder Frage erzwingt, vorgegeben.

Verbesserungsvorschläge

Am Ende eines Fragebogens oder Interviews werden noch Verbesserungsvorschläge von den Testpersonen eingefordert.

9.4.3 Auswertung Usability-Labor und Befragungen

Die Auswertungen der Aufzeichnungen aus dem Usability-Labor, der Fragebögen und der Interviews geben Aufschluss über die Stärken und Schwächen der Web-Anwendung, indem folgende Fragen beantwortet werden:

Allgemeine Aussagen:

- Wie viel Prozent der Teilnehmer haben Ihre Aufgabe erfüllt?
- Wie viel Prozent der Teilnehmer haben aufgegeben?
- Wie oft wurde die Hilfefunktion aufgerufen?
- Warum wurde die Hilfefunktion bei Problemen nicht aufgerufen?

---

[14] s. Kap. 2.1

- Haben die Personen zur Lösung der Aufgabe immer den kürzesten Pfad durch die Anwendung genommen?
- Wie lange haben die Personen für die Erledigung ihrer Aufgabe gebraucht?
- An welchen Dokumenten und Funktionen haben sich die Personen relativ lange aufgehalten?

Aufgabenspezifische Aussagen[15]:

Zur 1. Aufgabe:

- Waren die Eingaben für eine Finanzierungsberechnung auf Anhieb fehlerfrei?
- Konnten alle Personen die Finanzierungsberechnung drucken?

Zur 2. Aufgabe:

- Konnten die Testpersonen alle Finanzierungsvarianten aufzählen?
- Konnten die Testpersonen alle Finanzierungsvarianten erklären?

Zur 3. Aufgabe:

- Wurde die Funktion zur Anforderung der Finanzierungsunterlagen für den Postweg ohne Umwege gefunden?
- Waren die Eingaben zur Anforderung der Finanzierungsunterlagen für den Postweg fehlerfrei?

Bewertung von Effektivität und Effizienz

Wenn die Mehrzahl der Testpersonen die ihnen gestellte Aufgabe nicht, mit Problemen oder nur in unverhältnismäßig langer Zeit lösen konnte, müssen Effektivität und Effizienz der Gebrauchstauglichkeit verbessert werden.

Bewertung der Zufriedenstellung

Der Grad der Zufriedenstellung der Benutzer ergibt sich aus den Antworten des ausgefüllten Fragebogens und der geführten Interviews. Haben sich die Benutzer darin zu den gleichen Themen negativ geäußert oder gleichartige Verbesserungsvorschläge gemacht, besteht Verbesserungsbedarf.

[15] Es handelt sich in diesem Beispiel um die im Abschnitt 9.4.1 gestellten Aufgaben.

9.4.4 Online-Umfrage

Online-Umfragen

Wenn die Einrichtung eines Usability-Labors zu aufwendig oder zu teuer ist, bietet sich eine Pilotphase (Abschnitt 14.6.1) an, in der dem Benutzer Online-Umfragen zur Verfügung gestellt werden. Anreize für Benutzer, einen Fragebogen online auszufüllen, können auf der Website zum Beispiel durch besondere Angebote oder ein Preisausschreiben gegeben werden.

Der Online-Fragebogen enthält Fragen zum allgemeinen Nutzungsverhalten und zur Bewertung der Gebrauchstauglichkeit, wie sie in den Tabellen A.1 und A.2 im Anhang A aufgeführt sind.

9.4.5 Blickregistrierung

Eye Tracking

Ein ergänzendendes Verfahren zum Usability-Test ist die Blickregistrierung (Eye Tracking), bei der die Blickbewegungen einer Testperson aufgezeichnet und ausgewertet werden.

Die Blickregistrierung zur Analyse des Benutzerverhaltens auf einer Website zeigt zum Beispiel auf,

- was Benutzer auf einer Bildschirmseite wahrnehmen,
- wie oft sie welche Bereiche im Blick erfassen,
- wie lange und intensiv sie ein einzelnes Element (Seite, Menüpunkt, Überschrift, Grafik, Text,...) betrachten,
- ob sie bestimmte Elemente besonders häufig oder gar nicht wahrnehmen.

Zur Blickregistrierung werden spezielle Werkzeuge, sogenannte Eye Tracker benötigt. Sie werden im folgenden Abschnitt beschrieben.

9.4.6 Werkzeuge für den Usability-Test

User Tracking Tools

Das Benutzerverhalten der Testpersonen wird mittels User Tracking Tools protokolliert, um zum Beispiel zu folgenden Fragen Auskunft zu bekommen:

- An welchen Tagen und zu welcher Uhrzeit waren wie viele Besucher auf der Website?

- Welche Seiten wurden am häufigsten und mit welcher durchschnittlichen Dauer besucht?
- Wann hat wer wie lange mit welcher IP-Adresse über welchen Provider die Website besucht?
- Wie lange war ein Benutzer auf welcher Seite?
- Welchen Klickpfad hat der Benutzer genommen? D.h., auf welcher Seite ist der Benutzer eingestiegen, welchen Pfad hat er durch die Anwendung genommen und auf welcher Seite ist er wieder ausgestiegen?
- Mit welchen Suchwörtern gelangen Benutzer über Suchmaschinen auf die Website?

Diese Aufzählung ist nur eine Auswahl der Auswertungsmöglichkeiten, die User Tracking Tools bieten. User Tracking Tools werden nicht nur im Usability-Labor im Rahmen des Usability-Tests eingesetzt, sondern auch zur Erhebung von Statistiken während des laufenden Betriebs.

phpOpenTracker

Ein Beispiel für ein Werkzeug zum Erstellen von Web-Statistiken ist phpOpenTracker. phpOpenTracker ist Open Source Software unter Apache License zur Messung des Datenverkehrs und zur Analyse des Benutzerverhaltens von Web-Applikationen ([URL: phpOpenTracker]).

Eye Tracker

Für die Blickregistrierung werden Eye Tracker benötigt, deren Einsatz aufgrund der technischen Anforderungen sehr kostspielig ist. Sie sind im Web unter den Stichworten „Eye Tracker“ oder „Blickregistrierung“ zu finden. Eye Tracker messen zum Beispiel:

- Anzahl der Fixationen zur Durchführung einer bestimmten Aufgabe in einem bestimmten Zeitintervall
- Anzahl der Fixationen eines Objektes auf der Web-Seite im Verhältnis zur Anzahl aller Fixationen auf der Web-Seite in einem bestimmten Zeitintervall in Prozent
- Anzahl der Sakkaden zur Durchführung einer bestimmten Aufgabe in einem bestimmten Zeitintervall. Eine große Anzahl der Sakkaden kann bedeuten, dass der Benutzer viele Informationen auf der Web-Seite „zusammensuchen“ muss
- Länge des Blickpfades (Summe der Abstände zwischen den Orten der einzelnen Fixationen) zur Durchführung einer bestimmten Aufgabe

Fixation und Sakkade

Bleiben noch die Begriffe Fixation und Sakkade zu definieren (Zitat aus [URL: wiki-Blick]):

„Fixationen und Sakkaden machen den größten Teil der bewussten Augenbewegungen aus. Während einer Fixation nimmt das Auge über die Netzhaut Informationen aus der Umgebung auf und leitet diese nach einer Vorverarbeitung an das Gehirn weiter. Während einer Sakkade hingegen nimmt das Auge keine visuellen Informationen auf. Man ist in dieser Phase tatsächlich blind und sieht darin eine der Mitursachen der Unaufmerksamkeitsblindheit, also der Unempfänglichkeit für visuelle Reize durch mangelnde Aufmerksamkeit."

9.4.7 Qualitätsanforderungen zum Usability-Test

Qualitätsanforderungen zu den Qualitätsmerkmalen Gebrauchstauglichkeit und Benutzerfreundlichkeit müssen nicht nur für die Auswertungen im Usability-Labor festgelegt werden, sondern auch für die Zeit nach der Freigabe der Anwendung. Das setzt voraus, dass das Benutzerverhalten auch nach der Einführung der Software aufgezeichnet und ausgewertet wird. Qualitätsanforderungen zum Usability-Test können wie folgt definiert werden:

Qualitätsanforderungen an das Usability-Labor:

- Die Testpersonen haben zu 90% die ihnen gestellte Aufgabe in der vorgegebenen Zeit gelöst.
- 75% der Testpersonen haben sich positiv geäußert.
- Jede Frage aus dem Bewertungsfragebogen wurde von mindestens 65% der Befragten positiv („++" oder „+") bewertet.

Qualitätsanforderungen an den laufenden Betrieb:

- 30% der Benutzer verweilen durchschnittlich mindestens 2 Minuten auf der Website.
- Maximal 3% der Benutzer, die länger als 5 Minuten auf der Website verweilen, stellen Supportanfragen.
- Die Anzahl der Anforderungen von Informationsmaterial steigt im 1. Jahr um 20% pro Monat.
- Die Anzahl der Vertragsabschlüsse steigt im 1. Jahr um 10% pro Monat.

9.4.8 Empfehlungen zum Usability-Test

neutrale Testpersonen einsetzen

Der Usability-Test sollte in der Phase des Systemtests vor Freigabe der Anwendung im Usability-Labor durchgeführt werden. Die Testpersonen sollten repräsentativ für jede Zielgruppe ausgewählt und nicht aus dem Projektumfeld rekrutiert werden, damit die Testergebnisse authentisch sind.

Fachleuten einbeziehen

Die Planung, Durchführung und Auswertung des Usability-Tests sollte von Fachleuten aus dem Marketingbereich vorgenommen werden. Wenn der geschäftliche Erfolg des Unternehmens von der Web-Anwendung abhängt, ist es ratsam, den Usability-Test vom externen Spezialisten durchführen zu lassen.

Usability-Test als permanente Maßnahme

Die Auswertungen des Benutzerverhaltens sollte nicht nur während der Tests im Usability-Labor vorgenommen werden, sondern auch für die im Web freigeschaltete Anwendung. So kann rechtzeitig eine Veränderung im Benutzerverhalten erkannt und mit Verbesserungen des Web-Angebotes reagiert werden. Der Usability-Test ist eine permanente Qualitätssicherungsmaßnahme im laufenden Betrieb.

Informationen

Interessante Informationen und weiterführende Links zur Usability sind beim „Informationsdienst Arbeit und Gesundheit“ zu finden ([URL: Ergo]).

9.5 Zugänglichkeitstest

Nicht jeder Benutzer kann die PC-Arbeitsmittel wie Tastatur oder Maus einsetzen. Einige können nur schwer oder gar nicht sehen oder hören, einige benutzen einen Sprach-Browser. Deswegen muss sichergestellt werden, dass die Inhalte einer Website für behinderte Benutzer zugänglich sind.

Das W3C (World Wide Web Consortium, [URL: W3C]) setzt die Standards für die Zugänglichkeitsrichtlinien, die insbesondere im öffentlichen Bereich unter dem Begriff „barrierefreies Web“ gefordert werden, fest.

Eine Web-Anwendung ist barrierefrei[16], wenn sie für behinderte Menschen in der allgemein üblichen Weise, ohne besondere Erschwernis und grundsätzlich ohne fremde Hilfe zugänglich und nutzbar ist.

[16] Barrierefreiheit s. Kap. 2.2

Zugänglichkeitstest

Der Zugänglichkeitstest stellt den barrierefreien Zugriff auf die Inhalte eines Web-Angebotes sicher.

9.5.1 Zugänglichkeitsanforderungen

✓ Checkliste Zugänglichkeit des WAI (extern)

Die Web Accessibility Initiative (WAI) des W3C beschreibt die Zugänglichkeitsanforderungen in Form einer Checkliste, die aktuell unter [URL: WAI-CL] abrufbar ist. Daher werden die Inhalte an dieser Stelle nicht aufgeführt, sondern es wird auf die deutsche Übersetzung von René Hartmann [URL: WAI-CLdt] verwiesen. Die Prüffragen zum Zugänglichkeitstest sind von der WAI priorisiert worden:

muss

- Priorität 1: Der Checkpunkt muss erfüllt sein, damit bestimmte Gruppen die Web-Dokumente benutzen können. Zum Beispiel muss Bildschirmflackern vermieden werden, um beim Benutzer keine epileptischen Anfälle auszulösen.

sollte

- Priorität 2: Der Checkpunkt sollte erfüllt sein, damit alle Benutzergruppen ohne Probleme auf die Informationen zugreifen können. Zum Beispiel muss für Benutzer, die eine Website schwarz-weiß betrachten oder eine Rot-Grün-Schwäche haben, der Kontrast von Vorder- zu Hintergrund ausreichend sein.

kann

- Priorität 3: Der Checkpunkt kann erfüllt sein, damit der Zugriff auf die Web-Dokumente möglichst einfach ist. Zum Beispiel erleichtern Shortcuts (Tastaturkurzbefehle) für alle Funktionen und wichtige Links die Navigation ohne Maus.

✓ Checkliste Zugänglichkeit des BITV (extern)

In Deutschland legt die Barrierefreie Informationstechnik-Verordnung (BITV) die Zugänglichkeitsrichtlinien für Web-Inhalte fest ([URL: BITV]). In der Anlage zur BITV werden 14 Anforderungen in zwei Checklisten festgelegt, die sich an den Richtlinien der WAI orientierenden. Für jede vom BITV festgelegten Prioritätenstufe ist eine Checkliste vorgegeben[17]:

- Priorität I: für alle Angebote, die neu gestaltet oder in wesentlichen Bestandteilen oder größerem Umfang verändert oder angepasst werden

[17] Die beiden Checklisten des BITV ([URL: BITVCL]) sind im Anhang B im Originaltext abgebildet.

- Priorität II: zusätzlich für alle zentralen Navigations- und Einstiegsangebote

Beispiel zur Barrierefreiheit

Ein Beispiel aus der Checkliste der Priorität I ist die Vermeidung von Bildschirmflackern:

- Anforderung 7: „Zeitgesteuerte Änderungen des Inhalts müssen durch die Nutzerin, den Nutzer kontrollierbar sein."
- Bedingung 7.1: „Bildschirmflackern ist zu vermeiden."

Das Beispiel in Abb. 9.5 zeigt das negative Prüfergebnis einer Web-Seite zu diesem Prüfpunkt.

9.5.2 Werkzeuge für den Zugänglichkeitstest

Werkzeuge für Zugänglichkeitsprüfungen sind zum Beispiel bei der WAI unter [URL: WAI-Tools] aufgeführt. Einige der in der Regel kostenlosen Tools werden im Folgenden vorgestellt.

HTML-Kit

Wie einige andere Editoren auch, bietet zum Beispiel der kostenlose Editor „HTML-Kit" ([URL: HTMLkit]) eine automatische Prüfung auf die Web-Zugänglichkeitsrichtlinien an. Als Ergebnis einer Prüfung sind die Nummern aus der Checkliste der Zugänglichkeitsrichtlinien bewertet, wie zum Beispiel in der Abb. 9.5 der Prüfpunkt 7.1, der die Vermeidung von Bildschirmflackern fordert.

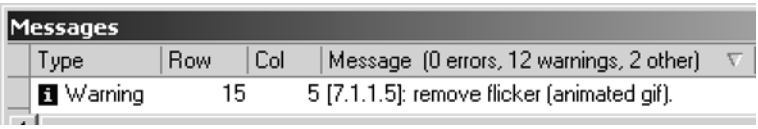


Abb. 9.5: HTML-Kit Prüfprotokoll

Lynx

Zur visuellen Überprüfung der Zugänglichkeit einer Website empfiehlt sich der textbasierte Browser Lynx ([URL: Lynx]). Mit Lynx wird schnell sichtbar, ob komplizierte Navigationen oder grafische Spielereien die Website für Behinderte unlesbar machen. In dem Beispiel in Abb. 9.6 wird deutlich, dass in der aufgerufenen Web-Seite der Umlaut „ä" (im Wort „nächsten") nicht richtig angezeigt wird und dass kein alternativer Text für die Grafik angegeben ist (in der eckigen Klammer erscheint nur der Name der gif-Datei), was nach der Checkliste des WAI ein Fehler der Priorität 1 ist.

Abb. 9.6: Lynx-Browser

WAVE

Einen Online-Dienst zur Überprüfung einer Website bietet das WAVE Accessibility Tool des Temple University Institute on Disabilities ([URL: WAVE]) an. WAVE stellt die zu prüfenden Web-Seiten mit erläuternden Symbolen dar, die je nach Schwere des Verstoßes gegen die Zugänglichkeitsrichtlinien farbig markiert sind. Interessanter Weise werden Umlaute von WAVE nicht moniert – oder erklärlicher Weise, weil es kein deutschsprachiges Produkt ist? Zumindest macht dieses Beispiel deutlich, dass Werkzeuge die visuelle Überprüfung einer Website nicht vollständig ersetzen können.

AbI-Tools

Weitere nützliche Tools werden vom Aktionsbündnis für barrierefreie Informationstechnik (AbI) empfohlen, wie die Web Accessibility Toolbar, der Barrierenprüfer A-Prompt und der Farbkontrast-Analyzer ([URL: AbITools]).

Web Accessibility Toolbar

Mit der im Abschnitt 9.2.2 erwähnten Web Accessibility Toolbar können Zugänglichkeitsprüfungen direkt oder mit aufrufbaren Drittwerkzeugen durchgeführt werden. Allerdings ist dieses Werkzeug nur für den MS Internet Explorer verfügbar.

A-Prompt

A-Prompt dient unter Berücksichtigung der Zugänglichkeitsrichtlinien der Überprüfung und Korrektur von HTML-Dokumenten.

Abb. 9.7 zeigt ein Anwendungsbeispiel mit dem Farbkontrast-Analyzer. Darin ist zu sehen, dass die untersuchte Web-Seite nicht den Anforderungen der Barrierefreiheit genügt, weil die Farbdifferenz der ausgewählten Farben nicht hinreichend ist.

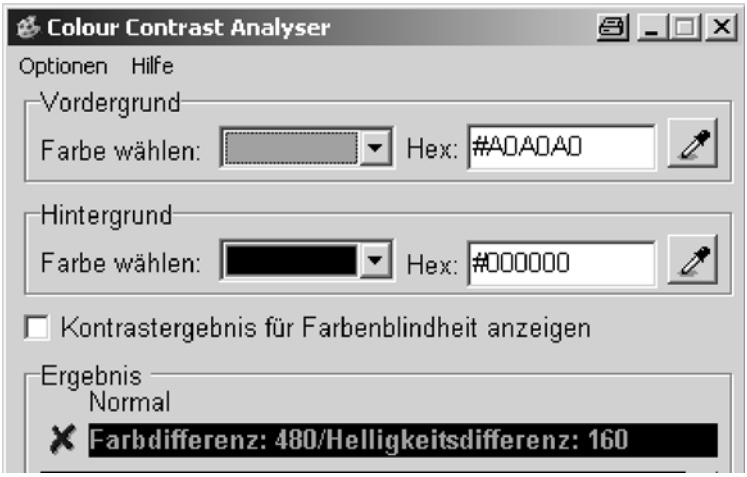


Abb. 9.7: Farbkontrast-Analyzer

9.5.3 Qualitätsanforderungen zum Zugänglichkeitstest

Der Zugänglichkeitstest prüft das Qualitätsmerkmal Barrierefreiheit (Accessibility). Die konkreten Qualitätsanforderungen werden aus den Prioritäten der oben vorgestellten Checklisten abgeleitet. Wird die Checkliste des WAI zu Grunde gelegt, lassen sich zum Beispiel folgende Qualitätsanforderungen festlegen:

- 100% Erfüllungsgrad der Punkte, die in der Checkliste „Zugänglichkeit des WAI“ mit Priorität 1 und 2 bewertet sind.
- 75% Erfüllungsgrad der Punkte, die in der Checkliste „Zugänglichkeit des WAI“ mit Priorität 3 bewertet sind.

9.5.4 Empfehlungen zum Zugänglichkeitstest

Der Zugänglichkeitstest kann von Testern ohne Spezialkenntnisse durchgeführt werden. Es bietet sich an, die Tests von den Entwicklern während der Programmierung mit Werkzeugen durchführen und protokollieren zu lassen. Die Qualitätssicherung überprüft dann vor der Produktfreigabe die Testprotokolle und führt zusätzlich die visuelle Prüfung mit einem geeigneten Browser durch, zum Beispiel mit Lynx.

9.6 Auffindbarkeitstest

Weil das beste Web-Angebot nichts nutzt, wenn es von niemandem gefunden wird, ist der Auffindbarkeitstest sehr wichtig.

Auffindbarkeit

Auffindbarkeit ist die Eigenschaft einer Website, von der Zielgruppe im Internet einfach gefunden werden zu können. Das bedeutet für eine Website, dass die Web-Adresse (URL = Uniform Ressource Locator) einen sprechenden, leicht zu merkenden Namen hat, und dass Suchmaschinen die Seiten einer Website für bestimmte Stichworte unter den ersten Treffern anzeigen.

Auffindbarkeitstest

Der Auffindbarkeitstest prüft, ob eine Website durch direkte Eingabe einer URL im Browser oder durch Eingabe themenbezogener Stichworte in Suchmaschinen problemlos gefunden wird.

9.6.1 Namensgebung der Web-Adresse

Die für die Namensgebung einer Web-Adresse zu beachtenden Regeln können in einer kurzen Checkliste festgehalten werden.

✓ Checkliste Web-Adresse

- Ist die Web-Adresse sprechend und möglichst kurz?
- Ist die Web-Adresse einprägsam?
- Werden nicht gängige Abkürzungen vermieden?
- Sind alternative Web-Adressen – zum Beispiel mit einer gängigen Abkürzung – belegt, die auf die eigentliche URL verweisen?
- Sind ähnlich klingende URLs belegt, die zum Beispiel bei Schreibfehlern auf die eigentliche URL verweisen?
- Hat die Web-Adresse einen sinnvollen Bezug zum Unternehmen oder angebotenen Dienstleistung?

Diese Fragen lassen sich recht leicht überprüfen, wogegen die Optimierung einer Website für Suchmaschinen eine Wissenschaft für sich ist.

9.6.2 Suchmaschinenoptimierung

SEO

Die Methoden der Suchmaschinenoptimierung (SEO = Search Engine Optimization) sorgen dafür, dass Web-Seiten für bestimmte Suchbegriffe in der Ergebnisliste von Suchmaschinen möglichst weit oben stehen. Bei der Programmierung einer Website muss darauf geachtet werden, dass sie von den Instrumenten der Suchmaschinen vollständig erfasst und optimal ausgewertet werden kann.

Sollen die Web-Seiten für bestimmte Stichworte von Suchmaschinen mit hoher Priorität versehen und in der Trefferliste möglichst weit oben angezeigt werden, müssen die Inhalte der einzelnen Seiten für diese Stichworte optimiert werden.

Ranking

Für das Ranking der Suchmaschinen, d.h. die Sortierung der Treffer nach Relevanz, spielen sowohl die Inhalte als auch die Aktualität des Web-Auftritts eine Rolle. Daher muss das Informationsangebot einer Website nach ihrer Freigabe regelmäßig überarbeitet und erweitert werden.

Die Überprüfung einer Website unter dem Gesichtspunkt der Suchmaschinenoptimierung besteht aus zwei Phasen:

SEO Code-Inspektion

1. Phase: Parallel zur Entwicklung der Web-Seiten prüfen Experten, ob das Design, der Programmcode und die Inhalte eines Web-Auftrittes für Suchmaschinen hinreichend optimiert sind. Dazu werden Checklisten und/oder Reviews eingesetzt. Zuvor müssen Checklisten zur Code-Inspektion um die Anforderungen zur Suchmaschinenoptimierung ergänzt werden.

Ranking-Überprüfung

2. Phase: Im späteren Betrieb wird das Ranking der Web-Seiten regelmäßig überprüft. Das ist notwendig, weil das Ranking von den Suchmaschinen permanent neu berechnet wird und ggf. Maßnahmen ergriffen werden müssen, wenn sich das Ranking einer Web-Seite verschlechtert.

Informationen

Die einzuhaltenden Standards und Empfehlungen zur Suchmaschinenoptimierung sind auf den entsprechenden Seiten der Suchmaschinen wie zum Beispiel von Google oder Microsoft zu finden ([URL: SEO-Google], [URL: SEO-MSN]). Sie sollten in die entsprechenden Checklisten zur Inspektion einer Website einfließen.

9.6.3 Werkzeuge für den Auffindbarkeitstest

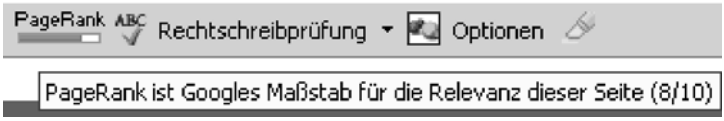
Trefferlisten

Die Position eines Suchergebnisses in einer Trefferliste lässt sich direkt durch die Eingabe des Stichwortes in einer Suchmaschine überprüfen.

PageRank

PageRank von Google zeigt in der Google Toolbar die Beliebtheit einer Web-Seite bei Google an (Abb. 9.8). Die Grundlage für das Maß bilden die von anderen Web-Seiten hereinführenden Links auf diese Web-Seite (Inbound Links).

Abb. 9.8: PageRank



Ranking Tools

Es gibt eine Vielzahl von in der Regel kostenpflichtigen SEO-Tools, die im Web unter den entsprechenden Stichworten wie „Page Ranking Tool", „Ranking Report" oder „SEO-Tool" mit Suchmaschinen gefunden werden können.

9.6.4 Qualitätsanforderungen zum Auffindbarkeitstest

Die Einhaltung der Richtlinien zur Namensgebung von Web-Adressen kann einfach und ohne Werkzeugunterstützung anhand der vorgegebenen Checkliste nachgewiesen werden.

Das Ranking der Web-Seiten und die gewünschte Position innerhalb der Trefferliste zu bestimmten Suchwörtern kann in Zahlen gemessen werden. Dazu müssen die Qualitätsanforderungen konkret vorgegeben sein, wie in den folgenden Beispielen.

Qualitätsanforderungen zum Ranking

Das Ranking einer Web-Seite muss nach ihrer Freischaltung bei der Suchmaschine „Xyz" folgende Werte auf der 10-wertigen Ranking-Skala erreichen:

- ab der sechsten Woche den Ranking-Wert 3
- ab der Woche 10 den Wert 5
- ab der Woche 14 und folgenden mindestens den Wert 6

Qualitätsanforderungen zur Suchmaschinenpositionierung
Eine ehrgeizige Qualitätsanforderung zur Positionierung einer Website in den Trefferlisten von Suchmaschinen wäre folgende:

Zum Stichwort „Webtesting“ rangiert die Web-Seite zu diesem Buch im Web-Auftritt des Verlages vier Wochen nach ihrer Freischaltung und die 24 darauf folgenden Monate bei den beiden Suchmaschinen Xxx und Yyy unter den Top 20.

9.6.5 Empfehlungen zum Auffindbarkeitstest

SEO durch Experten

Der Auffindbarkeitstest ist als permanente Aktivität über die gesamte Lebensdauer einer Website durchzuführen, sofern der Auftraggeber die Qualitätsanforderungen zur Auffindbarkeit konkret definiert hat. Weil zur Suchmaschinenoptimierung besonderes Fachwissen über Webdesign, Marketing und die Algorithmen der Suchmaschinen benötigt wird, ist es ratsam, diese Dienstleistung – zumindest in der Startphase eines Web-Angebotes – von Experten einzukaufen.

Suchmaschinen mögen optimale Zugänglichkeit

Mit der Erfüllung der Kriterien zur Zugänglichkeit wird auch die Auffindbarkeit verbessert, weil Suchmaschinen eine Website so „sehen“, wie sie sich im Lynx-Browser darstellt. Das heißt, dass sich die Barrierefreiheit eines Web-Auftritts positiv auf seine Auffindbarkeit auswirkt.

Websites anmelden

Um eine Website bekannt zu machen, sollte sie natürlich bei den wichtigsten Suchmaschinen angemeldet werden. Die Anmeldung kann direkt manuell bei den Anbietern der Suchmaschinen oder automatisiert durch Programme, die von SEO-Dienstleistern angeboten werden, erfolgen. Eine regelmäßige Wiederanmeldung verbessert die Auffindbarkeit und das Ranking der Web-Seiten.

9.7 Zusammenfassung

- Ziel der Benutzbarkeitstest ist die Sicherstellung der Qualitätsmerkmale Bedienbarkeit, Erlernbarkeit, Verständlichkeit sowie der web-spezifischen Qualitätsmerkmale Auffindbarkeit, Barrierefreiheit und Rechtskonformität einer Web-Applikation. Ziel ist nicht die Überprüfung der fachlich angeforderten Funktionalitäten, die in den Tests zur Funktionalität stattfindet.
- Die Benutzbarkeit einer Web-Applikation wird durch Content-, Oberflächen-, Browser-, Usability-, Zugänglichkeits- und Auffindbarkeitstests nachgewiesen.

- Der Content-Test ist eine Erweiterung des Dokumententests für Web-Auftritte. Er stellt für eine Website sicher, dass die Benutzererwartungen erfüllt werden, der Aufklärungspflicht genüge getan ist und keine rechtlichen Verstöße vorliegen.
- Der Content-Test wird von Rechts- und Marketingexperten anhand von Checklisten in Reviews oder schriftlichen Stellungnahmen an einem Prototyp durchgeführt.
- Der Oberflächentest überprüft die Darstellung, die Erreichbarkeit und allgemeingültige Funktionalitäten der Anwendung auf der Browser-Oberfläche.
- Anwendungsspezifische Funktionen, für die mit den entsprechenden Methoden fachliche Testfälle erstellt werden, bleiben beim Oberflächentest unberücksichtigt.
- Eine Checkliste liefert die Testfälle zum Oberflächentest von Web-Anwendungen und standardisiert die Tests.
- Der Browser-Test prüft die Anwendung mit verschiedenen Browser-Typen, Browser-Versionen und Betriebssystemen. Eine Browser/Betriebssystem-Risikomatrix hilft, die Anzahl der Tests zu reduzieren. Je nach Risikoklasse werden unterschiedlich intensive Tests durchgeführt.
- Im Usability-Test wird der Benutzer im Usability-Labor beobachtet. Alternativ oder ergänzend dazu liefern mündliche und schriftliche Befragungen Informationen zur Gebrauchstauglichkeit einer Web-Anwendung. Der Usability-Test wird von Fachleuten durchgeführt.
- User Tracking Tools und Eye Tracker liefern Informationen über das Benutzerverhalten und den Datenverkehr auf einer Website. Die Gebrauchstauglichkeit einer Website muss während des produktiven Betriebes überwacht werden.
- Der Zugänglichkeitstest stellt den barrierefreien Zugriff auf eine Website sicher. Die Anforderungen zur Barrierefreiheit sind international durch die Web Accessibility Initiative (WAI) und national durch die Barrierefreie Informationstechnik-Verordnung (BITV) festgelegt und priorisiert.
- Auffindbarkeit wird durch richtige Namensgebung der URL und Optimierung der Web-Seiten für Suchmaschinen durch Experten erreicht. Eine barrierefreie Web-Seite ist für Suchmaschinen gut auffindbar.

10 Test zur Änderbarkeit und Übertragbarkeit

„Das Problem zu erkennen ist wichtiger, als die Lösung zu erkennen, denn die genaue Darstellung des Problems führt zur Lösung.“

Albert Einstein (1879 – 1955)

Eine Software, die nach vorgegebenen Programmierstandards und modular entwickelt ist, kann mit geringerem Aufwand geändert, erweitert und auf andere Systeme übertragen werden, als unstrukturiert entwickelte. Die Prüfung des Qualitätsmerkmals Änderbarkeit wird in Code-Analysen vorgenommen. Die Eigenschaft der Übertragbarkeit einer Anwendung auf unterschiedliche Zielsysteme wird im Installationstest getestet.

10.1 Code-Analysen

Code-Analysen sind als statische Prüfungen spezielle Ausprägungen des Dokumententests (s. Kap. 7.2), die auf Code-Ebene eingesetzt werden. Die zu prüfenden Dokumente enthalten entweder Pseudocode, welcher der Programmgenerierung dient, oder den programmierten Sourcecode. Zu den Code-Analysen gehören der Code-Walkthrough und die Code-Inspektion, die beide in Review-Sitzungen durchgeführt werden, der Schreibtischtest, der eine Ausprägung der schriftlichen Stellungnahme ist, und die

statische Code-Analyse, die mit Werkzeugen vorgenommen wird.

Regeln für Review-Sitzungen

Für die Durchführung von Code-Walkthroughs und Code-Inspektionen gelten folgende Regeln:

- Alle Teilnehmer müssen rechtzeitig die zu prüfenden Unterlagen erhalten und durcharbeiten.
- Der Moderator, der dem Qualitätssicherungsteam angehört, ist für die Versendung der Unterlagen und der Einladungen an die Teilnehmer verantwortlich.
- Alle Teilnehmer müssen zum einen mit den fachlichen Anforderungen und zum anderen mit der Programmiersprache, in der das zu prüfende Modul programmiert ist, vertraut sein.
- Fragen, die Teilnehmer in der Sitzung klären möchten, sind vorab schriftlich an den Moderator und den Autor zu senden, damit diese sich darauf vorbereiten können und die Sitzung zügig durchgeführt werden kann.
- Die Prüfungen werden in kleinen Gruppen von drei bis vier Teilnehmern durchgeführt und dauern ca. zwei Stunden.

10.1.1 Code-Walkthrough

Code-Walkthrough

Ein Code-Walkthrough[1] ist eine visuelle Überprüfung von Pseudo- oder Sourcecode. Dabei werden Programmabläufe anhand von Beispieldaten nachvollzogen.

Bei einem Code-Walkthrough wird das Programm nicht (vor-) gelesen, sondern das Testteam spielt anhand der ausgewählten Testfälle Computer, indem die Verarbeitung der Daten im Programmcode manuell nachvollzogen wird. Durch Diskussionen der Gruppe mit dem Programmierer werden Fehler und Unvollständigkeiten aufgedeckt. Während der Sitzung werden keine Lösungen diskutiert, sondern Fehler „nur“ aufgedeckt und dokumentiert.

[1] Hier wird bewusst der Begriff Code-Walkthrough und nicht Walkthrough verwendet, weil sich ein Walkthrough im allgemeinen Sinne nicht auf Dokumente beschränkt, die Programmcode enthalten (s. [Spillner_2005]).

Die Testszenarien und Beispieldaten für einen Code-Walkthrough werden vor der Sitzung mit Blackbox-Verfahren (Kap. 4.1) entworfen.

10.1.2 Code-Inspektion

Code-Inspektion

Eine Code-Inspektion[2] ist eine visuelle Überprüfung von Sourcecode. Dabei wird das Programm mit Hilfe von Checklisten zu Programmierstandards und häufig vorkommenden Fehlern analysiert.

In der Inspektionssitzung erklärt der Autor die Ablauflogik des Moduls. Hierbei stellt das Team Fragen, wobei durch die auftretenden Diskussionen Fehler und Unvollständigkeiten aufgedeckt werden. Zusätzlich wird das Programm mit Hilfe einer Checkliste geprüft.

Die folgende Checkliste zur Code-Inspektion ist, obwohl objektorientiert ausgerichtet, programmiersprachenneutral. Sie ist als „Ideenspender" zu sehen, um daraus eigene, programmiersprachenspezifische Checklisten zu entwickeln.

✓ Checkliste Code-Inspektion

Konformität

- Entsprechen alle Namen den gültigen Konventionen?
- Sind Namensgebungen eindeutig und selbstbeschreibend für
  - Dateien
  - Grafiken
  - Links
  - Menüs
- Sind die Programmierrichtlinien eingehalten?
- Sind die Layout-Konventionen eingehalten?
- Sind wiederverwendbare Module ausgelagert?

[2] Mit der gleichen Begründung wie beim Code-Walkthrough wird auch hier eine Code-Inspektion und keine Inspektion (s. [Spillner_2005]) im allgemeinen Sinn beschrieben.

Namenskonventionen

- Haben Variablen-, Konstanten-, Methoden- oder Klassennamen ausgeschriebene Namen, die den Inhalt oder die Aufgabe beschreiben?
- Besitzen die Namen der Methoden, die Klassenvariablen mit neuen Werten belegen, das Präfix „set“?
- Besitzen die Namen der Methoden, mit denen der Wert von Klassenvariablen ausgelesen wird, das Präfix „get“?
- Gibt es keine Namen, die sich nur in der Groß-/Klein-Schreibweise voneinander unterscheiden?
- Beginnen Variablen- und Methodennamen immer mit einem Kleinbuchstaben?
- Beginnen keine Variablennamen mit einem „_“ oder „$“.
- Werden Variablennamen für unterschiedliche Aufgaben oder Datentypen innerhalb einer Methode nicht mehrfach verwendet?
- Entsprechen die Namen der Variablen für mathematische Aufgaben den in der Mathematik geläufigen Namen?
- Beginnen Klassennamen mit einem Großbuchstaben?
- Bestehen Konstanten nur aus Großbuchstaben, Ziffern und dem Underscore?

Kommentare

- Ist der Code ausreichend kommentiert?
- Beschreiben die Code-Kommentare die inhaltliche Wirkung der Anweisungen und nicht nur die reine Syntax?
- Ist „ineffizienter Code“, der bewusst codiert wurde, mit Namen, Datum und Begründung versehen?
- Geht die Methodenbeschreibung auf alle bekannten Randbedingungen und Sonderfälle ein, sofern diese existieren?
- Ist jeder Autor, der an einer Klasse Veränderungen vorgenommen hat, mit dem vollständigen Namen, dem verwendeten Kürzel und der Firma angegeben?
- Sind Programmmodifikationen in einer Änderungshistorie mit Datum, Bearbeiter und Beschreibung festgehalten?

Berechnungen und Vergleiche

- Werden komplexe Ausdrücke durch zusätzliche Klammerpaare übersichtlich gestaltet?

- Wird eine Konstante stets für denselben Zweck verwendet?
- Werden Berechnungen nur mit numerischen Feldern ausgeführt?
- Sind bei Berechnungen mit unterschiedlichen Datentypen die Konvertierungen richtig?
- Haben bei Vergleichen alle Operanden den gleichen Datentyp bzw. sind die Konvertierungsregeln korrekt?
- Werden Datumsfelder richtig konvertiert?
- Wird Division durch 0 abgefangen?
- Kann es keine Über- oder Unterläufe geben?
- Kann es keine Rundungsfehler geben?
- Sind alle Ausdrücke und Formeln richtig?

Datenreferenzen

- Werden alle Felder explizit definiert?
- Werden alle Variablen richtig initialisiert?
- Werden alle Tabellen richtig initialisiert?
- Liegen alle Indexwerte innerhalb der Tabellengrenzen?
- Sind Indizes ganzzahlig definiert?
- Sind Zuweisungen bzgl. der Zielvariablen sinnvoll?

Steuerung

- Kommt jede Schleife zu einem Ende?
- Werden Schleifenzähler richtig initialisiert?
- Können Schleifenzähler nicht zu groß werden?
- Gibt es keine „toten" Bedingungszweige?[3]

Schnittstellen

- Sind Anzahl und Reihenfolge von Übergabeparametern korrekt?
- Stimmen Parameter und Argumente in Typ und Länge überein?
- Sind die gleichen Parameter immer in der gleichen Reihenfolge definiert, wenn sie von mehreren Methoden verwendet werden?

---

[3] Achtung, falls die C1-Testabdeckung gefordert ist (s. Abschnitt 4.2.4, „Eine Erfahrung zum C1-Testabdeckungsgrad"), sind „tote" Programmzweige ggf. erforderlich.

- Sind globale Variable notwendig und über alle Module konsistent?
- Werden Eingabeparameter nicht verändert?
- Stimmen Ein/Ausgabe-Bereiche mit den Satzlängen überein?
- Werden Cursor- und Dateiendebedingungen erkannt und richtig behandelt?
- Werden Cursor und Dateien rechtzeitig und korrekt eröffnet?
- Werden Cursor und Dateien geschlossen?

Fehlerbehandlung

- Werden alle Eingabedaten auf Gültigkeit geprüft, falsche Eingaben erkannt und richtig behandelt?
- Werden alle Datenbank-Fehler richtig erkannt und behandelt?
- Werden alle System-Fehler richtig erkannt und behandelt?
- Werden Commit-Points richtig gesetzt?
- Sind Fehler- und Hinweistexte korrekt und sinnvoll?

Website

- Hat jedes Unterverzeichnis der Website eine eigene Index-Seite (Index.htm)?
- Gibt es eine erklärende HTML-Seite, die bei Fehlern angezeigt wird?
- Sind Grafiken im Code mit Größenangaben versehen?
- Sind zum Download vorgesehne Dateien im Code mit Größenangaben versehen (Grafiken, Filme, Bilder, PDF-Dateien,...)?
- Sind Menüs mit Default-Werten belegt?
- Wird der Zielinhalt eines internen Links in dem Fenster geöffnet, in dem sich der interne Link befindet?
- Wird der Zielinhalt eines externen Links in einem neuen Fenster geöffnet?
- Sind die Web-Seiten für Suchmaschinen optimiert?[4]

[4] S. Abschnitt 9.6.2, Suchmaschinenoptimierung

10.1.3 Schreibtischtest

Code-Walkthroughs und Code-Inspektionen sind aufwendige Review-Arten. Eine Alternative ist der Schreibtischtest.

Schreibtischtest

> Der Schreibtischtest ist eine Ein-Personen-Code-Inspektion kombiniert mit einem Ein-Personen-Walkthrough.

Beim Schreibtischtest wird Sourcecode an Hand von Beispielen und Checklisten durch eine Person überprüft, die nicht der Autor des Dokumentes ist. Fragen, Fehler und Verbesserungsvorschläge werden notiert und anschließend mit dem Autor besprochen. Der Schreibtischtest fasst die Methoden Code-Walkthrough und Code-Inspektion in einer „Light"-Ausprägung zusammen, was diese Form der Qualitätssicherung effektiv und relativ kostengünstig macht.

10.1.4 Statische Code-Analyse durch Werkzeuge

statische Code-Analyse

Die Überprüfung von Programmcode auf Einhaltung von Programmierstandards und Richtlinien kann, wie oben beschrieben, mit Hilfe von Checklisten manuell vorgenommen werden. Effizienter wird eine statische Code-Analyse von Code Analyzern durchgeführt, die folgende Aufgaben erledigen:

- Prüfung der Syntax
- Prüfung auf korrekte und typgerechte Verwendung, Initialisierung und Referenzierung der Variablen und Klassen
- Aufspüren von nicht erreichbarem Code
- Erkennen von Speicherlecks (s. Abschnitt 11.1.4)
- Erstellung der Verwendungsnachweise von Variablen und Funktionen (Cross Reference)
- Ermittlung der Komplexität eines Programms

Compiler

Jeder Compiler bietet Möglichkeiten zur statischen Analyse des Quellcodes eines Programms.

Code Analyzer

Darüber hinaus gibt es für jede gängige Programmiersprache Code Analyzer, die Sourcecode nach vorgebbaren Regeln überprüfen ([URL: TestToolsFAQ] Rubrik „Static Analysis“) und Datenfluss- und Kontrollflussanalysen durchführen. Speziell für Java sind Open Source Tools unter [URL: TestToolsJava] in der Rubrik „Code Analyzers“ zu finden.

W3C-Validatoren

Für Web-Programmiersprachen gibt es kostenlose Hilfsmittel zur Code-Überprüfung. Zum Beispiel bietet das World Wide Web Consortium Validatoren für HTML, XHTML[GL], CSS[GL] und XML[GL] an ([URL: W3C-validatorMarkup], [URL: W3C-validatorCSS], [URL: W3C-validatorXML]).

HTML Tidy

Das Programm HTML Tidy von Dave Raggett repariert und verbessert sogar den HTML-Code. Die Prüftiefe, wie zum Beispiel bei der Analyse der Zugänglichkeitsrichtlinien (s. Kap. 9.5), kann in HTML Tidy voreingestellt werden. Erweiterungen dieses Open-Source Programms für Windows und andere Betriebssysteme sind neben dem Original im sourceforge.net zu finden ([URL: Tidy]). Es gibt allerdings eine Einschränkung. Dynamische HTML-Seiten, wie sie in der Regel bei umfangreicheren Web-Anwendungen generiert werden, können mit Tidy nicht geprüft werden.

10.1.5 Qualitätsanforderungen zu Code-Analysen

Die Qualitätsanforderungen zur Änderbarkeit und Übertragbarkeit werden durch Prüfprotokolle und Checklisten nachgewiesen, die während der Code-Analysen erstellt bzw. ausgefüllt werden.

Eine wichtige Rolle bei der Festlegung der Qualitätsanforderungen spielt die geplante Vermarktung und der Einsatzbereich einer Anwendung. Das bedeutet, dass die konkreten Qualitätsanforderungen von der Beantwortung der folgenden Fragen abhängen:

- Wird die Anwendung nur im eigenen Hause betrieben?
- Soll die Web-Applikation für mehrere Mandanten eingesetzt werden?
- Wird die Anwendung an Kunden geliefert, die individuelle Anpassungen beauftragen werden?
- Sind regelmäßige Updates und funktionale Erweiterungen geplant?

- Werden einzuhaltende Standards vom Auftraggeber vorgegeben?
- Wie viele und welche Module müssen wiederverwendbar sein, weil sie in anderen Anwendungen implementiert werden sollen?

Aspekt Wiederverwendbarkeit

Qualitätsanforderungen unter dem Aspekt der Wiederverwendbarkeit könnten lauten:

- Software-Komponenten, die als wiederverwendbar eingestuft sind, müssen einer Code-Inspektion mit mindestens drei Experten unterliegen. Für sie muss die Checkliste „Code-Inspektion" zu 100% erfüllt sein und eine automatische Code-Analyse durchgeführt werden, deren Ergebnisbericht keine Fehler und keine Warnungen ausweisen darf.
- Software-Komponenten, die als nicht wiederverwendbar eingestuft sind, müssen einen Schreibtischtest bestehen. Für sie muss vorab eine automatische Code-Analyse durchgeführt werden, deren Ergebnisbericht nur Warnungen und keine Fehler ausweisen darf.

10.1.6 Empfehlungen zu Code-Analysen

statische Code-Analyse zuerst durchführen

Jeder visuellen Code-Analyse sollte eine statische Code-Analyse mit einem Werkzeug vorangehen. Je nach Umfang der statischen Analyse kann in Abhängigkeit der geforderten Qualitätsanforderungen auf eine manuelle Code-Analyse verzichtet werden.

Code-Analysen durch Entwickler

Code-Analysen finden sehr entwicklungsnah statt und sollten daher in der Teststufe Komponententest durchgeführt werden. Automatisierte Code-Analysen sollten schon während des Entwicklertests stattfinden, um in der Entwicklungsphase Codier-Fehler aufzudecken. Die erzeugten Prüfprotokolle sollten als Testnachweis mit der zu testenden Komponente an das Testteam gegeben werden.

Checklisten als konstruktive Maßnahme

Die Checklisten zur Code-Inspektion sollten, sofern es diese im Unternehmen noch nicht gibt, von Experten für die eingesetzten Programmiersprachen erarbeitet werden. Als konstruktive Qualitätsmaßnahme sollten die Checklisten zum Beginn der Realisierungsphase den Entwicklern zur Verfügung gestellt werden.

10.2 Installationstest

Wenn eine Software auf unterschiedlichen Systemumgebungen eingesetzt werden soll, muss sie auf diese übertragen werden. Gegebenenfalls müssen Software-Komponenten nicht nur auf Servern, sondern auch auf Clients installiert werden. Neben den Installationsprozessen müssen auch die Prozesse der Deinstallation alter Programmversionen getestet werden.

Installationstest

> Der Installationstest prüft die Installierbarkeit und Deinstallierbarkeit einer Anwendung.

Installierbarkeit

Installierbarkeit ist die Fähigkeit eines Software-Produkts in einer festgelegten Umgebung installierbar zu sein. Sie wird am Aufwand gemessen, der zur Installierung der Software in einer festgelegten Umgebung notwendig ist.

Deinstallierbarkeit

Für die Deinstallierbarkeit einer Software gelten die entsprechenden Aussagen.

10.2.1 Installationsphasen

Der Installationstest stellt sicher, dass Installationsprozesse effizient und fehlerfrei durchgeführt werden können. Zu den Installationsprozessen gehören die Erstinstallation, die Installation von neuen Versionen und die Deinstallation einer Software.

Test der Erstinstallation

Für die Installation einer Software werden eigenständige Programme, die Installationsroutinen, benötigt. Diese müssen, wie die zu installierende Software selbst, getestet werden. Die Prüfung der Lizenzvergabe gehört ebenfalls zum Installationstest.

Eine Installationsroutine verlangt in der Regel vom „Installateur“ Eingaben und Parametereinstellungen. Für diese Eingabemöglichkeiten müssen Testfälle entworfen werden.

Auch technische Belange müssen berücksichtigt werden. Zum Beispiel muss die Klassen-ID eines OLE-Objektes (COM/DCOM) in der Registrierungsdatenbank des Betriebssystems eingetragen sein, damit die Anwendung funktioniert. Oder wenn die Anwendung JavaScript benutzt, muss diese Einstellung im Browser bei der Installation aktiviert werden (s. Abschnitt. 9.3.3). Eventuell benötigt der Benutzer Administratorrechte zur Installation von Komponenten auf dem Client.

Test der Update-Installation

Nicht nur die Erstinstallation, sondern auch die Installation eines Updates ist vor Auslieferung einer neuen Software-Version anhand von Testfällen zu testen.

Test der Deinstallation

Ebenso müssen Deinstallationsroutinen, die ebenfalls zum Lieferumfang einer Software gehören, getestet werden. Sie dürfen zum Beispiel keine falschen Daten löschen. Die Deinstallation einer Anwendung kann erforderlich sein, bevor eine neue Version installiert oder wenn die bestehende Version endgültig vom Rechner genommen werden soll.

Die Checkliste „Installationstest“ fasst die zu prüfenden Punkte zusammen:

✓ Checkliste Installationstest

Installation

- Wird der Benutzer ggf. darauf hingewiesen, dass er für die Installation Administratorrechte besitzen muss?
- Prüft die Installationsroutine, ob Hard- und Software des Zielsystems kompatibel sind?
- Werden im Fall der Inkompatibilität Fehlerhinweise erzeugt und wird die Installation korrekt abgebrochen?
- Wird geprüft, ob der von der Anwendung benötigte freie Hauptspeicherplatz vorhanden ist?
- Wird geprüft, ob genügend Speicherplatz auf der Festplatte vorhanden ist?
- Wird bei Abbruch des Installationsvorgangs der ursprüngliche Zustand wieder hergestellt?
- Wird der Benutzer darauf hingewiesen, wenn existierende Dateien mit neuen Versionen überschrieben werden?
- Sind alle Eingaben, die zur Installation notwendig und möglich sind, in Testfällen abgebildet?
- Werden die benötigten Komponenten (ActiveX-Controls, DLLs, Plugins,...) richtig installiert?
- Wird auf bereits installierte Komponenten (ActiveX-Controls, DLLs, Plugins,...), die mit den zu installierenden Komponenten kollidieren können, hingewiesen?
- Werden Browser richtig konfiguriert? (Werden zum Beispiel JavaScripts benutzt, so muss diese Option bei der Installation automatisch aktiviert bzw. angefordert werden.)
- Ist die Installation von allen erforderlichen Medien möglich (CD, DVD, Download vom Web-Server) und wurde sie damit getestet?

Lizenzen

- Werden die Lizenzvereinbarungen dargestellt und müssen sie bestätigt werden?
- Wird die Installation abgebrochen, wenn die Lizenzvereinbarungen nicht bestätigt werden?
- Wird die Lizenzschlüssel-Vergabe geprüft?

Update

- Werden Benutzer online über neue, bereitstehende Versionen (Updates) informiert?
- Ist das Verfahren zur Verteilung von Updates beschrieben und getestet?
- Werden vor dem Update die vorgefundenen Zustände gesichert?
- Werden im Bedarfsfall Sicherungen wieder richtig zurückgespielt?
- Wird beim Update überprüft, ob die installierte Version kompatibel zur neuen Version ist?
- Gehen beim Update keine Nutzerdaten verloren?
- Werden Benutzereinstellungen übernommen?
- Werden Datenbank- und Dateiinhalte vollständig und korrekt übernommen?

Deinstallation

- Werden nur zur Anwendung gehörende DLL's und Dateien gelöscht?
- Werden alle nicht benötigten Dateien, Icons, Menüeinträge und Directories vollständig entfernt?
- Werden die Registry Keys entfernt?

Für jeden in der Checkliste aufgeführten Themenbereich müssen Testfälle beschrieben werden. Dieses geschieht mit den Methoden der systematischen Testfallerstellung (Kap. 4), wobei die Anforderungen nicht nur fachlicher, sondern auch technischer Natur sind.

Installationshandbuch prüfen

Die Prozesse, die bei einer Installation durchzuführen sind, werden für den „Installateur" in einem Installationshandbuch beschrieben. Diese spezielle Benutzerdokumentation muss vor Auslieferung einem Dokumententest (s. Kap. 7) unterzogen werden.

10.2.2 Werkzeuge für den Installationstest

Capture Replay Tools

Für den Installationstest können Capture Replay Tools (Kap. 12.4.2) zur Testautomatisierung eingesetzt werden. Wenn der Installationstest für unterschiedlich konfigurierte Zielsysteme mehrfach durchgeführt werden muss, amortisiert sich der Automatisierungsaufwand sehr schnell.

10.2.3 Qualitätsanforderungen zum Installationstest

Testfallabdeckung

Die Installierbarkeit ist ein Teilmerkmal des Qualitätsmerkmals Übertragbarkeit. Sie wird durch den Testnachweis aller Testfälle zum Installationstest sichergestellt. Eine Testfallabdeckung von 100% ist hier gefordert.

Usability der Installationsanweisungen

Wenn der Auftraggeber bzw. der Benutzer die Software selbständig installieren muss, müssen Qualitätsanforderungen an die Benutzerfreundlichkeit der Installationsanweisungen und den mit der Installation verbundenen Aufwand gestellt werden.

Dazu könnte von einem Usability-Test gefordert werden, dass 20 Testpersonen mit unterschiedlichen IT-Kenntnissen die Software anhand der Installationsanweisungen selbständig installieren und wieder deinstallieren müssen. Alle Testpersonen müssen die Installation innerhalb von 12 Minuten und die Deinstallation innerhalb von 7 Minuten fehlerfrei durchführen.

10.2.4 Empfehlungen zum Installationstest

Ein Problem des Installationstests kann die unbekannte Vielzahl der möglichen Systemvarianten sein. Daher ist es auch für den Installationstest sinnvoll, die wichtigsten Systemkonfigurationen in einer Risikoanalyse zu priorisieren und die Zielumgebungen entsprechend der vorgenommenen Bewertung nachzustellen. Das Vorgehen entspricht dem, wie es zur Erstellung der Risikomatrix zum Browser-Test im Abschnitt 9.3.1 beschrieben ist.

10.3 Zusammenfassung

- Code-Analysen und Installationstests stellen die Qualitätsmerkmale Änderbarkeit und Übertragbarkeit einer Software sicher.
- Mit Code-Analysen werden Dokumente, die Pseudo- oder Programmcode enthalten, geprüft.
- Eine Code-Analyse kann von Experten ohne Werkzeuge in einem Code-Walkthrough, einer Code-Inspektion oder einem Schreibtischtest durchgeführt werden.
- Programmiersprachenspezifische Checklisten unterstützen Code-Inspektionen und Schreibtischtests.
- Der Schreibtischtest ist eine kostengünstige Alternative zu Code-Walkthrough und Code-Inspektion.
- Eine statische Code-Analyse wird automatisiert mit einem Compiler oder einem speziellen Code Analyzer vor der visuellen Code-Analyse durchgeführt.
- Ein Installationstest muss durchgeführt werden, falls die Anwendung auf mehrere Rechner übertragen werden soll.
- Neben der Erstinstallation müssen auch die Installationen von Updates und die Deinstallation der Software getestet werden.
- Installationshandbücher werden einem Dokumententest unterzogen.

11 Tests zur Effizienz und Zuverlässigkeit

Januar 2006: „Nach massiven Verkäufen ist in Tokio der Aktienhandel vorzeitig beendet worden - zum ersten Mal in der 56-jährigen Geschichte der Börse.

...

Die Zahl der Transaktionen hatte die Marke von vier Millionen erreicht und war damit an die Grenze der Kapazität des Computersystems der Börse gestoßen.“

Zitat aus [URL: SZ]

Die in diesem Kapitel beschriebenen Tests zur Effizienz und Zuverlässigkeit sollen Situationen vermeiden helfen, wie sie im Januar 2006 in die Schlagzeilen gekommen sind.

Ein Anwendungssystem muss sowohl unter normalen Betriebsbedingungen als auch in Ausnahmesituationen akzeptables Zeitverhalten aller Funktionen aufweisen und stabil bleiben. Diese Eigenschaften werden durch die Performanz/Lasttests sichergestellt.

Der Ausfallsicherheitstest sorgt dafür, dass ein System auch in unvorhergesehenen Situationen beherrschbar bleibt und keine Daten verloren gehen.

Der Verfügbarkeitstest misst die Zuverlässigkeit eines Systems während der Betriebszeit.

11.1 Performanz-/Lasttests

Wenn in diesem Buch von Performanz-/Lasttests gesprochen wird, ist eine Gruppe von verwandten Testtypen gemeint. Dabei handelt es sich um Performanz-, Last-, Skalierbarkeits- und Speicherlecktest, die alle das Qualitätsmerkmal Effizienz überprüfen, allerdings mit unterschiedlichen Zielrichtungen und jeweils unter anderen Bedingungen. Sie werden in den folgenden Abschnitten beschrieben und versetzen die Tester in die Lage, folgende Fragen beantworten zu können:

- Werden Funktionen und Transaktionen[GL] innerhalb der geforderten Zeitvorgaben ausgeführt?
- Werden die geforderten Mengen in der geforderten Zeit übertragen und verarbeitet?
- Hält das System die Grundlast über längere Zeit?
- Können Belastungsspitzen des Systems ohne Qualitätsverlust abgefangen werden?
- Wird der Nutzer bei Überlastung informiert?
- Werden wohldefinierte Lastspitzen richtig ausbalanciert (Load Balancing)?
- Ist das System problemlos aufrüstbar?
- Tritt im Dauerbetrieb kein Performanzproblem durch Speicherlecks auf?

11.1.1 Performanztest

Performanz

Performanz (Zeitverhalten) beschreibt das Ausmaß, in dem eine Komponente oder ein System die geforderten Funktionen und Leistungen im Rahmen vorgegebener Bedingungen erbringt.

Performanztest

> Der Performanztest prüft das Zeitverhalten, die Mengenverarbeitung und den Ressourcenverbrauch eines Anwendungssystems unter normalen Systembedingungen.

Der Performanztest wird in vier Schritten durchgeführt.

1. Schritt: Testszenarien pro Nutzergruppe festlegen

Im Performanztest wird ein realistischer Betrieb mit der zu testenden Anwendung simuliert. Die Testszenarien für den Performanztest bilden die Geschäftsprozesse ab, die von unterschiedlichen Nutzergruppen der Web-Anwendung ausgeführt werden. Sie können mit Hilfe der anwendungsfallbasierten Testfallermittlung entworfen werden (s. Abschnitt 4.1.5).

Nehmen wir unser Beispiel des Finanzierungsrechners wieder auf. Hier können für den Performanztest die Szenarien mit den drei Nutzergruppen Kunden, Interessenten und Informationssuchende aus dem Usability-Test (Kap. 9.4) übernommen werden:

1. Kunden konfigurieren ihr Wunschauto und fordern ein verbindliches Finanzierungsangebot an.
2. Interessenten konfigurieren mehrere Autos und berechnen dazu unterschiedliche Finanzierungsmöglichkeiten.
3. Informationssuchende surfen durch die Anwendung und fordern allgemeine Finanzierungsunterlagen auf dem Postweg an.

2. Schritt: Anzahl, Zeitpunkt, Dauer und Frequenz festlegen

Um realistische Messergebnisse durch den Performanztest zu erhalten, wird für jedes Szenario aufgrund von Schätzungen, Umfragen oder Erfahrungswerten festlegt,

- wie viele Benutzer es parallel ausführen,
- zu welchen Zeitpunkten es ausgeführt wird,
- wie lange es durchgeführt wird,
- wie häufig es ausgeführt wird.

3. Schritt: Belastungsspitzen berücksichtigen

Bekannte, regelmäßig auftretende Situationen, die für eine bestimmte Dauer eine hohe Belastung für die Anwendung zur Folge haben, müssen beim Performanztest berücksichtigt werden.

Zum Beispiel muss bei einer Bankanwendung bedacht werden, dass sich täglich bei Börsenöffnung alle Aktienhändler gleichzeitig in die Broker-Software einloggen oder dass die Bank zu festen Terminen Massenüberweisungen veranlasst. Beides kann Auswirkungen auf die Performanz des Gesamtsystems haben.

Ein Freemail-Anbieter muss damit rechnen, dass sich zwischen 12:00 und 13:00 Uhr, wenn in der Mittagspause privat gesurft wird, die Anzahl der Besucher seiner Website verdoppelt.

Solche Anwendungsspitzen treten im normalen Betrieb auf, sind planbar und müssen bei der Durchführung der Tests simuliert werden. Sie gehören nicht zu den Ausnahmesituationen, wie sie beim Lasttest (s.u.) betrachtet werden.

4. Schritt: Tests durchführen und auswerten

Die mit Anzahl, Zeitpunkt, Dauer und Frequenz versehenen Testszenarien werden in einem „Testdrehbuch" zum Performanztests zusammengefasst und nach diesem durchgeführt.

Dabei werden unter anderem Antwortzeit- und Speicherverhalten des Systems gemessen und mit den Qualitätsanforderungen aus den Spezifikationen verglichen. Welche Größen im Performanztest gemessen und ausgewertet werden können, ist im Abschnitt 11.1.6 beschrieben.

11.1.2 Lasttest

Wenn die Ergebnisse der Performanzmessungen zufriedenstellend ausgefallen sind, müssen mögliche Ausnahmesituationen nachgestellt werden. Solche entstehen zum Beispiel, wenn auf Grund gestarteter Werbekampagnen oder besonderer Nachrichtensituationen ungewöhnlich viele Benutzer die Website aufsuchen.

Lasttest

Der Lasttest prüft das Verhalten des Gesamtsystems bei steigender Systemlast.

Last wird für das zu testende System dadurch erzeugt, dass die Testszenarien aus dem Performanztest stetig gesteigert werden. Das heißt, die Anzahl der Benutzer, die Wiederholungsraten der Transaktionen und die zu verarbeitenden Mengen werden so lange erhöht, bis das System abnormale Reaktionen zeigt. Der geneigte Tester hat natürlich Spaß daran, das Ganze bis zum Absturz des Systems zu treiben.

Stresstest

Dieser sogenannte Stresstest wird durchgeführt, um ein System an oder oberhalb der Grenzen, die in den Anforderungen spezifiziert wurden, bewerten zu können.

Massentest/ Volumentest

Das System kann auch gestresst werden, indem ihm große Datenmengen zur Verarbeitung zugeführt werden. Diese Art des Tests wird Massentest oder Volumentest genannt. Ein Massentest ist für eine Online-Anwendung von besonderer Bedeutung, wenn Systemprozesse zur Verarbeitung großer Datenmengen parallel auf dieselben Ressourcen wie die Online-Komponenten zugreifen müssen und die Online-Anwendung rund um die Uhr zur Verfügung stehen muss.

Schwachstellen suchen

Durch den Lasttest werden Schwachstellen (bottle necks) im System festgestellt. Gerade bei komplexen Web-Applikationen sind die potentiellen Schwachstellen sehr vielfältig und nicht immer einfach zu lokalisieren. Engpässe können in einer komplexen Systemlandschaft überall auftreten, zum Beispiel im Web-Server, im Applikations-Server, im DNS-Server[GL], im Router, in der Firewall, im Datenbank-Server oder in der Datenbank.

11.1.3 Skalierbarkeitstest

Während der Lasttest sein Augenmerk auf das Erkennen von Schwachstellen legt, wird mit dem Skalierbarkeitstest ermittelt, ob das System problemlos technisch aufgerüstet werden kann.

Skalierbarkeit

Skalierbarkeit ist die Fähigkeit eines Software-Produkts, technisch aufgerüstet werden zu können, um eine erhöhte Last zu verkraften.

Ein System skaliert zum Beispiel gut, wenn es bei der doppelten Anzahl von Prozessoren die Hälfte der Rechenzeit benötigt oder bei Verdoppelung der Belastung (Anzahl User, Anzahl Transaktionen) mit den doppelten Hardware-Ressourcen auskommt, ohne dass Eingriffe in die Software notwendig sind. Ein System skaliert schlecht, wenn bei doppelter Last die fünffachen Ressourcen zum Ausgleich der erhöhten Last benötigt werden.

Skalierbarkeitstest

> Der Skalierbarkeitstest prüft die Fähigkeit, ein Software-Produkt auf Rechnern von unterschiedlicher Größe einsetzen zu können.

Beim Skalierbarkeitstest wird die Belastung des Systems wie beim Lasttest sukzessive gesteigert:

1. Stufe: Für jede Nutzergruppe werden die Testszenarien (Geschäftsvorfälle) einzeln durchgeführt.

2. Stufe: Die Szenarien werden pro Nutzergruppe zusammengefasst und parallel durchgeführt.
3. Stufe: Alle Szenarien aller Nutzergruppen werden gemeinsam durchgeführt.

11.1.4 Speicherlecktest

Speicherleck

Ein Speicherleck (memory leak) ist ein Software-Fehler, bei dem ein Teil des Arbeitsspeichers nach Benutzung nicht wieder freigegeben wird. Über die Zeit verliert der betroffene Web-Server verfügbaren Speicher, bis er „leckt", d.h. arbeitsunfähig ist oder sogar abstürzt.

Speicherlecktest

Der Speicherlecktest überprüft als ein auf Dauer angelegter Performanztest das Entstehen von Speicherlecks.

Speicherlecktest über mehrere Tage.

Speicherlecks werden nach der Durchführung des Performanztests gesucht. Um sie aufzudecken, muss ein Performanztest gefahren werden, der vier bis fünf Tage dauert. Speicherlecks werden daran erkannt, dass Antwortzeiten oder Übertragungsraten stetig nachlassen.

Da schon „kleine" Speicherlecks die Leistung eines Systems markant beeinflussen können, ist dieser Test für die Zuverlässigkeit eines Systems sehr wichtig.

11.1.5 Automatisierung der Performanz-/Lasttests

Die vier beschriebenen Ausprägungen des Performanz-/Lasttests haben gemeinsam, dass sie dieselben Geschäftsprozesse simulieren. In der Testdurchführung unterscheiden sie sich allerdings in Dauer und Intensität. Es liegt auf der Hand, dass die Tests nicht manuell durchgeführt werden können, denn es müssten einige hundert oder sogar einige tausend Benutzer die Web-Anwendung über einen längeren Zeitraum bedienen.

Load Test Tools

Performanz-/Lasttests werden mit Lasttestwerkzeugen (Load Test Tools) durchgeführt. Sie simulieren beliebig viele virtuelle Benutzer, welche die ihnen zugewiesenen Testszenarien parallel durchführen. Dabei werden Antwortzeiten aufgezeichnet, übertragene Datenmengen gezählt und verbrauchte Ressourcen (CPU,

temporärer Speicher,...) gemessen. Die Testszenarien werden wie beim Regressionstest aufgezeichnet, allerdings nicht über die Objekte auf der Browser-Oberfläche, sondern auf Netzwerkprotokollebene, wie zum Beispiel HTTP (Hypertext Transfer Protocol). Damit die Skripte mehrfach und parallel wieder abgespielt werden können, werden die aufgezeichneten Testskripte parametrisiert.

Monitore

Mit zusätzlichen Monitoren können Schwachstellen des Systems genau geortet werden. Sie können an unterschiedlichen Stellen im Netzwerk angelegt werden (s. Abb. 11.1).

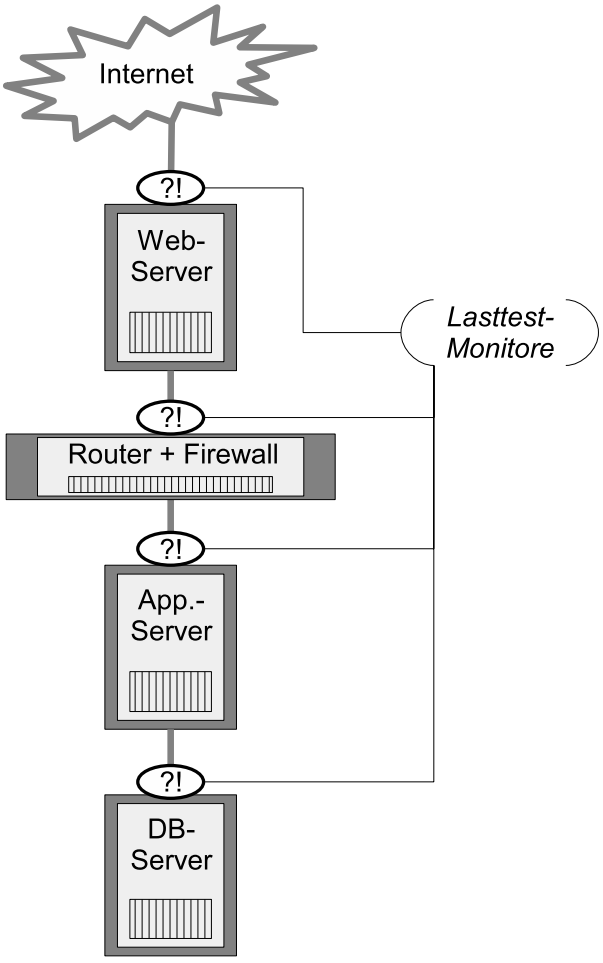


Abb. 11.1: Monitore

Monitore messen zum Beispiel die Transaktionszeiten zwischen Client und Server, die im Netzwerk auftretenden Verzögerungen, den Ressourcenverbrauch von Web- und Applikations-Servern und die Effizienz von Datenbank-Zugriffen.

11.1.6 Metriken der Performanz-/Lasttests

Im Folgenden werden einige Metriken zur Durchführung von Performanz-/Lasttests vorgestellt, die mittels Lasttestwerkzeugen gemessen und ausgewertet werden.

Requests Per Second

- RPS (Requests Per Second) – Anzahl der bearbeiteten Anfragen pro Sekunde: Sie zeigen den Umfang der Interaktionen zwischen Browser und Web-Server. Wenn diese Größe mit steigender Last konstant bleibt, ist der Web-Server am Limit.

Prozessorauslastung

- CPU – Prozessorauslastung des Servers in Prozent: Liegt sie dauerhaft deutlich über 50%, ist die Leistung des Prozessors zu gering ausgelegt.

Queued Requests

- QR (Queued Requests) – Anzahl der Abfragen in der Warteschlange: Wenn sie über eine längere Zeit größer Null ist, steht der Server unter Stress.

↓
Beispiele für Performanzmessungen

Im folgenden Beispiel setzen wir den Web-Server[1] unseres Finanzierungsrechners in zehn Testläufen langsam aber stetig unter Last. Als Testergebnis erhalten wir drei Messkurven zu den Metriken RPS, CPU, QR.

1. Beispiel: Requests Per Second

Die erste, in Abb. 11.2 dargestellte Messkurve zeigt die Anzahl der bearbeiteten Anfragen an den Web-Server pro Sekunde (RPS) bei steigender Anzahl der Threads[GL]. Die Anzahl der geladenen Threads steigt von 1 auf 250, wobei die RPS maximal auf 41 steigen.

[1] ... und auch den Systemadministrator ☺

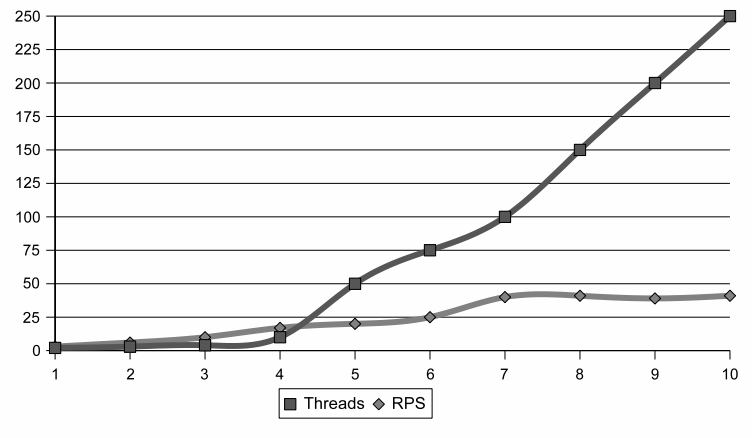


Abb. 11.2: Messung der Requests Per Second

Erkenntnis: Vom System können nicht mehr Requests verarbeitet werden, es ist an seine Grenzen gestoßen.

2. Beispiel: Prozessorauslastung

Die nächste Messkurve (Abb. 11.3) zeigt, dass der Prozessor des Web-Servers keine Schwachstelle darstellt, denn seine Auslastung liegt bei steigender Anzahl der Threads bei maximal 55%.

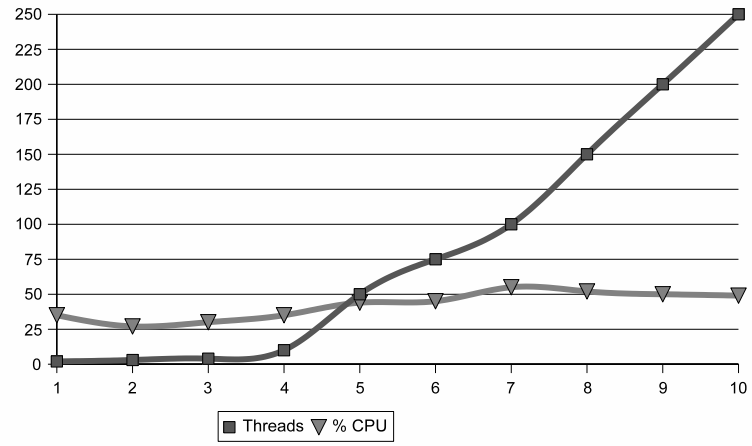


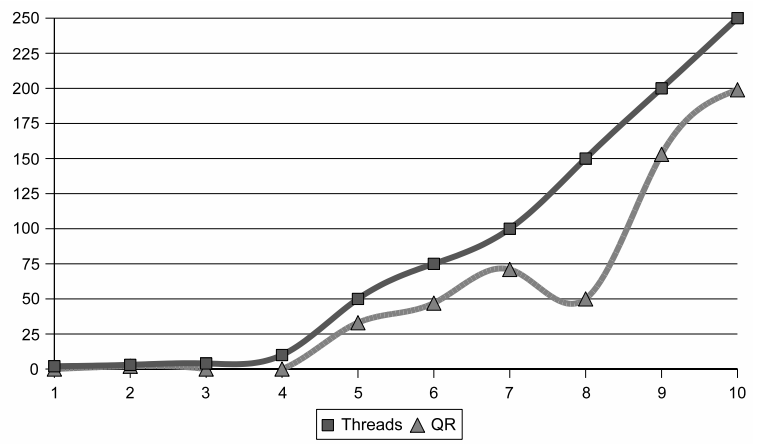
Abb. 11.3: Messung der CPU-Auslastung

Erkenntnis: In diesem Aspekt skaliert das System gut.

3. Beispiel: Queued Requests

Die dritte Messreihe in Abb. 11.4 zeigt die Anzahl der Queued Requests in Abhängigkeit der Thread-Anzahl. Der Anstieg der Queued Requests auf fast 200 bei diesem Testszenario ist extrem.

Abb. 11.4: Messung der Queued Requests



Erkenntnis: Eine Kommunikationskomponente des Web-Servers stellt eine Schwachstelle dar.

↗

Es gibt weitere aussagekräftige Metriken zu den Performanz-/Lasttests:

Transaktionen

- Eine Transaktion umfasst die Anfrage (Request) an den Server, die Bearbeitung der Anfrage und die Antwort (Response) vom Server. Hier ist interessant zu wissen, ob die Anzahl abgeschlossener Transaktionen pro Sekunde proportional zur Steigerung der Anfragen bleibt.

Umlaufzeit

- Die Umlaufzeit misst die Dauer zwischen Beginn und Abschluss einer Transaktion und sollte unter Stress konstant bleiben. Werden die Transaktionen generell langsamer und misslingen, kann der Server keine weiteren Anfragen mehr bearbeiten.

Transaktionsdauer

- Unterschiedliche Testszenarien lösen unterschiedliche Transaktionen aus. Stellt beim Testen von mehreren Transaktionen eine Transaktion einen zeitlichen Ausreißer dar, so deutet das auf Fehler in der Programmierung hin.

- Steigt die Anzahl der gleichzeitigen Verbindungen zum Web-Server bei gleich bleibender Anzahl von Anfragen, bedeutet das, dass die Verbindungen länger als notwendig offen bleiben. *Verbindungen*
- Der verfügbare Speicherplatz steigt mit jeder neuen Verbindung und muss nach Beendigung der Verbindung wieder freigegeben werden. Verringert sich bei gleichbleibender Anzahl von wechselnden Verbindungen die Größe des verfügbaren Speichers stetig, so ist das ein Hinweis auf ein Speicherleck. *Speicherplatz*
- Der Datendurchsatz misst die Menge an übertragenen Daten vom und zum Web-Server in Kilobyte pro Sekunde. Ist die Anzahl Kilobyte nicht mehr proportional zur Anzahl der Nutzer, hat das System seine Grenzen erreicht. *Datendurchsatz*

↓

4. Beispiel: Datendurchsatz

Eine typische Messung des Datendurchsatzes ist in Abb. 11.5 dargestellt. Sie zeigt, dass ab 25 Usern der Datendurchsatz im System rapide abnimmt.

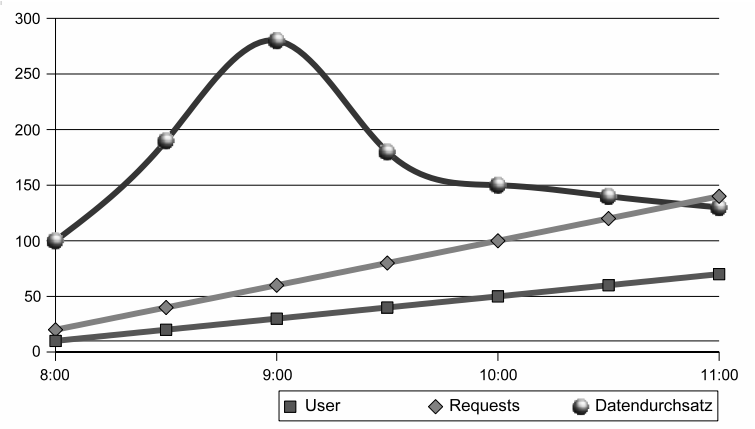


Abb. 11.5: Messung des Datendurchsatzes

Erkenntnis: In diesem Fall hat das Load Balancing[2] nicht richtig gearbeitet.

↑

[2] Load Balancing siehe auch Abschnitt 11.2.1

11.1.7 Planung der Performanz-/Lasttests

produktionsnah testen

Performanz-/Lasttests müssen in einer separaten, produktionsnahen Systemumgebung durchgeführt werden, damit die Testergebnisse nicht von äußeren Einflüssen verfälscht werden und wiederholte Testläufe vergleichbar sind.

Es ist zum Beispiel schon vorgekommen, dass eine Anwendung durch falsche Zuweisung von IP-Adressen nur im internen Netz getestet wurde. Die Firewall, die sich später nach der Freigabe des Systems als Schwachstelle herausstellte, war unerkannter Weise nicht in den Test mit einbezogen worden.

technische Ausstattung prüfen

Ein weiterer kritischer Punkt für den Performanztest ist die technische Ausstattung des Testnetzes. Es wird ein Netz von Test-Clients benötigt, auf denen virtuelle User ihre Tests (d.h. die Testskripte) ausführen. Jeder virtuelle User verbraucht je nach eingesetztem Testwerkzeug nicht wenig Speicherplatz. Hier muss bei Auswahl des Load Test Tools genau geprüft werden, ob die zur Verfügung stehende Kapazität ausreicht.

Dazu ein einfaches Beispiel: Wenn 5.000 User simuliert werden sollen und jeder virtuelle User 10 MB Hauptspeicher benötigt, werden mindestens 50 Rechner mit 1 GB großem Hauptspeicher benötigt. Der Performanztest muss also frühzeitig geplant werden. Unter Umständen kann es sich rechnen, für die Dauer des Performanztests eine Test-Server-Farm anzumieten.

Die bisherigen Ausführungen zu den Performanz-/Lasttests münden in einer Checkliste, die zur Planung und zur Durchführung der Tests zu Rate gezogen werden sollte:

✓ Checkliste Planung Performanz-/Lasttests

Testziele und -szenarien

- Sind die Testziele definiert (Performanz, Last, Volumen, Speicherleck)?
- Sind die Qualitätsanforderungen definiert (durchschnittliche Transaktionszeit, maximale Download-Zeit, durchschnittliche Prozessorauslastung,...)?
- Sind die Testszenarien und Testskripte den Testzielen entsprechend festgelegt?
- Beruhen die Testszenarien auf realistischen Geschäftsprozessen?
- Sind die trotz Automatisierung notwendigen manuellen Testschritte festgelegt?

- Gibt es eine konkrete Planung für die Testdurchführung, d.h. ein detailliertes „Testdrehbuch“?

Technische Vorraussetzungen

- Ist die Plattform der zu messenden Hardware bekannt?
- Sind alle technischen Komponenten der Testumgebung bekannt?
- Sind alle technischen Daten der Netzwerkumgebung bekannt?
- Sind alle Adressen der Netzwerkumgebung bekannt?
- Ist die Netzwerkkonfiguration für die geplanten Tests ausreichend?
- Spiegelt die Testumgebung die Produktionsumgebung hinreichend wieder oder besser: wird in einer Kopie der Produktionsumgebung getestet?
- Ist die Hardware-Ausstattung der Testrechner (Server, Clients) für die Tests ausreichend?
- Können die Testziele mit dem beschafften/existierenden Load Test Tool erreicht werden (Plattform- und Netzwerkkompatibilität, Speicherverbrauch)?
- Sind die Messpunkte definiert, um die geforderten Messwerte zu erhalten?
- Sind die notwendigen Monitore vorhanden, um alle definierten Messpunkte abgreifen zu können?
- Sind die Monitore richtig installiert und parametrisiert?

Testdurchführung

- Hat die Anwendung vor dem Performanztest alle anderen Prüfungen und Web-Tests erfolgreich bestanden?
- Stehen die benötigten Testdaten zur Verfügung?
- Sind alle für die Tests notwendigen Berechtigungen eingerichtet?
- Ist die exklusive Verfügbarkeit der Testumgebung über die gesamte Testdauer sichergestellt?
- Ist sichergestellt, dass die Testdurchführung nicht durch Fremdeinwirkung verfälscht wird?
- Sind länder- und zeitzonenübergreifende Performanz- und Lasttests geplant?

11.1.8 Werkzeuge für Performanz-/Lasttests

Load Test Tools, Monitore

Die Arbeitsweise von Lasttestwerkzeugen (Load Test Tools) und Monitoren ist bereits im Abschnitt 11.1.5 beschrieben.

Es gibt eine ganze Reihe von kommerziellen Lasttestwerkzeugen ([URL: TestToolsFAQ] Rubrik „Load and Performance“, [URL: TestToolsOPENSOURCE] Rubrik „Testing tools - Performance testing“, [URL: TestToolsSQA] Rubrik „Load and Performance Test Tools“). Einige Tools werden im Folgenden vorgestellt.

Mercury LoadRunner, QALoad

Zwei oft eingesetzte Load Test Tools sind Mercury LoadRunner von der Mercury Interactive ([URL: Mercury]) und QALoad von Compuware ([URL: Compuware]). Je nach Anzahl der benötigten virtuellen User und erforderlichen Monitore sind die Kosten für kommerzielle Lasttestwerkzeuge nicht unerheblich.

Frei verfügbare Tools können je nach Zielsetzung und Wichtigkeit der Web-Applikation eine Alternative sein, auch wenn sie nicht immer einen so großen Funktionsumfang besitzen und so komfortabel sind wie die kommerziellen Tools. WAS und JMeter sind zwei frei verfügbare Werkzeuge.

Web Application Stress Tool

Das Web Application Stress Tool von Microsoft ([URL: WAS]) ist Freeware und ermöglicht dem Tester mit seiner recht einfachen Bedienung, viele Requests an einen Server zu simulieren. Die Auswertungsmöglichkeiten sind für die Microsoft-Systemlandschaft besonders umfangreich.

JMeter

Das von der Apache Jakarta Group entwickelte JMeter ([URL: JMeter]) ist in Java geschrieben und steht als Open Source unter Apache Lizenz. JMeter ist besonders für die Performanzmessung von HTTP-Servern geeignet. Übertragene Texte können auch inhaltlich überprüft werden.

Mit sogenannten Samplern wird ein Server unter Last gesetzt. Mit einem HTTP Request wird zum Beispiel ein Web-Server unter Last gesetzt, mit einem FTP Request ein FTP-Server und mit einem JDBC Request eine Datenbank. Unter anderem stehen folgende Sampler zur Verfügung:

- FTP Request
- HTTP Request
- JDBC Request
- Java Request

- SOAP/XML-RPC Request
- LDAP Request
- WebService (SOAP) Request
- JUnit Sampler

Profiler

Sollen schon im Rahmen des Komponententests für einzelne Module Aussagen zur Performanz getroffen werden oder müssen auf Grund der Testergebnisse der Performanz-/Lasttests einzelne Module untersucht werden, kommen Profiler zum Einsatz. Sie prüfen Zeit- und Speicherplatzverbrauch auf Code-Ebene, was mit Werkzeugen wie JMeter nicht möglich ist. Java Profiler sind zum Beispiel unter [URL: TestToolsJava] in der Rubrik „Profilers“ zu finden.

11.1.9 Qualitätsanforderungen zu Performanz-/Lasttests

Leider geben Auftraggeber von Software-Anwendungen nicht immer konkrete und messbare Werte zum Qualitätsmerkmal Effizienz vor. Die Folge sind undankbare Diskussionen bei der Abnahme, ob das Zeitverhalten der Funktionen zumutbar oder inakzeptabel ist. Um das zu vermeiden, muss der Auftraggeber im Rahmen der Spezifikation der nicht funktionalen Anforderungen einer Web-Applikation konkrete Vorgaben zu den im Abschnitt 11.1.6 beschriebenen Metriken machen, wie zum Beispiel:

- Die Transaktion zur verbindlichen Bestätigung einer Finanzierungsanfrage muss in 5 Sekunden abgeschlossen sein.
- Der Download eines 500 KB großen PDF-Dokuments muss bei einer 128-KB ISDN-Verbindung in 7 Sekunden abgeschlossen sein.
- Bei Spitzen von 300 gleichzeitig angemeldeten Benutzern darf sich die Performanz des Systems höchstens um 20% verschlechtern (Antwortzeiten, Downloadzeiten,...). Dabei wird von einer maximalen Anzahl von 150 gleichzeitig angemeldeten Benutzern ausgegangen.
- Der Speicherlecktest von 5 Tagen Dauer zeigt keine Verschlechterung in der durchschnittlichen Transaktionsdauer. Vorrausetzung ist, dass folgende Szenarien über den Tag verteilt durchgeführt werden: 2000 Benutzer suchen nach Informatio-

nen, 500 laden Dokumente herunter, 200 fordern Informationsmaterial an, 50 berechnen Leasingraten und 10 schließen online einen verbindlichen Leasingvertrag ab.

- Auf dem Client muss das Laden eines Java-Applets innerhalb von 6 Sekunden abgeschlossen sein.

Systembedingungen festlegen

Natürlich müssen die konkreten Systembedingungen, unter denen diese Qualitätsanforderungen nachgewiesen werden sollen, festgelegt werden. Dazu gehören zum Beispiel die Prozessorleistungen der Rechner (Clients und Server), die Anzahl der parallel geschalteten Prozessoren oder die Bandbreiten zur Datenübertragung.

11.1.10 Empfehlungen zu Performanz-/Lasttests

Performanztests in allen Testphasen

Die Performanz sollte schon bei der Entwicklung und Integration der einzelnen Komponenten eine Rolle spielen. Wenn möglich sollten in den Testphasen Komponenten- und Integrationstest erste Aussagen zum Laufzeitverhalten gemacht werden, um die Ergebnisse auf das Gesamtsystem hochrechnen zu können. Dann können Performanzprobleme, die auf einer nicht optimalen Programmierung einzelner Komponenten beruhen, frühzeitig beseitigt werden.

Messungen, die endgültige Aussagen über das Verhalten des Gesamtssystems erlauben, können sicherlich erst in den späteren Teststufen Systemtest und Abnahmetest durchgeführt werden. Im laufenden Betrieb muss die Performanz permanent überwacht werden, um frühzeitig einen Leistungsabfall des Systems zu erkennen.

rechtzeitig Skalierbarkeit prüfen

Wenn ein System schlecht skaliert oder die Performanz eines Systems unzureichend ist, ist der Zukauf von Prozessorleistung ein beliebter Lösungsversuch, der in der Regel jedoch nur begrenzten Erfolg verspricht. Ein Skalierbarkeitstest in einer frühen Testphase schützt vor unliebsamen Überraschungen, wenn das System aufgerüstet werden muss.

Testszenarien aus Usability-Test verwenden

Wenn ein Usability-Test für die Web-Applikation geplant ist, können die dafür entworfenen Testszenarien auch für den Performanztest verwendet werden, wie im Beispiel des Finanzzierungsrechners mit den nach Nutzergruppen unterschiedenen Aufgabenstellungen für Kunden, Interessenten und Informationssuchende (siehe Abschnitt. 9.4.1).

länderübergreifend testen

Wenn Web-Anwendungen global eingesetzt werden, ist es wichtig, länder- und zeitzonenübergreifend Performanz- und Lasttests durchzuführen. Durch Benutzer in unterschiedlichen Zeitzonen können über den ganzen Tag hinweg „Lastwellen“ auf die Anwendung „zurollen“, die vom System aufgefangen werden müssen.

11.2 Ausfallsicherheitstest

Zur Zuverlässigkeit eines Systems gehört, dass es gegen Ausfall gesichert ist. Sollte das System trotz aller Vorsorgemaßnahmen doch ausfallen, muss gewährleistet sein, dass keine Daten verloren gehen und das System wieder in einem wohldefinierten Zustand gestartet werden kann.

11.2.1 Redundanzen und Failover-Verfahren

Ausfallsicherheit wird durch den Einsatz von Redundanzen erhöht. Dazu werden Systemkomponenten doppelt oder dreifach ausgelegt, damit im Notfall ein Failover stattfinden kann.

Failover

Bei einem Failover übernimmt eine Backup-Komponente ohne menschliches Eingreifen die Aufgaben einer fehlerhaft arbeitenden oder ausgefallenen Komponente. Ein Failover ist der automatische und ungeplante Wechsel von einer primären Systemkomponente auf eine redundante, sekundäre Systemkomponente.

Bei den betroffenen Systemkomponenten eines Failover kann es sich um die Anwendung, einen Teil der Anwendung, die Datenbank, um Hardware (Web-Server, Mail-Server, Load Balancer,...), die Firewall, das Netzwerk, das komplette System oder sogar das gesamte Rechenzentrum handeln.

Zum Ausfallsicherheitstest gehört daher auch der Test, dass bei einem Stromausfall die unterbrechungsfreie Stromversorgung gewährleistet ist, d.h. ein Failover durch die Notstromversorgungseinheiten stattfindet.

Zwei Beispiele für die redundante Auslegung von Systemkomponenten sind im Folgenden beschrieben.

1. Beispiel - Redundante Web-Server:

Zwei Server überwachen sich gegenseitig über eine Netzwerkverbindung, indem sie Signale austauschen. Antwortet ein Server

nicht, weil er ausgefallen ist, so startet der andere Server automatisch die Übernahme der Dienste.

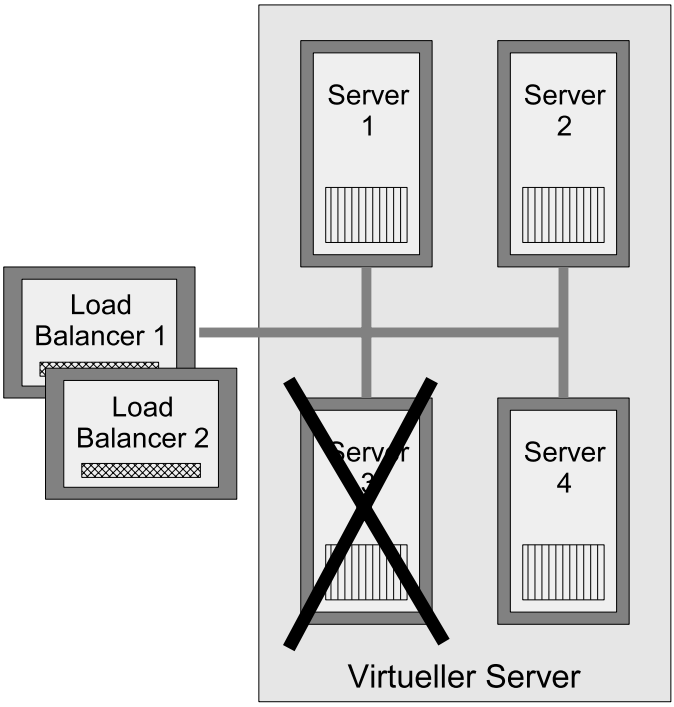
2. Beispiel Redundante Load Balancer:

Load Balancer

Ein Load Balancer[GL] verteilt die anfallende Last auf mehrere Web-Server, die zu einem virtuellen Server (Clustered Server[GL]) zusammengeschlossen sind. Fällt einer der Web-Server aus, so darf die Anwendung nicht oder zumindest nur unwesentlich davon beeinträchtigt werden.

Testfälle, um die Funktionalität eines Load Balancers zu testen, sind zum Beispiel das Stoppen der Web-Dienste oder das ziehen des Steckers eines Servers aus dem Pool des virtuellen Servers (Abb. 11.6).

Abb. 11.6: Test Load Balancer



Aber was passiert, wenn der Load-Balancer ausfällt? Auch in diesem Fall muss das System stabil bleiben. Zum Beispiel übernimmt ein zweiter, bereit stehender Load Balancer in einem Failover automatisch die Aufgaben des ausgefallenen.

Failover-Verfahren sind Bestandteile der Ausfallsicherheitstests.

Der Ausfallsicherheitstest prüft die Funktionalität und Korrektheit der Failover-Prozesse. Der Ausfallsicherheitstest besteht aus Failover-Test, Failback-Test und Restart-/Recovery-Test.

Ausfallsicherheitstest

11.2.2 Failover-Test

Der Failover-Test prüft das rechtzeitige Zuschalten von Ressourcen beim Ausfall einer Systemkomponente und die Funktionsfähigkeit des Systems nach einem Failover.

Failover-Test

Für den Failover-Test werden die für den Betrieb der Anwendung kritischen Hard- und Softwarekomponenten bestimmt. Anschließend wird in Testfällen beschrieben, wie der Ausfall der einzelnen Systemkomponenten herbeigeführt wird und wie das System darauf zu reagieren hat.

Zur Testdurchführung werden die automatisierten Testszenarien aus dem Performanz-/Lasttest abgespielt. Zeitgleich wird nacheinander jede der zuvor definierten Fehlersituationen herbeigeführt. Das System wird über einen längeren Zeitraum mit den im Performanztest beschriebenen Werkzeugen beobachtet und ausgewertet.

Eine Verschärfung des Tests besteht darin, Fehlersituationen gleichzeitig hervorzurufen, also den gleichzeitigen Ausfall von mehreren Komponenten herbeizuführen.

11.2.3 Failback-Test

Nach dem provozierten Ausfall einer Systemkomponente (Failover) wird die Fehlersituation wieder rückgängig gemacht (Failback) und der ursprüngliche Systemzustand wieder hergestellt.

Failback

Wie bei einem Failover wird das System nach einem Failback mit den entsprechenden Werkzeugen beobachtet und darauf hin überprüft, ob es wieder die Leistungswerte wie vor dem Failover erreicht.

Der Failback-Test prüft das korrekte Wiederzuschalten einer zuvor ausgefallenen Systemkomponente.

Failback-Test

11.2.4 Restart-/Recovery-Test

Wenn trotz aller Sicherheitsvorkehrungen eine Komponente oder gar das gesamte System ausfallen sollte, ohne dass ein Failover stattgefunden hat, muss durch Restart-Recovery-Verfahren gewährleistet sein, dass keine Daten verloren gehen. Datenbanken und Dateisysteme müssen sich nach einem Notfall in einem konsistenten Zustand befinden oder zumindest wieder in einen solchen versetzt werden können.

Restart-/ Recovery-Test

Der Restart-/Recovery-Test (Wiederherstellbarkeitstest) prüft die Wiederherstellbarkeit eines Systems nach seinem Ausfall. Er stellt die Funktionstüchtigkeit und Korrektheit der festgelegten Restart-/Recovery-Verfahren sicher.

Datenkonsistenz

Nach dem Restart des Systems und der Wiederherstellung des ursprünglichen Datenbestandes (Recovery) muss der Benutzer seine Transaktionen mit der Web-Applikation wieder problemlos aufnehmen können. Daten dürfen nicht verloren gegangen, nicht mehrfach angelegt oder fälschlicher Weise verändert worden sein. Falls eingegebene Daten nicht abschließend bearbeitet oder gesichert werden konnten, ist der Benutzer davon zu unterrichten und wieder aufgenommene Funktionen müssen auf einem definierten und konsistenten Datenbestand aufsetzen.

Der Ausfall von Systemkomponenten muss sowohl auf der Server-Seite als auch auf der Client-Seite getestet werden. Was passiert zum Beispiel, wenn ein Benutzer seinen Client-PC abschaltet? Auch dann dürfen keine Daten verloren gehen und die Datenbestände auf Client und Servern müssen nach einem solchen Ausnahmefall in einem konsistenten Zustand sein. Transaktionen müssen korrekt beendet werden, damit sie nicht von unbefugten Dritten aufgenommen werden können. Diese Thematik wird auch im Sicherheitstest (Kap. 8.8) behandelt.

Die folgende Checkliste fasst die Aktivitäten zum Ausfallsicherheitstest zusammen:

✓ Checkliste Ausfallsicherheitstest

- Wurde eine Ausfallrisikoanalyse durchgeführt?
- Sind darin alle Systemkomponenten berücksichtigt?
- Sind alle kritischen Komponenten doppelt ausgelegt?

- Ist der Ausfall jeder Komponente in mindestens einem Testfall vorgesehen?
- Sind die Failover- und Failback-Zeiten definiert und in die Testerwartungen eingeflossen?
- Ist der Failover-Test erfolgreich durchgeführt und dokumentiert worden?
- Wurde im Testbetrieb die geforderte Verfügbarkeit[3] nachgewiesen?
- Ist der Failback-Test erfolgreich durchgeführt und dokumentiert worden?
- Wurden im Testbetrieb die geforderten Failback-Zeiten nachgewiesen?
- Konnten die vorgegebenen Instandsetzungszeiten im Testbetrieb eingehalten werden?
- Sind Restart-Recovery-Verfahren definiert und geprüft?
- Ist der Restart-/Recovery-Test erfolgreich durchgeführt und dokumentiert worden?

Tipp

Der Wiederherstellbarkeitstest ist in jedem Fall im Rahmen des Ausfallsicherheitstests durchzuführen, auch wenn keine Failover-Verfahren für das System implementiert sind.

11.2.5 Werkzeuge für den Ausfallsicherheitstest

Der Ausfallsicherheitstest wird mit den Szenarien des Performanz-/Lasttests durchgeführt. Das bedeutet, dass auch hier die im Abschnitt 11.1.5 beschriebenen Lasttestwerkzeuge und Monitore zum Einsatz kommen. Zum einen, um die Geschäftsprozesse zu simulieren, zu denen parallel ein Failover oder Failback getestet wird, und zum anderen, um die Messungen und Auswertungen des Systemverhaltens während der Notfallsituationen vorzunehmen.

[3] Verfügbarkeit ist im Abschnitt 11.3.1 definiert.

11.2.6 Qualitätsanforderungen zum Ausfallsicherheitstest

Neben der Erfüllung der Checkliste „Ausfallsicherheitstest“ können konkrete Anforderungen an die Failover- und Instandsetzungszeiten wie in den folgenden Beispielen gestellt werden:

Failover-Zeit

- Die Backup-Firewall muss innerhalb von 5 Sekunden die Aufgaben der ausgefallenen Firewall übernehmen.

Instandsetzungszeit

- Ist die Anwendung aufgrund eines Systemausfalles nicht verfügbar, muss das System innerhalb von einer Stunde instand gesetzt werden können und die Anwendung wieder zur Verfügung stehen.

11.2.7 Empfehlungen zum Ausfallsicherheitstest

Ausfallrisikoanalyse durchführen

Um ein System ausfallsicher zu machen, können alle Hardwarekomponenten doppelt oder dreifach ausgelegt werden. Bei dieser Lösung ist natürlich die Bedeutsamkeit der Anwendung den entstehenden Kosten gegenüberzustellen. Daher ist es ratsam, eine Risikoanalyse (Kap. 5) durchzuführen, in der bewertet wird, welches Risiko der Ausfall einer Systemkomponente bzw. des Systems bedeutet. Daraus werden die entsprechenden Maßnahmen zur Ausfallsicherheit abgeleitet. Ein reines Informationssystem darf sicherlich einige Zeit nicht erreichbar sein, sofern der Benutzer entsprechende Meldungen erhält. Ein Onlineshop, der plötzlich nicht verfügbar ist, kann fatale Auswirkungen für ein Unternehmen haben.

Ausfallsicherheitstest zum System- und Abnahmetest

Die Messung und die Bewertung der Failover- und Instandsetzungszeiten sollten möglichst früh während des Ausfallsicherheitstests in den Teststufen System- und Abnahmetest begonnen werden, denn so kann rechtzeitig abgeschätzt werden, ob die geforderten Werte in Produktion theoretisch erreicht werden können. Das ist ein weiteres Argument dafür, Abnahmetests[4] in einer möglichst produktionsidentischen Systemumgebung durchzuführen.

[4] siehe Kap. 14.5, Teststufe Abnahmetest

11.3 Verfügbarkeitstest

Zuverlässigkeit

Zuverlässigkeit (Reliability) ist die Fähigkeit eines Systems, die verlangte Funktionalität unter gegebenen Randbedingungen für eine gegebene Zeit zu erfüllen, also in dieser Zeit ohne Ausfälle zu funktionieren. Um dieses Qualitätsmerkmal messen zu können, wird die Verfügbarkeit definiert.

11.3.1 Definitionen zur Verfügbarkeit

Die DIN 40041 ([DIN 40041]) beschreibt Verfügbarkeit als die Wahrscheinlichkeit, ein System zu einem gegebenen Zeitpunkt in einem funktionsfähigen Zustand anzutreffen. Verfügbarkeit ist demnach das Verhältnis der mittleren fehlerfreien Zeit während des Betrachtungszeitraums zum gesamten Betrachtungszeitraum.

MTBF - Mean Time Between Failure

Die mittlere fehlerfreie Zeit während des Betrachtungszeitraums wird als MTBF (Mean Time Between Failure) bezeichnet. Dieser Begriff ist sinnvoll, wenn man davon ausgeht, dass ein auftretender Fehler behoben wird und das System nicht für den Rest des Betrachtungszeitraums in dem fehlerhaften Zustand verbleibt.

MTTR - Mean Time To Repair

Die Zeit vom Eintritt des fehlerhaften Zustandes bis zur Wiederherstellung des fehlerfreien Zustandes wird als MTTR (Mean Time To Repair) bezeichnet.

Wenn also jeder auftretende Fehler stets behoben wird, ist der Betrachtungszeitraum die Summe aus MTBF und MTTR. Verfügbarkeit lässt sich somit darstellen als:

Verfügbarkeit

$$Verfügbarkeit = MTBF/(MTBF+MTTR)$$

Als Zahl ausgedrückt, liegt die Verfügbarkeit zwischen 0 und 1. Dementsprechend wird die Ausfallrate eines Systems (in der DIN 40041 auch Unverfügbarkeit genannt) als Einerkomplement der Verfügbarkeit dargestellt:

Ausfallrate

$$Ausfallrate = 1- Verfügbarkeit$$

Um eine hohe Verfügbarkeit eines Anwendungssystems zu erhalten, muss die MTTR niedrig gehalten werden, d.h. gegen 0 gehen.

Wartbarkeit

Die MTTR einer Software ist somit auch ein Maß für die Wartbarkeit eines Systems, denn nur ein gut wartbares System erlaubt es, Fehler effizient zu lokalisieren und zu beheben.

Verfügbarkeitstest

Der Verfügbarkeitstest prüft die Verfügbarkeit bzw. Ausfallrate eines Systems.

11.3.2 Qualitätsanforderungen und Empfehlungen zum Verfügbarkeitstest

Verfügbarkeit festlegen

Die Verfügbarkeit des Systems wird in Prozent der Betriebszeit angegeben. Zum Beispiel kann eine 99,5-prozentige Verfügbarkeit vereinbart werden. Von 100 Tagen darf das System also einen halben Tag ausfallen.

Verfügbarkeit im Abnahmetest messen

Die Messung der Verfügbarkeit ist für den Testbetrieb nur bedingt tauglich, da die tatsächliche Verfügbarkeit eines Systems erst über einen längeren Zeitraum im produktiven Betrieb gemessen werden kann. Aber schon während des Abnahmetests (Kap. 14.5) und eines Pilotbetriebes (Abschnitt 14.6.1) sollten Verfügbarkeit und Ausfallrate vorgegeben, gemessen und für die Produktion hochgerechnet werden.

11.4 Zusammenfassung

- Die Performanz-/Lasttests stellen fest, ob unter normalen Bedingungen Antwortzeiten und Ressourcenverbrauch zufriedenstellend sind und sich unter Last nicht maßgeblich verschlechtern. Sie prüfen die Effizienz eines Systems.
- Zu den Performanz-/Lasttests zählen Performanztest, Lasttest, Skalierbarkeitstest und Speicherlecktest.
- Der Performanztest prüft das Zeitverhalten und den Ressourcenverbrauch eines Anwendungssystems.
- Der Lasttest findet heraus, wann ein System zusammenbricht.
- Skalierbarkeit beschreibt die Ausbaufähigkeit von Software-Produkten, die bei steigender Last oder steigendem Ressourcenbedarf notwendig werden kann.

- Der Skalierbarkeitstest stellt sicher, dass steigende (oder fallende) Anforderungen an eine Software allein durch Hardware-Anpassungen erfüllt werden können.
- Der Speicherlecktest ist ein Performanztest auf Zeit und sucht nach Speicherproblemen.
- Performanz-/Lasttests werden in einer produktionsnahen Systemumgebung mit Load Test Tools durchgeführt. Schwachstellen werden mit Monitoren geortet.
- Redundante Systemkomponenten und Failover- und Failback-Maßnahmen erhöhen die Ausfallsicherheit eines Systems.
- Der Ausfallsicherheitstest testet die Zuverlässigkeit eines Systems und besteht aus Failover-, Failback- und Restart-/Recovery-Test.
- Der Restart-/Recovery-Test stellt sicher, dass die Datenbasis auch nach Systemstörungen in einem konsistenten Zustand ist.
- Verfügbarkeit ist der Grad, zu dem eine Komponente oder ein System für die Nutzung zur Verfügung gestanden hat. Sie ist ein Maß der Zuverlässigkeit und wird im Verfügbarkeitstest überprüft.

Teil III
Testmanagement

12. Testwiederholungen
13. Planung der Testtypen
14. Planung der Teststufen

Im dritten Teil des Buches werden die Testprozesse aus Sicht des Testmanagers beschrieben, der die notwendigen Maßnahmen zur Qualitätssicherung von Web-Applikationen planen und steuern muss.

Nach Änderungen in der Software oder am Systemumfeld müssen Tests wiederholt werden. Die bei der Testplanung zu berücksichtigenden Möglichkeiten und Aspekte sind Thema des Kapitels Testwiederholungen.

Im Testplan eines Projektes wird festgelegt, welche Testtypen mit welcher Intensität von welchen Personen durchgeführt und welche Testmittel dafür eingesetzt werden sollen. Die Kriterien für die zu treffenden Entscheidungen werden im Kapitel zur Planung der Testtypen erläutert.

Im letzten Kapitel – Planung der Teststufen – werden die Teststufen beschrieben, die ein Software-Entwicklungsprojekt durchläuft. Darin ist auch die Zuordnung der im Teil II des Buches beschriebenen Testtypen zu den einzelnen Teststufen vorgenommen. Sie ist die Basis für die Planung der Tests eines Software-Anwendungssystems.

12 Testwiederholungen

„Die Reparatur alter Fehler kostet oft mehr als die Anschaffung neuer.“

Wieslaw Brudzinski (*1920)

Wiederholungen von bereits durchgeführten Tests sind notwendige Tätigkeiten, deren Aufwand bei der Testplanung nicht unterschätzt werden darf.

Die drei Arten der Testwiederholung (Abb. 12.1) – Fehlernachtest, Regressionstest, Wartungstest – sind keine Testtypen im Sinne der im Teil II des Buches beschriebenen Tests, denn sie stellen „nur“ indirekt die Qualitätsmerkmale einer Software sicher.

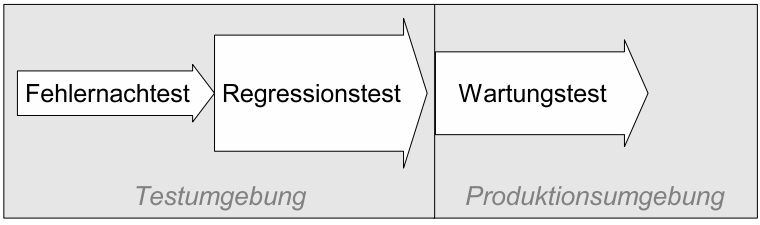


Abb. 12.1: Testwiederholungen

In der Testumgebung wird nach einem Fehlernachtest in der Regel ein umfangreicherer Regressionstest durchgeführt. Der kann wiederum einen Wartungstest in der Produktionsumgebung zur Folge haben.

Diese Tests, die nach Änderungen in einem Softwaresystem durchgeführt werden, wiederholen bereits durchgeführte Tests

und prüfen, ob Fehler korrekt beseitigt wurden und keine ungeplanten Abweichungen gegenüber den vorhergehenden Testläufen auftreten.

12.1 Fehlernachtest

Fehlernachtest

> Mit einem Fehlernachtest wird geprüft, ob ein Fehlerzustand korrekt beseitigt wurde.

Beim Fehlernachtest werden nur diejenigen Testfälle wiederholt, die eine Fehlerwirkung aufgedeckt haben und deren Fehlerzustand inzwischen behoben wurde.

Ob einem Fehlernachtest ein umfassenderer Regressionstest (s. nächsten Abschnitt) folgen muss, ist von der Bedeutung des behobenen Fehlers abhängig und muss von Fall zu Fall vom Testmanagement im Rahmen der Teststeuerung entschieden werden.

Wurde zum Beispiel auf der Benutzeroberfläche eine Zahl mit einer statt mit zwei Nachkommastellen dargestellt, genügt ein lokaler Fehlernachtest. Wurde die Zahl dagegen auf Grund einer fehlerhaft programmierten Formel falsch berechnet, dann ist, wenn die Formel in anderen Programmteilen der Anwendung benutzt wird, ein entsprechender Regressionstest notwendig.

12.2 Regressionstest

Regressionstest

> Der Regressionstest ist die Wiederholung der Tests zu einer bereits getesteten Komponente nach einer Modifikation. Er hat das Ziel sicherzustellen, dass durch die vorgenommenen Änderungen keine neuen Fehler entstanden sind und dass bisher maskierte Fehlerwirkungen aufgedeckt werden.

Für den Regressionstest werden keine besonderen Testfälle entwickelt, sondern bestehende Testfälle benutzt. Der Regressionstest untersucht die Abweichungen der Testergebnisse im Vergleich zum Testlauf mit der Vorgängerversion des Testobjektes.

Je nachdem, welche Systemkomponente in welcher Projektphase modifiziert wurde, kann ein Regressionstest in jeder Teststufe (s. Kap. 14) notwendig sein. Drei Ereignisse können einen Regressionstest erforderlich machen:

1. Bekannte Fehler wurden behoben.
2. Neue Funktionen wurden realisiert oder bestehende wurden geändert.
3. Die Systemlandschaft wurde verändert.

12.2.1 Regressionstests nach Fehlerbehebung

Bei der Korrektur von Fehlerzuständen kann es vorkommen, dass neue entstehen. Zudem können vorhandene Fehlerzustände bisher nicht gefunden worden sein, weil die Fehlerwirkungen von anderen überdeckt (maskiert) worden sind. In einem Regressionstest wird geprüft, ob die Anwendung nicht „verschlimmbessert" wurde und ob bisher maskierte Fehlerwirkungen auftreten.

Die Testwiederholung auf Grund eines behobenen Fehlers beschränkt sich nicht nur auf die Komponente, in welcher der Fehler korrigiert wurde. Zumindest sind die mit der modifizierten Komponente direkt kommunizierenden Systemkomponenten zu überprüfen, d.h. der Integrationstest muss für diesen Bereich wiederholt werden.

Soll nur ein begrenzter Regressionstest durchgeführt werden, muss das Testmanagement das Risiko gegen den Aufwand eines kompletten Regressionstests abwägen und eine begründete Entscheidung treffen, warum ein kompletter Regressionstest nicht notwendig ist.

12.2.2 Regressionstests nach Funktionsveränderungen

Wenn eine Web-Applikation um neue Funktionen erweitert oder wenn bestehende Funktionen verändert werden, müssen diese neuen bzw. geänderten Funktionalitäten nach den in den vorhergehenden Kapiteln beschriebenen Testverfahren einzeln getestet werden.

Anschließend muss ein umfangreicher Regressionstest sicherstellen, dass keine der bestehenden, unveränderten Funktionen durch die Neuerungen ungewollt beeinflusst werden. Der Regressionstest nach funktionalen Änderungen erstreckt sich in der Regel über alle Testphasen, bis hin zum erneuten Abnahmetest.

12.2.3 Regressionstests nach Systemveränderungen

Wenn sich die Systemkonfiguration oder das technische Umfeld, in der eine Anwendung betrieben wird, ändert, ist für bestimmte Bereiche des Systems eine Testwiederholung fällig, wie einige Beispiele deutlich machen:

- Wird die Anwendung auf eine neue Datenbank umgestellt, müssen Tests für die Komponenten wiederholt werden, die auf die Datenbank zugreifen. Performanz-, Last- und Restart-/Recovery-Tests müssen erneut durchgeführt werden.
- Wird eine neue Hardware-Generation der Server installiert, müssen alle Performanz-/Lasttests wiederholt werden.
- Wird die bestehende Client-Server-Kommunikation auf das neue Konzept AJaX[GL] (Asynchronous JavaScript and XML) umgestellt, müssen alle Funktionstests überarbeitet und die Usability neu bewertet werden. Denn mit AJaX wird die Benutzerfreundlichkeit verbessert, weil der Benutzer Aktionen auf der Website ausführen kann, während die Daten vom Server asynchron nachgeladen werden.
- Kommt ein neuer Browser oder eine neue Version eines bestehenden Browsers auf den Markt, so müssen Browser- , Plugin- und Sicherheitstests wiederholt werden.
- Soll die Anwendung unter einem neuen PC-Betriebssystem eingesetzt werden, sind zumindest die Tests zu wiederholen, die sich auf den Client beziehen (Browser-Test, Plugin-Test, Sicherheitstest und ggf. Installationstest).

↓

Beispiel: Regression des Plugin-Tests

Die Notwendigkeit des Regressionstests wird an dem Beispiel in Kap. 8.7, „Eine wahre Plugin-Testgeschichte“, deutlich. Wir erinnern uns, dass die Web-Anwendung ein Java-Plugin ab einer bestimmten Version benötigt und dass das Plugin inzwischen erfolgreich installiert und ohne Probleme im Einsatz ist. Die Testgeschichte nimmt nun ihren Lauf:

Szenario Plugin-Test - Teil 5

Ein neues Betriebssystem für Clients kommt auf den Markt. Der Regressionstest der Anwendung wird mit dem neuen Betriebssystem durchgeführt. Das Ergebnis zeigt, dass unser Java-Applet mit dem Java-Plugin 1.4.1 von diesem neuen Betriebssystem nicht geladen wird.

Warum? Die Kompatibilität ist seitens des Herstellers nicht gewährleistet. Erst ein Service Pack zum Plugin schafft Abhilfe. Dem Benutzer wird umgehend diese Information zur Verfügung gestellt und noch besser: Stellt die Anwendung beim Start fest, dass ein Benutzer das neue Betriebssystem ohne das notwendige Service Pack installiert hat, wird dieses automatisch aus dem Internet heruntergeladen und installiert. Das ist natürlich eine neue Anforderung an die Web-Anwendung, die spezifiziert, programmiert und getestet wird.

Szenario Plugin-Test - Teil 6

Der Fall geht noch weiter: Monate später gibt der Hersteller eine neue Version 2.0 des Plugins frei, die zu allen Vorgängerversionen des Plugins abwärtskompatibel sein soll.

Ha! Der Regressionstest zeigt, dass die Funktionen, die Swing-Klassen benutzen, nicht mehr korrekt arbeiten. Das Java-Applet der Web-Anwendung funktioniert nicht mehr richtig.[1]

↑

12.3 Wartungstest

Wartung

Unter Wartung versteht man die Änderung eines Software-Produkts nach seiner Auslieferung, um Fehler zu korrigieren, die Performanz oder andere Merkmale zu verbessern oder das Produkt für eine andere Umgebung zu adaptieren (nach [IEEE 1219]).

Das bedeutet, dass Änderungen an der Software oder im systemtechnischen Umfeld nicht nur durch den Regressionstest in der Testumgebung, sondern auch in der Produktionsumgebung überprüft werden müssen.

Wartungstest

> Der Wartungstest überprüft, ob Änderungen eines Software-Produkts in der produktiven Systemumgebung unerwartete Wirkungen oder Inkonsistenzen der Daten zur Folge haben.

Der Wartungstest wird umgehend nach der Freigabe einer neuen Systemkonfiguration anhand wohldefinierter Geschäftsprozesse durchgeführt, so dass im Fehlerfall noch kein Benutzer Gelegenheit hatte, auf diese Fehlerwirkung zu stoßen.

[1] Natürlich: Die alte Version des Plugins wird vom Hersteller nicht mehr unterstützt...

Produktionsumgebung spiegeln

Das Problem des Wartungstests ist der produktive Datenbestand. Sollten die Tests in der Produktionsumgebung Daten verfälschen, unbrauchbar machen oder die Produktion behindern, kann das schwerwiegende Folgen haben. Daher ist es – wenn man sich es leisten kann – sinnvoll, die Produktionsumgebung zu spiegeln und den Wartungstest auf der replizierten Umgebung durchzuführen. In dieser Umgebung kann die Abnahme der Änderungen durch den Auftraggeber stattfinden.

Testmandant

Wenn kein gespiegeltes Produktivsystem zur Verfügung steht, ist es ratsam, einen eigenständigen Testmandanten für Testzwecke im Produktionssystem zu führen.

Nummernkreis für Testdaten

Die minimale Lösung, den Wartungstest von den Produktivdaten zu trennen, besteht darin, einen abgegrenzten Nummernkreis für Testdaten festzulegen, auf die ein Benutzer mit normalen Rechten nicht zugreifen kann.

Rückfalllösung

Eine getestete Rückfalllösung, die eine alte Systemkonfiguration wieder herstellt, muss für den Fall existieren, dass im Wartungstest Fehler festgestellt werden, die einen produktiven Betrieb der neuen Systemversion nicht zulassen.

12.4 Werkzeuge für die Testwiederholung

Es liegt nahe, dass Tests nur effektiv und bezahlbar in größerem Umfang und mehrmals wiederholt werden können, wenn die Testläufe automatisiert sind. Zur automatisierten Testwiederholung werden Load Test Tools und Capture Replay Tools eingesetzt.

12.4.1 Load Test Tools (Lasttestwerkzeuge)

Load Test Tools (Lasttestwerkzeuge), mit denen Regressionstests zur Performanz und Ausfallsicherheit automatisiert werden, sind ausführlich im Abschnitt 11.1.8 beschrieben und werden daher an dieser Stelle nicht weiter erläutert.

12.4.2 Capture Replay Tools (Testroboter)

Tests, die über die Benutzeroberfläche der Anwendung durchgeführt werden, können mit Capture Replay Tools – auch Testroboter genannt – automatisiert werden.

Capture Replay Tools erkennen die Objekte auf der Benutzeroberfläche der zu testenden Anwendung (Texte, Buttons, Eingabefelder, Grafiken,...) und zeichnen bei der Testdurchführung die Eingaben, Mausklicks und Angaben des Benutzers auf. Der Tester legt bei der Aufzeichnung fest, welche Objekte, Eigenschaften der Objekte und Daten später bei der Wiederholung des Tests auf Veränderungen überprüft werden sollen.

Beim Regressionstest werden die aufgezeichneten und ggf. durch Programmierung erweiterten Testskripte automatisch vom Testwerkzeug abgespielt. Abweichungen gegenüber den vorangegangenen Aufzeichnungen werden protokolliert. Der Tester muss dann entscheiden, ob eine ausgewiesene Veränderung gewollt war oder nicht.

Die einzelnen Regressionstest-Tools haben unterschiedliche Fähigkeiten und Einsatzmöglichkeiten. Komfortable Werkzeuge erlauben es, mehrere Browser-Typen und unterschiedliche grafische Oberflächen mit denselben Testskripten zu testen. Testskripte können so programmiert werden, dass die Tests nicht komplett aufgezeichnet werden müssen, sondern parallel zur Programmierung erstellt werden können. Zudem können Datei- und Datenbankinhalte automatisiert verglichen werden.

WinRunner, Quicktest

Eines der wohl bekanntesten Werkzeuge ist der WinRunner von Mercury Interactive, dessen Nachfolgerwerkzeug der QuickTest Professional ist ([URL: Mercury]).

Abhängig von den Anforderungen, die an eine Web-Anwendung gestellt werden, können kostengünstige (wie z.B. JStudio SiteWalker) oder gar kostenlose Capture Replay Tools (wie z.B. Selenium) eine Alternative zu kommerziellen Testrobotern sein.

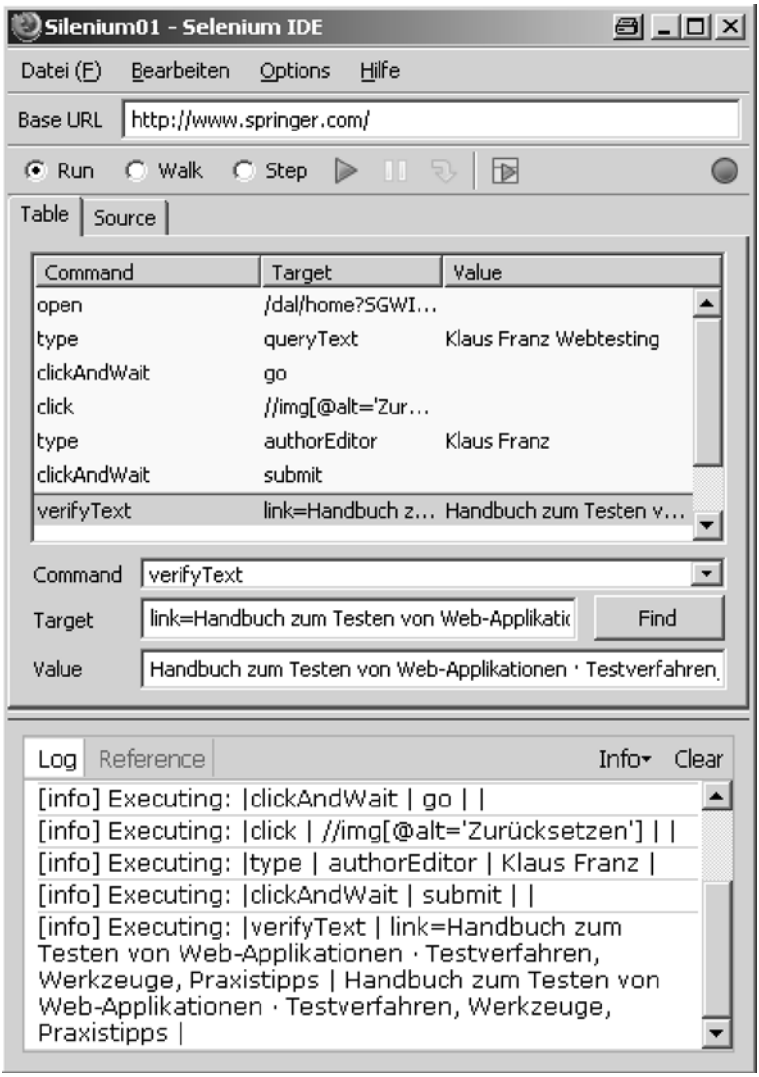
JStudio SiteWalker

„JStudio SiteWalker Professional“ ([URL: Sitewalker]) ist als kommerzielles Werkzeug speziell zur Automatisierung der Tests von Webseiten und Web-Applikationen unter dem MS Internet Explorer entwickelt worden. Der Einsatz dieses Tools bietet sich zum Beispiel an, wenn eine Anwendung im Intranet, in dem nur der MS Internet Explorer als Browser zugelassen ist, benutzt wird.

Selenium

Ein vielseitiges Open Source Tool unter Apache License zur Automatisierung von Funktions- und Regressionstests ist Selenium ([URL: Selenium]). Selenium besteht aus mehreren Werkzeugen. Eines davon ist Selenium-IDE, das eine komplette Testumgebung als Firefox-Plugin bereitstellt. Abb. 12.2 zeigt ein Beispielskript von Selenium-IDE.

Abb. 12.2: Selenium



In der Abb. 12.2 ist im Reiter „Table“ das aufgezeichnete Skript zu sehen und darunter der im Skript selektierte Befehl, der an dieser Stelle editiert werden kann. Im unteren Teil des Fensters

wird das Protokoll der letzten, automatisch ausgeführten Testwiederholung aufgeführt.

Übersichten von Capture Replay Tools

Umfassende Übersichten von Capture Replay Tools sind zum Beispiel unter [URL: TestToolsFAQ] Rubrik „GUI Test Drivers“, [URL: TestToolsOPENSOURCE] Rubrik „Functional testing“ oder [URL: TestToolsSQA] Rubrik „Web Functional/Regression Test Tools“ zu finden.

12.5 Empfehlungen zur Testwiederholung

Testwiederholungen planen

Will man nach einer Änderung auf der sicheren Seite sein, müssten alle bisher durchgeführten Tests wiederholt werden. Das ist natürlich nicht immer notwendig und auch nicht bezahlbar. Daher müssen für jede Veränderung einer Software die Testwiederholungen bewertet und neu geplant werden.

Capture Replay Tools für Komponententests

Capture Replay Tools ermöglichen schon in frühen Testphasen eine Automatisierung von Tests, in dem die entworfenen Testfälle in programmierten Testskripts abgebildet und ausgeführt werden. So können nicht nur die später notwendigen Regressionstests, sondern auch die ersten Testläufe für eine Komponente automatisch ablaufen.

Spezialisten für Testautomatisierung

In Projekten hat sich gezeigt, dass Testroboter zur Testautomatisierung nicht von jedem Tester im Testteam eingesetzt werden sollten, weil spezielle Programmierkenntnisse zum optimalen Einsatz dieser Tools notwendig sind. Auch wenn der Trend zum benutzerfreundlichen – oder besser gesagt zum testerfreundlichen – Aufzeichnen und Abspielen der Testszenarien geht, ist es in der Regel effektiver, wenn ein Spezialistenteam für die Testautomatisierung zuständig ist. Dieses Team übernimmt die Erstellung und Pflege aller Testskripte. Der Aufwand für die Anpassung von Testskripten durch Programmänderungen oder neue Versionen des eingesetzten Testtools ist weder von der Komplexität noch vom Umfang her zu unterschätzen.

Nicht alle Tests müssen automatisiert werden

Nicht alle Tests müssen unbedingt automatisiert werden. Weil die Ersterstellung und die Pflege von Testskripten aufwendig ist, muss bei der Testplanung genau überlegt werden, für welche Testszenarien sich ein automatisierter Regressionstest rechnet. Kandidaten für automatisierte Tests sind natürlich kritische Funktionen, die bei jeder Programmanpassung überprüft werden müssen. Dass sich Testautomatisierung auszahlen kann, zeigt das folgende Beispiel.

↓
Beispiel zur Testautomatisierung

Wie sich Testautomatisierung auszahlen kann

Ein Programm – es ist hoffentlich immer noch im Einsatz – berechnet anhand vorgegebener Formeln Parametereinstellungen für technische Messgeräte. Aufgrund der gültigen Äquivalenzklassen und deren möglichen Kombinationen ergeben sich cirka 400 Berechnungsmöglichkeiten.

Für die Durchführung der Tests war eine Reduzierung der zu testenden Kombinationen zu riskant, weil jede Kombination ein anderes Ergebnis liefert. Um die 400 Testszenarien effektiv wiederholen zu können, wurden die Tests automatisiert.

Folgendes Fazit kann gezogen werden: Die Testautomatisierung brachte erst einmal keine Zeitersparnis gegenüber einem manuellen Test, denn die im ersten Testlauf eingesparte Zeit war zuvor für die Programmierung und die Tests der Testskripte verbraucht worden.

Aber: Unter optimalen Bedingungen würde ein manueller Regressionstest über alle 400 Testfälle durch eine erfahrene Person vier Tage dauern. Ein automatisierter Testdurchlauf benötigt zwei Stunden!

Und allein im Laufe des Projektes wurde der Regressionstest fünfmal durchgeführt. Hinzu kamen (und kommen immer noch) die Testläufe im Rahmen von Wartungsarbeiten. Das hat sich gelohnt!

↑

12.6 Zusammenfassung

- Testwiederholungen sind notwendig, nachdem bekannte Fehler behoben, neue Funktionen realisiert, bestehende Funktionen geändert oder die Systemlandschaft verändert wurden.
- Der Fehlernachtest ist die Wiederholung eines Testfalls, der einen Fehler aufgedeckt hat.
- Der Regressionstest ist die umfassende Testwiederholung bereits durchgeführter Tests. Er stellt in der Testumgebung sicher, dass sich keine neuen Fehler in das System eingeschlichen haben.
- Der Wartungstest überprüft, ob geplante Änderungen keine unerwünschten Nebenwirkungen im produktiven System haben.
- Alte Systemkonfigurationen müssen wieder hergestellt werden können, falls im Wartungstest Fehler festgestellt werden, die

den produktiven Betrieb einer neuen Systemversion nicht zulassen.

- Bei der Testplanung muss festgelegt werden, welche Regressionstests automatisiert werden müssen und welche effizienter manuell durchgeführt werden können.
- Regressionstests, die sich auf die Oberfläche einer Web-Anwendung beziehen, werden mit Capture Replay Tools automatisiert. Dabei werden aufgezeichnete und durch Programmierung erweiterte Testskripte wiederholt abgespielt.
- Für die automatische Wiederholung von Performanz-/Lasttests werden Load Test Tools eingesetzt.
- Für die Automatisierung von Regressionstests werden Testtool-Spezialisten benötigt.

13 Planung der Testtypen

„Jede Dummheit findet einen, der sie macht."

Tennessee Williams (1911 – 1983)

Um die Tests einer Web-Anwendung für jede Teststufe detailliert planen zu können, muss geprüft werden, welche Testtypen mit welcher Intensität, mit welchen Testmitteln und mit welchen Ressourcen durchgeführt werden sollen.

13.1 Bewertung der Testtypen

Testtypen bewerten

Im Rahmen der Testplanung wird festgelegt, welche Testtypen mit welcher Priorität eingesetzt werden sollen. Je nachdem, welche Ziele verfolgt und welche Qualitätsanforderungen an die Web-Anwendung gestellt werden und in welchem technischem Umfeld sie betrieben werden soll, sind die Tests zu planen und durchzuführen.

Zum Beispiel ist für eine Website, die hauptsächlich Informationen bereitstellt, ein Ausfallsicherheitstest nicht von großer Bedeutung, dafür aber ein Auffindbarkeitstest.

Wird eine Applikation in einem Intranet betrieben, in dem alle Benutzer den gleichen Browser benutzen, muss kein Browser-Test über alle existierenden Browser-Typen stattfinden. Wenn aber der Nutzerkreis aus der weltweiten Web-Gemeinde besteht, haben Browser- und Zugänglichkeitstest eine hohe Priorität.

Soll die Anwendung bei mehreren Kunden eingesetzt werden und muss sie deshalb individuell angepasst werden können, sind Code-Inspektionen, Code-Walkthroughs und Installationstests wichtig, um ihre Änderbarkeit und Übertragbarkeit zu gewährleisten.

Die Liste der Beispiele lässt sich beliebig fortführen. In der Tabelle 1.1 des Kap. 1 sind alle in diesem Buch beschriebenen Testtypen aufgeführt. Sie dient als Grundlage für die Planung der in einem Projekt durchzuführenden Tests.

↓

Beispiel zur Bewertung von Testtypen

In der Tabelle 13.1 ist für unser Beispiel aus den vorangehenden Kapiteln, dem Finanzierungsrechner, exemplarisch eine Bewertung aller Testtypen vorgenommen worden:

Tabelle 13.1: Beispielbewertung Testtypen

Testtyp	Bewertung	Begründung
Dokumententest	Hoch	Anforderungsspezifikationen enthalten geschäftsrelevante Funktionsbeschreibungen
Tests zur Funktionalität		
Klassentest	Niedrig	Intensiver Komponententest durch das Testteam
Komponententest	Hoch	Geschäftsrelevante Funktionen
Integrationstest	Hoch	Integration geschäftsrelevanter Subsysteme
Funktionaler Systemtest	Hoch	Geschäftsrelevante Funktionen
Link-Test	Niedrig	Keine externen Links vorgesehen
Cookie-Test	Hoch	Geschäftsrelevante Funktionen setzen Cookies ein
Plugin-Test	Niedrig	Nur zum Lesen von PDF-Dokumenten werden Plugins benötigt
Sicherheitstest	Hoch	Verarbeitung sensibler Daten; Durchführung geschäftsrelevanter Transaktionen
Tests zur Benutzbarkeit		
Content-Test	Hoch	Geschäftsrelevanter Web-Auftritt
Oberflächentest	Hoch	Unternehmensstandards sind einzuhalten

Testtyp	Bewertung	Begründung
Browser-Test	Mittel	Benutzer des Firmenportals setzen in der Regel nur den bekannten Browser XYZ ein
Usability-Test	Hoch	Kundenzufriedenheit ist ein kritischer Erfolgsfaktor
Zugänglichkeitstest	Mittel	Zugänglichkeit nicht explizit gefordert
Auffindbarkeitstest	Niedrig	Benutzer kommen über das gut besuchte Firmenportal
Tests zur Änderbarkeit und Übertragbarkeit		
Code-Analysen	Mittel	Nur stichprobenweise Code-Inspektionen notwendig
Installationstest	Nicht relevant	Anwendung wird im Hause betrieben, keine Client-Installationen notwendig
Tests zur Effizienz und Zuverlässigkeit		
Performanz-/Lasttests	Hoch	Verarbeitung von geschäftsrelevanten Transaktionen, Performanz ist daher kritischer Erfolgsfaktor
Ausfallsicherheitstest	Niedrig	Sicherheitsvorkehrungen zum bestehenden Firmenportal sind etabliert und geprüft
Verfügbarkeitstest	Hoch	24 Std. Verfügbarkeit gefordert

↗ *Testverfahren festlegen*

Wenn Alternativen in der Durchführung eines Testtyps bestehen, muss das Testmanagement in Abhängigkeit der vorgenommenen Bewertung bestimmen, welches konkrete Verfahren eingesetzt werden soll. Folgende Fragen müssen vor der Erstellung des Testplans beantwortet werden:

- Dokumententest

Für welche Typen von Dokumenten sollen nur formale Prüfungen durchgeführt werden und für welche zusätzliche Review-Sitzungen oder schriftliche Stellungnahmen?

- Komponententest

Sollen Testabdeckungsgrade nachgewiesen werden? Wenn ja, für welche Testabdeckung (Methodenabdeckung, Anweisungsabdeckung, Zweigabdeckung und/oder eine der Bedingungsabdeckungen)?

- Integrationstest

Nach welchen Verfahren werden Komponenten und Subsysteme integriert (Anwendungsfallbasiert, Backbone, Ad-hoc oder eine Mischform)?

- Usability-Test

In welchem Umfang soll die Gebrauchstauglichkeit der Anwendung getestet werden (Usability-Labor inkl. Blickregistrierung und/oder Befragung, Online-Umfrage)?

- Zugänglichkeitstest

Soll die Zugänglichkeit nach den Anforderungen der BITV geprüft werden? Wenn ja, genügt der Nachweis der Punkte der Priorität 1?

- Code-Analysen

Für welche Komponenten sollen welche Code-Analysen durchgeführt werden (Statische Code-Analyse, Code-Walkthrough, Code-Inspektion, Schreibtischtest)?

- Ausfallsicherheitstest

Ist zur Überprüfung der Ausfallsicherheit ein anwendungsspezifischer Restart-/Recovery-Test hinreichend, weil die Failover-Verfahren durch das bestehende, produktive Systemumfeld gewährleistet sind? Oder werden neue Failover-Maßnahmen implementiert, die getestet werden müssen?

13.2 Bereitstellung der Testmittel

Nachdem die Testtypen in ihren Ausprägungen feststehen, müssen Checklisten und Testwerkzeuge für die Testdurchführung ausgewählt und bereitgestellt werden.

13.2.1 Bereitstellung von Checklisten

Zur Planung der Qualitätssicherungsmaßnahmen in einem Projekt gehört die Festlegung, welche Checklisten eingesetzt werden sollen. Wenn zum Beispiel die Anwendung in der Web-Programmiersprache PHP (Private Home Page) entwickelt wird und Code-Inspektionen für kritische Programme durchgeführt werden sollen, muss dafür eine entsprechende Checkliste „PHP-Code-Inspektion" bereit gestellt werden.

Der Qualitätssicherungsverantwortliche muss prüfen, ob neue Checklisten erstellt oder ob existierende Checklisten „nur" überarbeitet werden müssen. Der Aufwand und die Ressourcen zur Bereitstellung der Checklisten müssen in die Testplanung einfließen.

Hinweise

In der Tabelle 1.1 des Kapitels 1 sind alle in diesem Buch vorgestellten Checklisten aufgeführt. Sie sollen als fundierte Grundlage für die Erarbeitung projektspezifischer Checklisten dienen.

Zum Einsatz von Checklisten als konstruktive und analytische Qualitätssicherungsmaßnahme wird an dieser Stelle auf das Kapitel 6 verwiesen.

13.2.2 Bereitstellung der Testwerkzeuge

In Abhängigkeit der geplanten Tests müssen Testwerkzeuge ausgewählt und zur Verfügung gestellt werden. Bei der Planung des Einsatzes von Testtools wird geprüft, ob die Werkzeuge schon im Unternehmen im Einsatz sind oder ob sie erst eingeführt und geschult werden müssen. Gegebenenfalls müssen entsprechende Aktivitäten geplant und durchgeführt werden. Eine Vorgehensweise zur Auswahl und Einführung von Testwerkzeugen ist zum Beispiel in [Spillner_2006] beschrieben.

Der Aufwand für die Aktivitäten zur Auswahl und Einführung von Testwerkzeugen ist in keinem Fall zu unterschätzen.

spezielle Testwerkzeuge

Die Tabelle 13.2 zeigt eine Übersicht der unterschiedlichen Arten von Testwerkzeugen, die im Teil II bei den jeweiligen Testtypen beschrieben sind.

Tabelle 13.2: Testtools

Testtyp	Testwerkzeug
Dokumententest	CASE-Tools Review-Managementwerkzeuge
Tests zur Funktionalität	
Klassentest	Testframeworks (xUnits) Treiber und Platzhalter
Komponententest	Capture Replay Tools Code Analyzer Code Coverage Tools Treiber und Platzhalter
Integrationstest	Capture Replay Tools Monitore Treiber und Platzhalter
Funktionaler Systemtest	Capture Replay Tools Monitore
Link-Test	Link-Checker
Cookie-Test	Browser Cookie Viewer
Plugin-Test	Capture Replay Tools User Tracking Tools
Sicherheitstest	Security Test Tools
Tests zur Benutzbarkeit	
Content-Test	Web-Impressum-Assistenten
Oberflächentest	Capture Replay Tools User Tracking Tools
Browser-Test	Browser Capture Replay Tools User Tracking Tools
Usability-Test	Eye Tracker User Tracking Tools
Zugänglichkeitstest	Spezielle Browser Tools zur Zugänglichkeitsprüfung
Auffindbarkeitstest	Ranking Tools

Testtyp	Testwerkzeug
Tests zur Änderbarkeit und Übertragbarkeit	
Code-Analyse	Code Analyzer Compiler Validatoren
Installationstest	Capture Replay Tools
Tests zur Effizienz und Zuverlässigkeit	
Performanz-/Lasttests	Load Test Tools Monitore Profiler
Ausfallsicherheitstest	Load Test Tools Monitore
Verfügbarkeitstest	Monitore

allgemeine Testwerkzeuge

Neben diesen speziellen Testwerkzeugen werden weitere Tools benötigt, die bei mehreren Testtypen und in jeder Teststufe eingesetzt werden können und daher nicht bei den Beschreibungen der einzelnen Testtypen aufgeführt sind. Zu diesen Werkzeugen gehören:

- Datenmanipulationswerkzeuge zum Auslesen und Manipulieren von Dateien und Datenbanken
- Komparatoren zum Vergleichen von Datenbankinhalten und Dateien
- Testdatengeneratoren zum Generieren von Testdaten aus Datenstrukturen oder bestehendem Sourcecode

prozessunterstützende Werkzeuge

Zudem können folgende Werkzeuge die Qualitätsmanagementprozesse in einem Projekt effizient unterstützen:

- Anforderungsmanagementwerkzeuge zur Verwaltung der Spezifikationsdokumente
- Konfigurationsmanagementwerkzeuge zur Versions- und Konfigurationsverwaltung von Systemkomponenten und Dokumenten
- Fehlermanagementwerkzeuge zur Verfolgung von Fehlern und Erstellung von Fehlerstatistiken
- Projektmanagementwerkzeuge zur Planung und Überwachung der Testaktivitäten und Testressourcen

Testsuiten

Sogenannte Testsuiten integrieren mehrere Werkzeugtypen in einem, d.h. unter einer einheitlichen Benutzerführung können mehrere Aufgaben erledigt werden. In einer komfortablen Testsuite können zum Beispiel die Anforderungen und Testfälle beschrieben, die Testfälle mit Risikoklasse versehen, Testszenarien zusammengestellt, Skripte zur automatischen Testdurchführung generiert und ausgeführt, Tests protokolliert und Fehler verfolgt werden.

Diese integrierten Werkzeuge sind sehr nützlich, aber sie sind nicht preiswert und ihre Einführung für ein Unternehmen bzw. für ein Projekt will wohl überlegt und geplant sein.

Testtool-Übersichten

Übersichten zu allen Arten von Testwerkzeugen sind zu finden unter [URL: TestToolsFAQ], [URL: TestToolsJava], [URL: TestToolsOPENSOURCE] und [URL: TestToolsSQA].

13.3 Planung des Testteams

Last but not least muss das Testteam zusammengestellt und zeitlich eingeplant werden.

ausgebildete Tester

Für die Planung und Steuerung der Qualitätssicherungsmaßnahmen, den Entwurf von Testfällen und die Durchführung von Tests werden Tester benötigt, die in der Anwendung von Testmethoden und Testwerkzeugen ausgebildete und erfahren sind. Dieser Personenkreis stellt den größten Teil der Testmannschaft in einem Projekt.

Dieses „Stammtestteam“ muss zeitweise und aufgabenabhängig um Testexperten, neutrale Tester und Personen aus den Fachabteilungen des Auftraggebers erweitert werden.

Testexperten

Bei der Beschreibung der Testtypen im zweiten Teil dieses Buches wird an mehreren Stellen darauf hingewiesen, dass für bestimmte Tests Spezialisten eingesetzt werden sollten, um die Tests effizient durchführen zu können. Generell werden Testexperten eingesetzt, wenn fachliches oder technisches Wissen für einen Test notwendig ist oder wenn ein Testwerkzeug eine besondere Ausbildung erfordert:

- Fachliche Dokumente werden von Fachexperten geprüft, die mit den Themen der Inhalte vertraut sind und diese beurteilen können.
- Der Content-Test wird von Rechts- und Marketingexperten vorgenommen.

- Nicht automatisierte Code-Analysen werden von Personen durchgeführt, welche die eingesetzte Programmiersprache beherrschen.
- SEO-Fachleute überprüfen die Qualität des Programmcodes in Bezug auf seine Optimierung für Suchmaschinen.
- Der Sicherheitstest muss von Sicherheitsexperten und eventuell von „legalen Hackern“ durchgeführt werden.
- Für den Entwurf von technischen Testfällen, wie sie zum Beispiel für einen Plugin-Test (Kap. 8.7) oder den Integrationstest von SOAP-Komponenten (s. Kap. 8.3) benötigt werden, sind entsprechende Technologieexperten zuständig.
- Capture Replay Tools, Load Test Tools und Monitore werden von Testern eingesetzt, die dafür ausgebildet sind, denn diese Werkzeuge verlangen spezielle Programmierkenntnisse.
- Der Einsatz von Page Ranking Tools und User Tracking Tools verlangt Erfahrungen im Web-Marketing.

neutrale Tester

Nicht nur Testspezialisten müssen zum Testen eingeplant werden. Zum Beispiel können Personen ohne Testerausbildung den Oberflächentest durchführen (Kap. 9.2) oder neutrale Personen für den Gebrauchstauglichkeitstest im Usability-Labor (Kap. 9.4) rekrutiert werden.

Auftraggeber als Tester

Für den Abnahmetest und für Prüfungen von wichtigen Zwischenergebnissen müssen Fachleute vom Auftraggeber benannt werden und zum Zeitpunkt der Prüfung bereit stehen. Dieser Personenkreis muss, wie alle anderen am Test beteiligten Personen auch, mit den zu leistenden Aufwänden und den festgelegten Terminen in der Testplanung aufgenommen werden.

13.4 Zusammenfassung

- Im Rahmen der Testplanung wird festgelegt, welche Testtypen in welcher Ausprägung bei den Tests einer Web-Applikation eingesetzt werden sollen.
- Alternativen zur Durchführung eines Testtyps werden in Abhängigkeit der vorgenommenen Bewertung ausgewählt.
- Ebenso müssen die notwendigen Testmethoden und Testmittel (Checklisten und Werkzeuge) ausgewählt, bereitgestellt und ggf. geschult werden.

- Das Testteam wird so zusammengestellt, dass alle fachlichen und technischen Qualitätssicherungsmaßnahmen mit dem notwendigen Know-how durchgeführt werden können.
- Zum Testteam gehören ausgebildete Tester, Testexperten, neutrale Testpersonen und Fachleute des Auftraggebers.

14 Planung der Teststufen

„Management ist die Kunst, drei Leute dazu zu bringen, die Arbeit von drei Leuten zu tun."

William Feather (1889-1969)

Ein Software-Entwicklungsprojekt durchläuft mehrere Teststufen (Testphasen), je nach gewähltem Vorgehen auch mehrfach. Die Testobjekte werden dabei von einer Teststufe in die nächste übergeben, so dass sich eine feste Reihenfolge der Teststufen ergibt:

1. Entwicklertest
2. Komponententest
3. Integrationstest
4. Systemtest
5. Abnahmetest
6. Betrieb

Die erste und die letzte aufgeführte Teststufe, der Entwicklertest und der Betrieb, sind keine „offiziellen" Testphasen. Das soll heißen, dass sie normalerweise nicht als eigene Teststufen angesehen werden. Aber bei der Testplanung müssen sie berücksichtigt werden, denn Maßnahmen zur Qualitätssicherung einer Web-Anwendung beginnen bei der Entwicklung und sind auch während des gesamten

Betriebes notwendig. Daher werden diese beiden Phasen im Lebenszyklus eines Anwendungssystems als Teststufen betrachtet.

In den folgenden Kapiteln sind jeder Teststufe die Testtypen zugeordnet, die normalerweise in der beschriebenen Teststufe eingesetzt werden.

Der Dokumententest ist bei keiner Teststufe explizit aufgeführt, weil er prinzipiell immer eingesetzt wird, wenn ein Dokument geprüft wird. Er kann also jeder Testphase zugeordnet werden:

→ Dokumententest (Kap. 7.1)

14.1 Teststufe Entwicklertest

Jedem Komponententest einer Komponente durch das Testteam geht ein Klassentest durch den Entwickler voran. Wie in den Kapiteln 8.1 (Klassentest) und 8.2 (Komponententest) beschrieben, können bei entsprechender Planung dieser beiden Testtypen Synergien beim Testen genutzt werden.

Neben den Funktionstests kann es sinnvoll sein, dass statische Code-Analysen und Zugänglichkeitstests schon vom Entwickler mit den entsprechenden Werkzeugen durchgeführt werden.

Testtypen zur Teststufe Entwicklertest:

→ Klassentest (Kap. 8.1)
→ Statische Code-Analyse (Abschn. 10.1.4)
→ Zugänglichkeitstest (Kap. 9.5)

14.2 Teststufe Komponententest

Der Komponententest ist die erste Teststufe, in der vom Testteam in einer separaten Testumgebung systematische Tests einzelner Software-Komponenten durchgeführt werden.

Neben dem Testtyp Komponententest sind in der Teststufe Komponententest die web-spezifischen Testtypen Cookie-Test, Link-Test und Plugin-Test zum Nachweis der Funktionalität einzuplanen.

Zur Überprüfung der Benutzbarkeit einer Komponente sollten Oberflächen-, Browser- und Zugänglichkeitstests schon zu diesem Zeitpunkt begonnen werden, ebenso wie Reviews zu den Sicherheitsvorkehrungen.

Code-Analysen zur Überprüfung der Änderbarkeit und Übertragbarkeit einer Software gehören in die Testphase Komponententest, weil sie sehr entwicklungsnah durchgeführt werden und der Sourcecode den Testern zur Verfügung gestellt werden muss.

Performanztests für kritische Komponenten können während des Komponententests vorgesehen werden, denn sie decken nicht laufzeitoptimierten Sourcecode in den Programmmodulen auf.

Testtypen zur Teststufe Komponententest:

- → Komponententest (Kap. 8.2)
- → Link-Test (Kap. 8.5)
- → Cookie-Test (Kap. 8.6)
- → Plugin-Test (Kap. 8.7)
- → Sicherheitstest (Kap. 8.8)
- → Content-Test (Kap. 9.1)
- → Oberflächentest (Kap. 9.2)
- → Browser-Test (Kap. 9.3)
- → Zugänglichkeitstest (Kap. 9.5)
- → Code-Analysen (Kap. 10.1)
- → Performanztest (Abschn. 11.1.1)

14.3 Teststufe Integrationstest

Der Test der Integration von Komponenten zu Subsystemen und von Subsystemen zum Gesamtsystem stellt im Projektverlauf eine eigene Teststufe dar. Diese Tests werden vom Testteam in einer produktionsnahen Testumgebung durchgeführt, die unter Umständen auch im Systemtest genutzt werden kann.

Der in Kap. 9.3 beschriebene Testtyp Integrationstest stellt mit seinen verschiedenen Vorgehensweisen die Funktionalität einer Web-Anwendung sicher und ist der entscheidende Testtyp in der Integrationsteststufe. Plugins müssen spätestens in dieser Stufe integriert werden.

Tests zur Gebrauchstauglichkeit und zur Performanz von Subsystemen sollten in dieser Teststufe begonnen werden, weil Mängel in der Usability und beim Laufzeitverhalten ein Redesign der Anwendung erforderlich machen können, welches zu diesem Zeitpunkt mit noch relativ geringem Aufwand durchgeführt werden kann.

Testtypen zur Teststufe Integrationstest:

- → Plugin-Test (Kap. 8.7)
- → Integrationstest (Kap. 8.3)
- → Usability-Test (Kap. 9.4)
- → Performanztest (Abschn. 11.1.1)

14.4 Teststufe Systemtest

In der Teststufe Systemtest wird das komplette Anwendungssystem vom Testteam in einer produktionsnahen Systemtestumgebung gegen die Anforderungen des Auftraggebers getestet. Da funktionale und nicht funktionale Anforderungen an ein System gestellt werden, kann auch der Systemtest in einen funktionalen und einen nicht funktionalen Test unterteilt werden.

funktionaler Systemtest

Der funktionale Teil der Teststufe Systemtest, in dem die Funktionalität des Gesamtsystems überprüft wird, besteht nur aus dem Testtyp „funktionaler Systemtest“. Dieser Testtyp wird im Kapitel 8.4 beschrieben.

nicht-funktionaler Systemtest

Im nicht-funktionalen Teil der Teststufe Systemtest wird das Gesamtsystem gegen die nicht funktionalen Anforderungen getestet, d.h. die Qualitätsmerkmale Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit werden geprüft.

Somit können alle Testtypen, die das Gesamtsystem in einer produktionsnahen Testumgebung voraussetzen, aber nicht der endgültigen Abnahme durch den Auftraggeber dienen, in der Phase Systemtest eingesetzt werden. Dazu gehören Sicherheitstest, Usability-Test, Installationstest, Ausfallsicherheitstest und alle Ausprägungen der Performanz-/Lasttests. Zudem sollte, bevor die Anwendung zur Abnahme an den Auftraggeber übergeben wird, die Rechtskonformität, die Browser-Kompatibilität und die Zugänglichkeit der Web-Applikation abschließend sichergestellt werden.

Testtypen zur Teststufe Systemtest:

- → Funktionaler Systemtest (8.4)
- → Sicherheitstest (Kap. 8.8)
- → Content-Test (Kap. 9.1)
- → Browser-Test (Kap. 9.3)
- → Usability-Test (Kap. 9.4)
- → Zugänglichkeitstest (Kap. 9.5)
- → Installationstest (Kap. 10.2).

→ Performanztest (Abschn. 11.1.1)
→ Lasttest (Abschn. 11.1.2)
→ Skalierbarkeitstest (Abschn. 11.1.3)
→ Speicherlecktest (Abschn. 11.1.4)
→ Ausfallsicherheitstest (Kap. 11.2)

14.5 Teststufe Abnahmetest

Nachdem die Teststufe Systemtest erfolgreich abgeschlossen ist, muss der Auftraggeber im nächsten Schritt die Freigabe der Anwendung erteilen, d.h. das Gesamtsystem abnehmen.

Abnahmetest

Mit dem Abnahmetest prüft der Auftraggeber, ob die beauftragte Anwendung den vertraglich vereinbarten Anforderungen entspricht und alle Abnahmekriterien erfüllt sind. Der Abnahmetest hat das Ziel, dem Auftraggeber die Entscheidung zu ermöglichen, ob ein System abzunehmen (anzunehmen) ist oder nicht.

14.5.1 Abnahmephasen

Im Rahmen des Abnahmetests können die beiden Phasen Alpha- und Beta-Test eingeplant werden, die der endgültigen Freigabe einer Software vorangehen.

Alpha-Test

Im Alpha-Test wird eine Vorabversion der Software vom Auftraggeber in einer Testumgebung des Software-Lieferanten getestet.

Beta-Test

Im Beta-Test wird eine Vorabversion der Software vom Auftraggeber in der Produktionsumgebung des Auftraggebers getestet.

Alpha- und Betatest stellen sicher, dass der Auftraggeber frühzeitig die Umsetzung seiner Anforderungen und die Benutzbarkeit des Systems bestätigt und so „böse“ Überraschungen bei der endgültigen Abnahme ausbleiben.

Natürlich ist es sinnvoll und eine wichtige Aufgabe des Testmanagements, den Auftraggeber und die Endbenutzer schon vor der Endabnahme in die Testaktivitäten mit einzubeziehen, also auch schon vor dem Alpha-Test oder dem Beta-Test. Das trifft besonders auf den Usability-Test zu, weil die Akzeptanz der Web-Anwendung ein kritischer Erfolgsfaktor ist.

14.5.2 Abnahmekriterien

Kriterienkatalog zur Abnahme

Zu den Anforderungsbeschreibungen eines beauftragten Anwendungssystems gehört ein Katalog von Abnahmekriterien, gegen den der Auftraggeber die Anwendung im Abnahmetest prüft. Allein dieser Katalog ist für die Abnahme ausschlaggebend. Projektleitung und Qualitätssicherung müssen zu Projektbeginn dafür sorgen, dass ein solcher verbindlicher Kriterienkatalog vereinbart wird. Nur so können unnötige Diskussionen bei der Freigabe und nachträgliche Anforderungen vermieden werden.

14.5.3 Testtypen im Abnahmetest

Die Abnahme einer Web-Anwendung erfolgt durch Tests, die der Auftraggeber in einer produktionsnahen Systemumgebung oder sogar in der Produktionsumgebung durchführt.

Anhand wichtiger und „normaler" Geschäftsvorfälle werden Funktionalität, Benutzbarkeit und Effizienz aus Sicht des Auftraggebers geprüft.

Zur Abnahme einer Software gehört auch die Freigabe durch den Betreiber (Rechenzentrum), der für Übertragbarkeit, Zuverlässigkeit und Ausfallsicherheit des Systems verantwortlich ist und die entsprechenden Tests durchführt.

Funktionale und systemtechnische Ausnahmesituationen bleiben in der Regel beim Abnahmetest unberücksichtigt, weil sie in den vorangegangenen Teststufen durchgeführt und – hoffentlich – nachweislich dokumentiert worden sind. Es obliegt dem Auftraggeber, neben der Durchführung seiner Testfälle diese Testdokumentation zu überprüfen und das Verhalten der Anwendung in Ausnahmesituationen stichprobenartig zu testen.

Messungen zur Verfügbarkeit können in dieser Phase für den Produktionsbetrieb aufschlussreich sein.

Testtypen zur Teststufe Abnahmetest:

- → Dokumententest (Kap. 7)
- → Funktionaler Systemtest (8.4)
- → Sicherheitstest (Kap. 8.8)
- → Content-Test (Kap. 9.1)
- → Browser-Test (Kap. 9.3)

→ Usability-Test (Kap. 9.4)
→ Zugänglichkeitstest (Kap. 9.5)
→ Auffindbarkeitstest (Kap. 9.6)
→ Code-Analysen (Kap. 10.1)
→ Installationstest (Kap. 10.2)
→ Performanztest (Abschn. 11.1.1)
→ Lasttest (Abschn. 11.1.2)
→ Ausfallsicherheitstest (Kap. 11.2)
→ Verfügbarkeitstest (Kap. 11.3)

14.6 Teststufe Betrieb

Für die Betriebphase eines Anwendungssystems müssen wie für die vorangehenden Teststufen Qualitätssicherungsmaßnahmen geplant werden.

14.6.1 Pilotbetrieb

Bevor eine Anwendung für alle Benutzer freigegeben wird, kann ein zeitlich begrenzter Pilotbetrieb sinnvoll sein, wenn zum Beispiel Geschäfte mit Hilfe der Anwendung abgewickelt werden. Weil das Anwendungssystem zu diesem Zeitpunkt produktiv geschaltet ist, gehört ein Pilotbetrieb zur Betriebsphase.

Pilotbetrieb

Im Pilotbetrieb wird die Anwendung in der Produktionsumgebung des Auftraggebers einem eingeschränkten und wohldefinierten Benutzerkreis (Pilotanwender) für einen begrenzten Zeitraum freigeschaltet, mit dem Ziel, Funktionalität, Benutzbarkeit und Performanz der Anwendung zu bewerten.

Folgende Punkte sind bei der Planung des Pilotbetriebes einer Webanwendung zu beachten:

✓Checkliste Planung Pilotbetrieb

- Ist der Pilotbetrieb zeitlich begrenzt?
- Sind Kriterien für die Beendigung bzw. die Verlängerung des Pilotbetriebes festgelegt?
- Sind die Pilotanwender repräsentativ ausgewählt?
- Ist sichergestellt, dass nur die Pilotanwender die Anwendung erreichen können?
- Sind die Entwicklungs- und Testressourcen für die Pilotphase eingeplant?

- Ist ein Verfahren beschrieben, wie die von den Pilotanwendern gefundenen Fehler und eingebrachten Verbesserungsvorschläge dokumentiert, bewertet und umgesetzt werden?
- Sind die Verantwortlichen und Beteiligten für den Pilotbetrieb benannt?
- Sind Messungen der Performanz, des Ressourcenverbrauchs und der Verfügbarkeit geplant?
- Sind Usability-Umfragen und User Tracking im Pilotbetrieb geplant?
- Sind regelmäßige Feedback-Veranstaltungen mit den Pilotanwendern geplant?

Qualitätsanforderungen an den Pilotbetrieb

Erst wenn die „Kinderkrankheiten" während der Pilotphase behoben und die Pilotbenutzer „zufrieden" sind, darf die Anwendung allen Benutzern zugänglich gemacht werden. Das setzt voraus, dass die Kriterien zur Beendigung des Pilotbetriebes als Qualitätsanforderungen festgelegt worden sind, wie zum Beispiel:

- Der Usability-Test ist positiv abgeschlossen.
- Die vereinbarte Systemverfügbarkeit wird eingehalten.
- Die maximal erlaubte Anzahl neu auftretender Fehler pro Zeiteinheit und Risikoklasse wird nicht überschritten.

14.6.2 Qualitätsprüfungen im laufenden Betrieb

Während der Pilotphase und des sich anschließenden Betriebes eines Systems zeigt sich, ob die Qualitätsanforderungen Performanz, Ressourcenverbrauch, Zuverlässigkeit, Auffindbarkeit und Wartbarkeit erfüllt werden und dauerhaft gehalten werden können. Daher sind permanente Messungen und Prüfungen im laufenden Betrieb notwendig:

- Externe Links müssen regelmäßig daraufhin geprüft werden, dass sie ihr Ziel noch erreichen und, wenn dieses der Fall ist, dass die verlinkten Inhalte sich nicht signifikant verändert haben.
- Performanz und Ressourcenverbrauch eines produktiven Systems müssen überwacht werden, damit bei Verschlechterung der Systemleistung rechtzeitig gehandelt werden kann.

- Die Verfügbarkeit und die daraus abgeleitete Ausfallrate müssen gemessen werden.
- Da im Internet immer neue Sicherheitslücken und -risiken bekannt werden und auftreten können, müssen regelmäßig die Sicherheitsmaßnahmen überprüft und die Sicherheitstests wiederholt werden.
- Die Auffindbarkeit einer Website muss regelmäßig überprüft werden, um sicherzustellen, dass sie in Suchmaschinen dauerhaft in den vorderen Positionen zu finden ist.

In conclusio ist der produktive Betrieb einer Web-Anwendung wie eine Teststufe zu planen, zu steuern und auszuwerten.

Testtypen zur Teststufe Betrieb:

→ Link-Test (Kap. 8.5)
→ Sicherheitstest (Kap. 8.8)
→ Usability-Test (Kap. 9.4)
→ Auffindbarkeitstest (Kap. 9.6)
→ Performanztest (Abschn. 11.1.1)
→ Verfügbarkeitstest (Kap. 11.3)

14.6.3 Planung der Testzyklen

In den vorangegangenen Kapiteln sind die Teststufen erläutert, die in einem Projekt durchlaufen werden. Bei der Planung der Tests muss festgelegt werden, nach welchem Modell vorgegangen wird. Dazu sind Alternativen abzuwägen, welche die Planung der Tests entscheidend gestalten:

1. Soll eine Entwicklungs- bzw. Teststufe komplett abgeschlossen werden, bevor eine neue beginnt?
2. Oder soll die Anwendung in kleinere Einheiten unterteilt werden, die nacheinander realisiert und getestet werden (inkrementelles Vorgehen)? Auf diese Weise würde die Anwendung in mehreren Stufen fertiggestellt. In diesem Fall ist weiter zu entscheiden:
   2.1. Sollen die einzelnen Einheiten bis zur Produktionsreife entwickelt und freigegeben werden?

2.2. Oder sollen die Einheiten bis inklusive Abnahmetest fertiggestellt werden, weil die komplette Anwendung nur als Ganzes an den Betrieb übergeben werden soll?

2.3. Oder sollen die Einheiten nur bis einschließlich Systemtest fertiggestellt werden, weil der Auftraggeber die Anwendung nicht in mehreren Stufen abnehmen, sondern nur einen Abnahmetest des Gesamtsystems durchführen will?

Je nach Vorgehensweise werden die Teststufen im Laufe eines Projektes einmal oder in Testzyklen mehrfach durchlaufen. Dementsprechend müssen die Tests geplant und überwacht werden.

14.7 Zusammenfassung

- In einem Anwendungsentwicklungsprojekt sind die Teststufen Entwicklertest, Komponententest, Integrationstest, Systemtest, Abnahmetest und Betrieb zu planen.
- In jeder Teststufe kommen mehrere Testtypen zum Tragen.
- Der Abnahmetest kann in mehreren Stufen mit Alpha-Test und Beta-Test durchgeführt werden. Der Alpha-Test findet beim Lieferanten statt. Der Beta-Test findet beim Auftraggeber statt.
- In der Pilotphase wird eine Anwendung zum Start des Betriebes einem begrenzten Benutzerkreis zur Verfügung gestellt.
- Auch im laufenden Betrieb werden Qualitätssicherungsmaßnahmen durchgeführt.
- Je nach Vorgehensweise müssen die Teststufen für mehrere Testzyklen geplant und durchgeführt werden.
- In Abhängigkeit der geplanten Testzyklen ist die Planung der Testressourcen eine komplexe Aufgabe, zumal vielfältige Abhängigkeiten zu den übrigen Projektgrößen Budget, Fertigstellungstermine, Personal und technische Ressourcen bestehen.

Anhang

A Fragebögen zum Usability-Test
B Checklisten BITV zum Zugänglichkeitstest

A Anhang: Fragebögen zum Usability-Test

Im Folgenden sind zwei Tabellen zum Usability-Test abgebildet. Tabelle A.1 enthält einen Fragebogen zum Nutzungsverhalten und Tabelle A.2 einen Bewertungsbogen zur Usability einer Web-Anwendung.

Der Einsatz dieser beiden Befragungsmittel ist im Kapitel 9.4, Usability-Test, beschrieben.

Tabelle A.1: Fragebogen Nutzungsverhalten

Fragen zum Nutzungsverhalten	Antwort
Haben Sie einen Internet-Anschluss?	
1. dienstlich	
2. privat	
Welche(n) Browser benutzen Sie?	
1. dienstlich	
2. privat	
Welche Browser-Einstellungen haben Sie normalerweise aktiviert?	
1. JavaScript	
2. Java	
3. Cookies	
4. Warnmeldung bei Cookies	
5. Blocken von PopUp-Fenstern	
6. Sonstiges	
Welches Betriebssystem benutzen Sie?	
1. dienstlich	
2. privat	
Mit welchen Bildschirmauflösungen arbeiten Sie?	
1. 640 x 480 Pixel	
2. 800 x 600 Pixel	
3. 1024 x 768 Pixel	
4. 1280 x 1024 Pixel	
Wie viele Stunden surfen Sie pro Woche im Internet?	
1. dienstlich	
2. privat	
Welchen technischen Zugang haben Sie zum Internet?	
1. Analog	
2. ISDN	
3. DSL 1000	
4. DSL 2000 oder DSL 6000	
Haben Sie eine Flatrate?	
1. Ja	
2. Nein	
Haben Sie Programmierkenntnisse?	
1. Nein	
2. Ja, folgende Programmiersprachen:	

Merkmal	Zu bewertende Aussage	+ +	+	-	--
Aufgaben-angemessen	Ich hatte nie das Gefühl, mit meiner Aufgabe nicht weiter zu kommen.				
	Ich bin nie mit Informationen überfrachtet worden.				
Erwartungs-konform	Die Seitengestaltung entspricht meinen Vorstellungen.				
	Ich konnte die Schrift gut lesen.				
Fehler-tolerant	Hinweise und Fehlermeldungen sind eindeutig und verständlich.				
Lernförder-lich	Die Web-Applikation ist einfach zu erlernen.				
	Die Beispiele zu jeder Teilfunktion haben mir geholfen, die Vorgänge zu verstehen und nachzuvollziehen.				
	Es fiel mir bei keiner Funktion schwer, sie zu wiederholen.				
Selbst-beschrei-bungsfähig	Die Texte und Beschreibungen sind eindeutig und verständlich.				
	Ich habe alle Menüpunkte schnell gefunden.				
	Ich wusste immer, wo ich mich befand.				
	Die Beschreibungstexte der Links beschreiben eindeutig das jeweilige Ziel.				
	Ich war auf die Hilfefunktion nicht angewiesen.				
Steuerbar	Die Navigation ist klar und übersichtlich.				
	Ich wusste immer, wie ich zu einer bestimmten Funktion komme.				
	Die benötigten Links habe ich immer schnell gefunden.				
	Ich konnte mich an jeder Stelle einfach entscheiden, wie ich weiter machen sollte.				
	Ich wusste immer, wie ich zu einem bestimmten Punkt gekommen war.				
	Ich hatte nie Schwierigkeiten, den für die Aufgabe optimalen Weg zu finden.				
Meine Verbesserungsvorschläge:					

Tabelle A.2: Bewertungsbogen Usability

B Anhang: Checklisten BITV zum Zugänglichkeitstest

Im Folgenden sind die Checklisten zur Barrierefreien Informationstechnik Verordnung (BITV) vom Aktionsbündnis für barrierefreie Informationstechnik (AbI) mit deren freundlichen Genehmigung abgebildet ([URL: BITVCL], Stand vom 29.05.2006).

B.1 Checkliste BITV Priorität I

Für alle Angebote, die neu gestaltet oder in wesentlichen Bestandteilen oder größerem Umfang verändert oder angepasst werden:

			J	N	kA
Anforderung	1	Für jeden Audio- oder visuellen Inhalt sind geeignete äquivalente Inhalte bereitzustellen, die den gleichen Zweck oder die gleiche Funktion wie der originäre Inhalt erfüllen.			
Bedingung	1.1	Für jedes Nicht-Text-Element ist ein äquivalenter Text bereitzustellen. Dies gilt insbesondere für: Bilder, graphisch dargestellten Text einschließlich Symbolen, Regionen von Imagemaps, Animationen (z. B. animierte GIFs), Applets und programmierte Objekte, Zeichnungen, die auf der Verwendung von Zeichen und Symbolen des ASCII-Codes basieren (ASCII-Zeichnun-gen), Frames, Scripts, Bilder, die als Punkte in Listen verwendet werden, Platzhalter-Graphiken, graphische Buttons,			

			J	N	kA
		Töne (abgespielt mit oder ohne Einwirkung des Benutzers), Audio-Dateien, die für sich allein stehen, Tonspuren von Videos und Videos.			
	1.2	Für jede aktive Region einer serverseitigen Imagemap sind redundante Texthyperlinks bereitzustellen.			
	1.3	Für Multimedia-Präsentationen ist eine Audio-Beschreibung der wichtigen Informationen der Videospur bereitzustellen.			
	1.4	Für jede zeitgesteuerte Multimedia-Präsentation (insbesondere Film oder Animation) sind äquivalente Alternativen (z.B. Untertitel oder Audiobeschreibungen der Videospur) mit der Präsentation zu synchronisieren.			
Anforderung	2	Texte und Graphiken müssen auch dann verständlich sein, wenn sie ohne Farbe betrachtet werden.			
Bedingung	2.1	Alle mit Farbe dargestellten Informationen müssen auch ohne Farbe verfügbar sein, z.B. durch den Kontext oder die hierfür vorgesehenen Elemente der verwendeten Markup-Sprache.			
	2.2	Bilder sind so zu gestalten, dass die Kombinationen aus Vordergrund- und Hintergrundfarbe auf einem Schwarz-Weiß-Bildschirm und bei der Betrachtung durch Menschen mit Farbfehlsichtigkeiten ausreichend kontrastieren.			
Anforderung	3	Markup-Sprachen (insbesondere HTML) und Stylesheets sind entsprechend ihrer Spezifikationen und formalen Definitionen zu verwenden.			
Bedingung	3.1	Soweit eine angemessene Markup-Sprache existiert, ist diese anstelle von Bildern zu verwenden, um Informationen darzustellen.			
	3.2	Mittels Markup-Sprachen geschaffene Dokumente sind so zu erstellen und zu deklarieren, dass sie gegen veröffentlichte formale Grammatiken validieren.			

			J	N	kA
	3.3	Es sind Stylesheets zu verwenden, um die Text- und Bildgestaltung sowie die Präsentation von mittels Markup-Sprachen geschaffener Dokumente zu beeinflussen.			
	3.4	Es sind relative anstelle von absoluten Einheiten in den Attributwerten der verwendeten Markup-Sprache und den Stylesheet-Property-Werten zu verwenden.			
	3.5	Zur Darstellung der Struktur von mittels Markup-Sprachen geschaffener Dokumente sind Überschriften-Elemente zu verwenden.			
	3.6	Zur Darstellung von Listen und Listenelementen sind die hierfür vorgesehenen Elemente der verwendeten Markup-Sprache zu verwenden.			
	3.7	Zitate sind mittels der hierfür vorgesehenen Elemente der verwendeten Markup-Sprache zu kennzeichnen.			
Anforderung	4	Sprachliche Besonderheiten wie Wechsel der Sprache oder Abkürzungen sind erkennbar zu machen.			
Bedingung	4.1	Wechsel und Änderungen der vorherrschend verwendeten natürlichen Sprache sind kenntlich zu machen.			
Anforderung	5	Tabellen sind mittels der vorgesehenen Elemente der verwendeten Markup-Sprache zu beschreiben und in der Regel nur zur Darstellung tabellarischer Daten zu verwenden.			
Bedingung	5.1	In Tabellen, die tabellarische Daten darstellen, sind die Zeilen- und Spaltenüberschriften mittels der vorgesehenen Elemente der verwendeten Markup-Sprache zu kennzeichnen.			

			J	N	kA
	5.2	Soweit Tabellen, die tabellarische Daten darstellen, zwei oder mehr Ebenen von Zeilen- und Spaltenüberschriften aufweisen, sind mittels der vorgesehenen Elemente der verwendeten Markup-Sprache Datenzellen und Überschriftenzellen einander zuzuordnen.			
	5.3	Tabellen sind nicht für die Text- und Bildgestaltung zu verwenden, soweit sie nicht auch in linearisierter Form dargestellt werden können.			
	5.4	Soweit Tabellen zur Text- und Bildgestaltung genutzt werden, sind keine der Strukturierung dienenden Elemente der verwendeten Markup-Sprache zur visuellen Formatierung zu verwenden.			
Anforderung	6	Internetangebote müssen auch dann nutzbar sein, wenn der verwendete Benutzeragent neuere Technologien nicht unterstützt oder diese deaktiviert sind.			
Bedingung	6.1	Es muss sichergestellt sein, dass mittels Markup-Sprachen geschaffene Dokumente verwendbar sind, wenn die zugeordneten Stylesheets deaktiviert sind.			
	6.2	Es muss sichergestellt sein, dass Äquivalente für dynamischen Inhalt aktualisiert werden, wenn sich der dynamische Inhalt ändert.			
	6.3	Es muss sichergestellt sein, dass mittels Markup-Sprachen geschaffene Dokumente verwendbar sind, wenn Scripts, Applets oder andere programmierte Objekte deaktiviert sind.			
	6.4	Es muss sichergestellt sein, dass die Eingabebehandlung von Scripts, Applets oder anderen programmierten Objekten vom Eingabegerät unabhängig ist.			

			J	N	kA
	6.5	Dynamische Inhalte müssen zugänglich sein. Insoweit dies nur mit unverhältnismäßig hohem Aufwand zu realisieren ist, sind gleichwertige alternative Angebote unter Verzicht auf dynamische Inhalte bereitzustellen.			
Anforderung	7	Zeitgesteuerte Änderungen des Inhalts müssen durch die Nutzerin, den Nutzer kontrollierbar sein.			
Bedingung	7.1	Bildschirmflackern ist zu vermeiden.			
	7.2	Blinkender Inhalt ist zu vermeiden.			
	7.3	Bewegung in mittels Markup-Sprachen geschaffener Dokumente ist entweder zu vermeiden oder es sind Mechanismen bereitzustellen, die der Nutzerin, dem Nutzer das Einfrieren der Bewegung oder die Änderung des Inhalts ermöglichen.			
	7.4	Automatische periodische Aktualisierungen in mittels Markup-Sprachen geschaffener Dokumente sind zu vermeiden.			
	7.5	Die Verwendung von Elementen der Markup-Sprache zur automatischen Weiterleitung ist zu vermeiden. Insofern auf eine automatische Weiterleitung nicht verzichtet werden kann, ist der Server entsprechend zu konfigurieren.			
Anforderung	8	Die direkte Zugänglichkeit der in Internetangeboten eingebetteten Benutzerschnittstellen ist sicherzustellen.			
Bedingung	8.1	Programmierte Elemente (insbesondere Scripts und Applets) sind so zu gestalten, dass sie entweder direkt zugänglich oder kompatibel mit assistiven Technologien sind.			
Anforderung	9	Internetangebote sind so zu gestalten, dass Funktionen unabhängig vom Eingabegerät oder Ausgabegerät nutzbar sind.			

			J	N	kA
Bedingung	9.1	Es sind clientseitige Imagemaps bereitzustellen, es sei denn die Regionen können mit den verfügbaren geometrischen Formen nicht definiert werden.			
	9.2	Jedes über eine eigene Schnittstelle verfügende Element muss in geräteunabhängiger Weise bedient werden können.			
	9.3	In Scripts sind logische anstelle von geräteabhängigen Event-Handlern zu spezifizieren.			
Anforderung	10	Die Verwendbarkeit von nicht mehr dem jeweils aktuellen Stand der Technik entsprechenden assistiven Technologien und Browsern ist sicherzustellen, so weit der hiermit verbundene Aufwand nicht unverhältnismäßig ist.			
Bedingung	10.1	Das Erscheinenlassen von Pop-Ups oder anderen Fenstern ist zu vermeiden. Die Nutzerin, der Nutzer ist über Wechsel der aktuellen Ansicht zu informieren.			
	10.2	Bei allen Formular-Kontrollelementen mit implizit zugeordneten Beschriftungen ist dafür Sorge zu tragen, dass die Beschriftungen korrekt positioniert sind.			
Anforderung	11	Die zur Erstellung des Internetangebots verwendeten Technologien sollen öffentlich zugänglich und vollständig dokumentiert sein, wie z.B. die vom World Wide Web Consortium entwickelten Technologien.			
Bedingung	11.1	Es sind öffentlich zugängliche und vollständig dokumentierte Technologien in ihrer jeweils aktuellen Version zu verwenden, soweit dies für die Erfüllung der angestrebten Aufgabe angemessen ist.			
	11.2	Die Verwendung von Funktionen, die durch die Herausgabe neuer Versionen überholt sind, ist zu vermeiden.			

			J	N	kA
	11.3	Soweit auch nach bestem Bemühen die Erstellung eines barrierefreien Internetangebots nicht möglich ist, ist ein alternatives, barrierefreies Angebot zur Verfügung zu stellen, dass äquivalente Funktionalitäten und Informationen gleicher Aktualität enthält, soweit es die technischen Möglichkeiten zulassen. Bei Verwendung nicht barrierefreier Technologien sind diese zu ersetzen, sobald aufgrund der technologischen Entwicklung äquivalente, zugängliche Lösungen verfügbar und einsetzbar sind.			
Anforderung	12	Der Nutzerin, dem Nutzer sind Informationen zum Kontext und zur Orientierung bereitzustellen.			
Bedingung	12.1	Jeder Frame ist mit einem Titel zu versehen, um Navigation und Identifikation zu ermöglichen.			
	12.2	Der Zweck von Frames und ihre Beziehung zueinander ist zu beschreiben, soweit dies nicht aus den verwendeten Titeln ersichtlich ist.			
	12.3	Große Informationsblöcke sind mittels Elementen der verwendeten Markup-Sprache in leichter handhabbare Gruppen zu unterteilen.			
	12.4	Beschriftungen sind genau ihren Kontrollelementen zuzuordnen.			
Anforderung	13	Navigationsmechanismen sind übersichtlich und schlüssig zu gestalten.			
Bedingung	13.1	Das Ziel jedes Hyperlinks muss auf eindeutige Weise identifizierbar sein.			
	13.2	Es sind Metadaten bereitzustellen, um semantische Informationen zu Internetangeboten hinzuzufügen.			
	13.3	Es sind Informationen zur allgemeinen Anordnung und Konzeption eines Internetangebots, z.B. mittels eines Inhaltsverzeichnisses oder einer Sitemap, bereitzustellen.			

			J	N	kA
	13.4	Navigationsmechanismen müssen schlüssig und nachvollziehbar eingesetzt werden.			
Anforderung	14	Das allgemeine Verständnis der angebotenen Inhalte ist durch angemessene Maßnahmen zu fördern.			
Bedingung	14.1	Für jegliche Inhalte ist die klarste und einfachste Sprache zu verwenden, die angemessen ist.			

Legende: J = Ja, N = Nein, kA = keine Angabe

B.2 Checkliste BITV Priorität II

Zusätzlich für alle zentralen Navigations- und Einstiegsangebote:

			J	N	kA
Anforderung	1	Für jeden Audio- oder visuellen Inhalt sind geeignete äquivalente Inhalte bereitzustellen, die den gleichen Zweck oder die gleiche Funktion wie der originäre Inhalt erfüllen.			
Bedingung	1.5	Für jede aktive Region einer clientseitigen Imagemap sind redundante Texthyperlinks bereitzustellen.			
Anforderung	2	Texte und Graphiken müssen auch dann verständlich sein, wenn sie ohne Farbe betrachtet werden.			
Bedingung	2.3	Texte sind so zu gestalten, dass die Kombinationen aus Vordergrund- und Hintergrundfarbe auf einem Schwarz-Weiß-Bildschirm und bei der Betrachtung durch Menschen mit Farbfehlsichtigkeiten ausreichend kontrastieren.			
Anforderung	3	Markup-Sprachen (insbesondere HTML) und Stylesheets sind entsprechend ihrer Spezifikationen und formalen Definitionen zu verwenden.			
Anforderung	4	Sprachliche Besonderheiten wie Wechsel der Sprache oder Abkürzungen sind erkennbar zu machen.			

			J	N	kA
Bedingung	4.2	Abkürzungen und Akronyme sind an der Stelle ihres ersten Auftretens im Inhalt zu erläutern und durch die hierfür vorgesehenen Elemente der verwendeten Markup-Sprache kenntlich zu machen.			
	4.3	Die vorherrschend verwendete natürliche Sprache ist durch die hierfür vorgesehenen Elemente der verwendeten Markup-Sprache kenntlich zu machen.			
Anforderung	5	Tabellen sind mittels der vorgesehenen Elemente der verwendeten Markup-Sprache zu beschreiben und in der Regel nur zur Darstellung tabellarischer Daten zu verwenden.			
Bedingung	5.5	Für Tabellen sind unter Verwendung der hierfür vorgesehenen Elemente der genutzten Markup-Sprache Zusammenfassungen bereitzustellen.			
	5.6	Für Überschriftenzellen sind unter Verwendung der hierfür vorgesehenen Elemente der genutzten Markup-Sprache Abkürzungen bereitzustellen .			
Anforderung	6	Internetangebote müssen auch dann nutzbar sein, wenn der verwendete Benutzeragent neuere Technologien nicht unterstützt oder diese deaktiviert sind.			
Anforderung	7	Zeitgesteuerte Änderungen des Inhalts müssen durch die Nutzerin, den Nutzer kontrollierbar sein.			
Anforderung	8	Die direkte Zugänglichkeit der in Internetangeboten eingebetteten Benutzerschnittstellen ist sicherzustellen.			
Anforderung	9	Internetangebote sind so zu gestalten, dass Funktionen unabhängig vom Eingabegerät oder Ausgabegerät nutzbar sind.			
Bedingung	9.4	Es ist eine mit der Tabulatortaste navigierbare, nachvollziehbare und schlüssige Reihenfolge von Hyperlinks, Formularkontrollelementen und Objekten festzulegen.			

			J	N	kA
	9.5	Es sind Tastaturkurzbefehle für Hyperlinks, die für das Verständnis des Angebots von entscheidender Bedeutung sind (einschließlich solcher in clientseitigen Imagemaps), Formularkontrollelemente und Gruppen von Formularkontrollelementen bereitzustellen.			
Anforderung	10	Die Verwendbarkeit von nicht mehr dem jeweils aktuellen Stand der Technik entsprechenden assistiven Technologien und Browsern ist sicherzustellen, so weit der hiermit verbundene Aufwand nicht unverhältnismäßig ist.			
Bedingung	10.3	Für alle Tabellen, die Text in parallelen Spalten mit Zeilenumbruch enthalten, ist alternativ linearer Text bereitzustellen.			
	10.4	Leere Kontrollelemente in Eingabefeldern und Textbereichen sind mit Platzhalterzeichen zu versehen.			
	10.5	Nebeneinanderliegende Hyperlinks sind durch von Leerzeichen umgebene, druckbare Zeichen zu trennen.			
Anforderung	11	Die zur Erstellung des Internetangebots verwendeten Technologien sollen öffentlich zugänglich und vollständig dokumentiert sein, wie z.B. die vom World Wide Web Consortium entwickelten Technologien.			
Bedingung	11.4	Der Nutzerin, dem Nutzer sind Informationen bereitzustellen, die es ihnen erlauben, Dokumente entsprechend ihren Vorgaben (z.B. Sprache) zu erhalten.			
Anforderung	12	Der Nutzerin, dem Nutzer sind Informationen zum Kontext und zur Orientierung bereitzustellen.			
Anforderung	13	Navigationsmechanismen sind übersichtlich und schlüssig zu gestalten.			
Bedingung	13.5	Es sind Navigationsleisten bereitzustellen, um den verwendeten Navigationsmechanismus hervorzuheben und einen Zugriff darauf zu ermöglichen.			

			J	N	kA
	13.6	Inhaltlich verwandte oder zusammenhängende Hyperlinks sind zu gruppieren. Die Gruppen sind eindeutig zu benennen und müssen einen Mechanismus enthalten, der das Umgehen der Gruppe ermöglicht.			
	13.7	Soweit Suchfunktionen angeboten werden, sind der Nutzerin, dem Nutzer verschiedene Arten der Suche bereitzustellen.			
	13.8	Es sind aussagekräftige Informationen am Anfang von inhaltlich zusammenhängenden Informationsblöcken (z.B. Absätzen, Listen) bereitzustellen, die eine Differenzierung ermöglichen.			
	13.9	Soweit inhaltlich zusammenhängende Dokumente getrennt angeboten werden, sind Zusammenstellungen dieser Dokumente bereitzustellen.			
	13.10	Es sind Mechanismen zum Umgehen von ASCII-Zeichnungen bereitzustellen.			
Anforderung	14	Das allgemeine Verständnis der angebotenen Inhalte ist durch angemessene Maßnahmen zu fördern.			
Bedingung	14.2	Text ist mit graphischen oder Audio-Präsentationen zu ergänzen, sofern dies das Verständnis der angebotenen Information fördert.			
	14.3	Der gewählte Präsentationsstil ist durchgängig beizubehalten.			

Legende: J = Ja, N = Nein, kA = keine Angabe

Abkürzungen

AbI	Aktionsbündnis für barrierefreie Informationstechnik
AJaX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASP	Active Server Pages
BDSG	Bundesdatenschutzgesetz
BSI	Bundesamt für Sicherheit in der Informationstechnik
BITV	Barrierefreie Informationstechnik-Verordnung
CIAC	Computer Incident Advisory Capability
COM	Common Object Model
CSS	Cascading Style Sheet
Daemon	Disk And Execution MONitor
DDoS	Distributed Denial of Service Angriff
DNS	Domain Name System
DoS	Denial of Service
EGG	Gesetz zum elektronischen Geschäftsverkehr
FAQ	Frequently Asked Questions
FMEA	Failure Mode and Effects Analysis
FTP	File Transfer Protocol

GjS	Gesetz über die Verbreitung jugendgefährdender Schriften und Medieninhalte
GPL	General Puplic License
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer protocol
IuKT	Informations- und Kommunikationstechnologien
JAR	Java-Archiv
JDBC	Java Database Connectivity
JÖSchG	Gesetz zum Schutze der Jugend in der Öffentlichkeit
JSP	Java Server Pages
JMStV	Jugendmedienschutzstaatsvertrag
JuSchG	Jugendschutzgesetz
LDAP	Lightweight Directory Access Protocol
MarkenG	Markengesetz
MDStV	Mediendienste-Staatsvertrag
MitM	Man-in-the-Middle
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
OWASP	Open Web Application Security Project
PAngV	Preisangabenverordnung
PDF	Portable Document Format
PHP	Private Home Page (Web-Programmiersprache)
QR	Queued Requests
QS	Qualitätssicherung

RPC	Remote Procedure Call
RPS	Requests Per Second
RPZ	Risikoprioritätszahl
SAGA	Standards und Architekturen für E-Government-Anwendungen
SEO	Search Engine Optimization
SigG	Signaturgesetz
SigV	Signaturverordnung
SOPA	Simple Object Access Protocol
TDDSG	Teledienstedatenschutzgesetz
TDG	Teledienstegesetz
TDSV	Telekommunikations-Datenschutzverordnung
TKG	Telekommunikationsgesetz
UML	Unified Modeling Language
UrhG	Urheberrechtsgesetz
URL	Uniform Ressource Locator
W3C	World Wide Web Consortium
WAI	Web Accessibility Initiative
WAR	Web-Archive
Web	World Wide Web (Kurzform)
WWW	World Wide Web
XHTML	Extensible Hypertext Markup Language
ZKDSG	Zugangskontrolldiensteschutz-Gesetz

Synonyme

Änderbarkeit = Wartbarkeit
Anforderungen = Requirements
Ausfallrate = Unverfügbarkeit
Accessibility = Barrierefreiheit = Zugänglichkeit
Capture Replay Tool = Testroboter
Gebrauchstauglichkeit = Usability
Komponententest = Modultest = Unit-Test
Metrik = Maß
Performanz = Zeitverhalten
Risikoklasse = Risikostufe
Testart = Testtyp
Testmethode = Testverfahren
Testabdeckung = Testüberdeckung
Teststufe = Testphase
Übertragbarkeit = Portabilität
Whitebox-Test = Strukturtest
Web-Anwendung = Web-Applikation
zustandsbasierter Test = zustandsbezogener Test
Zustandsdiagramm = Zustandübergangsdiagramm

Glossar[1]

.NET

.NET ist eine Software-Technologie von Microsoft. Sie stellt eine virtuelle Laufzeitumgebung sowie ein Rahmenwerk von Klassenbibliotheken und Diensten für die Software-Entwicklung zur Verfügung.

ActiveX

ActiveX ist ein Software-Komponentenmodell von Microsoft, das es ermöglicht, aktive Objekte (Video, Sound,...) in Dokumente wie zum Beispiel Web-Seiten einzubauen.

AJaX – Asynchronous JavaScript and XML

AJaX ist eine Technik, mit der bei aktiviertem JavaScript Daten vom Server nachgeladen werden können, ohne dass die gesamte Seite neu geladen werden muss. Durch die asynchrone Datenübertragung werden die Wartezeiten für den Benutzer minimiert.

Clustered Server

Ein Clustered Server ist die Kopplung mehrerer Server zu einem virtuellen Server. Dadurch erhöht sich sowohl die Leistungsfähigkeit als auch die Ausfallsicherheit des Systems.

Cookie

Ein Cookie (engl. Keks) ist eine kleine Datei, die lokal auf dem Rechner des Nutzers einer Web-Anwendung abgelegt wird und in der Informationen abgespeichert werden, die im Zusammenhang mit der aktuellen Website stehen.

In einem Cookie sind Nutzdaten und Informationen darüber abgelegt, wer das Cookie gesetzt hat und wie lange es gültig bleiben soll. Cookies haben eine Lebensdauer von mehreren Tagen oder Wochen.

[1] In diesem Glossar sind nur Begriffe aufgenommen, die nicht im Text des Buches definiert und dort mit [GL] gekennzeichnet sind.

Sogenannte Session-Cookies sind nur so lange aktiv, wie der Browser geöffnet ist.

Ein Cookie kann nur von Servern ausgelesen werden, die den gleichen Domain-Namen haben wie der Server, der das Cookie geschrieben hat. Ein Cookie kann maximal 4 KB groß sein.

COM – Common Object Model

COM ist Microsofts Software-Architektur zur Kommunikation von Software-Komponenten.

CSS – Cascading Style Sheet

Cascading Style Sheets erweitern durch spezielle Formatanweisungen die Möglichkeiten von HTML, Websites zu gestalten. Sie ermöglichen die Trennung von Inhalt und Layout bei der Programmierung von Web-Seiten. Text- und Darstellungsattribute können in Klassen zusammengefasst und mit HTML-Seiten verknüpft werden.

Daemon – Disk And Execution MONitor

Ein Daemon ist ein Programm in einem Netzwerk, das für den Benutzer unsichtbar im Hintergrund auf bestimmte Ereignisse wartet. Tritt ein solches Ereignis ein, wird eine bestimmte Aktion des Daemons ausgelöst. Zum Beispiel bearbeitet ein Mailer-Daemon im Hintergrund den Eingang von E-Mails.

DNS – Domain Name System

DNS ist ein Dienst im Internet und anderen TCP/IP-Netzen, der den Namensraum im Internet verwaltet, d.h. für einen Hostnamen die entsprechende IP-Adresse zurückgibt und umgekehrt.

Gateway

Ein Gateway ist eine Hard- und/oder Softwarelösung, die eine Verbindung von inkompatiblen Netzwerken schafft.

Hyperlink

Ein Hyperlink ist die Verbindung eines Elements eines Dokumentes zu einer anderen Stelle im selben Dokument oder zu einem anderen Dokument.

Interoperabilität

Interoperabilität ist die Fähigkeit einer Software-Komponente, mit einer oder mehreren anderen definierten Software-Komponenten zusammenzuarbeiten. Die Interoperabilität stellt die gemeinsame Nutzung von Daten, Programmen und Prozessen sicher.

JDBC – Java Database Connectivity

JDBC ist eine Sammlung von Klassen, die es Java-Applikationen ermöglicht, mit Hilfe von relationalen Datenbankobjekten und den entsprechenden Methoden auf Datenbanken zuzugreifen.

Java-Applet

Ein Java-Applet ist ein in eine Web-Seite eingebettetes Java-Programm. Es wird vom Server auf den Client übertragen und dort innerhalb des Web-Browsers ausgeführt.

Java-Archiv

Ein Java-Archiv (JAR-Datei, Dateiendung „.jar“) ist ein plattformunabhängiges Dateiformat, in dem verschiedene Dateien zusammengefasst werden können, die zur Ausführung eines Java-Applets nötig sind.

Load Balancer

Ein Load Balancer verteilt die anfallende Last auf mehrere Web-Server, die zu einem virtuellen Server zusammengeschlossen sind. Load Balancer existieren in Form von Hardware, Software und als Kombination von beidem. Sie werden paarweise eingesetzt, damit sie nicht selbst zur Schwachstelle im System werden.

Mediendienste-Staatsvertrag (MDStV)

Zweck des Mediendienste-Staatsvertrages ist, in allen Ländern einheitliche Rahmenbedingungen für die verschiedenen Nutzungsmöglichkeiten der geregelten elektronischen Informations- und Kommunikationsdienste zu schaffen.

PDF – Portable Document Format

PDF ist ein plattformübergreifendes Dateiformat für Dokumente, das von der Firma Adobe Systems entwickelt wurde.

Proxy-Server

Proxy-Server dienen als Schnittstelle zwischen dem Internet und dem lokal eingesetzten Rechner. Zum einen sorgen sie für schnellere Datenbereitstellung, weil sie eine einmal aufgerufene Datei zwischenspeichern und sie bei einem erneuten Aufruf schneller bereitstellen. Zum anderen erhöhen sie die Sicherheit, denn die im Internet verbundenen Komponenten wissen zum Beispiel nicht, mit welchem Rechner aus dem internen Netzwerk sie verbunden sind, weil sie nur die Adresse des Proxy-Servers kennen.

Relay-Host

Ein Relay-Host ist ein wesentlicher Bestandteil von Firewall-Sicherheitssystemen. In einem durch Firewalls geschütztem Netz ist der Relay-Host der einzige Host, der eine Verbindung zum Internet herstellen kann.

Sitemap

Unter einer Sitemap versteht man eine besondere Übersicht der Navigationsstruktur einer Website, zum Beispiel in der Darstellung des Windows-Explorers.

Swing (Java-Swing)

Swing ist eine von Sun Microsystems für Java entwickelte Schnittstelle zur vereinfachten Programmierung von grafischen Benutzeroberflächen.

Teledienstegesetz (TDG)

Gesetz über die Nutzung von Telediensten, Zweck des Gesetzes ist es, einheitliche wirtschaftliche Rahmenbedingungen für die verschiedenen Nutzungsmöglichkeiten der elektronischen Informations- und Kommunikationsdienste zu schaffen.

Telekommunikations-Datenschutzverordnung (TDSV)

Diese Verordnung regelt den Schutz personenbezogener Daten der an der Telekommunikation Beteiligten bei der Erhebung, Verarbeitung und Nutzung dieser Daten durch Unternehmen und Personen, die geschäftsmäßig Telekommunikationsdienste erbringen oder an deren Erbringung mitwirken.

Telekommunikationsgesetz (TKG)

Zweck dieses Gesetzes ist es, durch Regulierung im Bereich der Telekommunikation den Wettbewerb zu fördern und flächendeckend angemessene und ausreichende Dienstleistungen zu gewährleisten sowie eine Frequenzordnung festzulegen.

Thread

Ein Thread (engl. thread = Faden) ist ein Programmteil, der zeitlich unabhängig von anderen Programmteilen abläuft.

Transaktion

Eine Transaktion ist ein ergebnisorientierter Kommunikations- oder Verarbeitungsabschnitt. Er besteht in der Regel aus einer Folge von

Nachrichten oder Operationen, die entweder alle übermittelt bzw. ausgeführt werden oder alle auf den Zustand vor dem Beginn der Transaktion zurückgesetzt werden. Das System ist vor Beginn und nach Ende einer Transaktion in einem konsistenten Zustand.

Web-Archiv

Ein Web-Archiv (WAR-Datei, Dateiendung „.war“) ist eine Datei im JAR- bzw. ZIP-Format, die eine vollständige Java-Web-Anwendung enthält.

XHTML – Extensible Hypertext Markup Language

XHTML ist eine Auszeichnungssprache für das World Wide Web. Das "X" steht für "Extensible", weil XHTML-Dokumententypen auf XML (Extensible Markup Language) basierend erweiterbar sind.

XML – Extensible Markup Language

XML ist ein Standard zur Erstellung maschinen- und menschenlesbarer Dokumente in Form einer Baumstruktur, der vom World Wide Web Consortium (W3C) definiert wird. XML definiert dabei die Regeln für den Aufbau solcher Dokumente. (aus [URL: wiki])

X-Windows

X-Windows ist ein Standard zum Betrieb von Grafikbildschirmen.

Quellen

Literatur

[Beier_2002] Beier, M.; Gizycki von, V.: Usability – Nutzerfreundliches Web-Design, Springer Verlag, 1. Auflage, Berlin, 2002

[Bergmann_2005] Bergmann, S.: PHPUnit kurz & gut, O'Reilly Verlag, 1. Auflage, 2005

[Hamill_2004] Hamill, P.: Unit Test Frameworks – Tools for High-Quality Software Development, First Edition, O'Reilly Verlag, 2004

[Hörmann_2006] Hörmann, K.; Dittmann, L.; Hindel, B.; Müller, M.: SPICE in der Praxis – Interpretationshilfe für Anwender und Assessoren, dpunkt.verlag, 1. Auflage, Heidelberg, 2006

[Kneuper_2003] Kneuper Dr., R.: CMMI – Verbesserung von Softwareprozessen mit Capability Maturity Model Integration; dpunkt.verlag, 1. Auflage, Heidelberg, 2003

[Lettau_2000] Lettau, C.: Das Webpflichtenheft, mitp Verlag, 1. Auflage, Bonn, 2000

[Myers_1987] Myers, G. J.: Methodisches Testen von Programmen: 2. Auflage, München Wien, R. Oldenbourg Verlag, 1987

[Spillner_2005] Spillner, A.; Linz, T.: Basiswissen Softwaretest; dpunkt.verlag, 3. Auflage, Heidelberg, 2005

[Spillner_2006] Spillner, A.; Roßner, T.; Winter, M.; Linz, T.: Praxiswissen Softwaretest – Testmanagement; dpunkt.verlag, 1. Auflage, Heidelberg, 2006

[Westphal_2005] Westphal, F.: Testgetriebene Entwicklung mit JUnit & FIT – Wie Software änderbar bleibt, dpunkt.verlag, 1. Auflage, Heidelberg, 2005

Normen und Standards

[DIN 25448] DIN 25448: 1990-05
Ausfalleffektanalyse (Fehler-Möglichkeits- und -Einfluss-Analyse)

[DIN 40041] DIN 40041: 1990-12
Zuverlässigkeit; Begriffe

[DIN 66272] DIN 66272: 1994-10
Bewerten von Softwareprodukten – Qualitätsmerkmale und Leitfaden zu ihrer Verwendung

[IEEE 1219] IEEE 1219: 1998
Standard for Software Maintenance, Institute of Electrical and Electronics Engineers, 1998

[ISO 9241-10] DIN EN ISO 9241-10: 1996
Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Teil 10: Grundsätze der Dialoggestaltung

[ISO 9241-11] DIN EN ISO 9241-11: 1998
Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Teil 11: Anforderungen an die Gebrauchstauglichkeit

[ISO/IEC 9126] ISO/IEC 9126-1: 2001-06, Software-Engineering - Qualität von Software-Produkten – Teil 1: Qualitätsmodell

[ISTQB_2006] Standard glossary of terms used in Software Testing, Editor : Erik van Veenendaal, Version 1.2, Produced by the "Glossary Working Party" – International Software Testing Qualification Board, 2006,

URLs[1]

[URL: AbITools]
http://www.wob11.de/programme.html
Werkzeuge zum Zugänglichkeitstest, Aktionsbündnis für barrierefreie Informationstechnik (AbI)

[URL: Abmahnwelle]
http://www.abmahnwelle.de
Forschungsstelle Abmahnwelle e.V.

[1] Die in diesem Buch aufgeführten URLs wurden mit Drucklegung überprüft. Eine Garantie für die Gültigkeit über dieses Datum (Dezember 2006) hinaus kann nicht übernommen werden.

[URL: ALLPAIRS]
http://www.satisfice.com/tools.shtml
ALLPAIRS Test Case Generation Tool, James Bach, Satisfice, Inc.

[URL: Beuth]
http://www.beuth.de/
Beuth Verlag GmbH

[URL: BGG]
http://www.wob11.de/bgg.html
Auszüge aus dem Behindertengleichstellungsgesetz des Bundes (BGG), Web ohne Barrieren nach Paragraph 11 im Bundesbehindertengleichstellungsgesetzes, vom Aktionsbündnis für barrierefreie Informationstechnik (AbI)

[URL: BITV]
http://www.wob11.de/bitv.html
Barrierefreie Informationstechnik-Verordnung (BITV), Web ohne Barrieren nach Paragraph 11 im Bundesbehindertengleichstellungsgesetzes, vom Aktionsbündnis für barrierefreie Informationstechnik (AbI)

[URL: BITVCL]
http://www.wob11.de/checklisten.html
Checklisten zur Barrierefreien Informationstechnik Verordnung (BITV), Web ohne Barrieren nach Paragraph 11 im Bundesbehindertengleichstellungsgesetzes, vom Aktionsbündnis für barrierefreie Informationstechnik (AbI)

[URL: Browser-Archiv]
http://browsers.evolt.org/
Browser-Archiv von evolt.org

[URL: Browser-Statistik]
http://www.joergkrusesweb.de/internet/browser/statistik/
Browser-Statistik von Jörg Kruse, Göttingen

[URL: BSI]
http://www.bsi.bund.de
Bundesamt für Sicherheit in der Informationstechnik (BSI)

[URL: CaseMaker]
http://www.casemaker.de/
CaseMaker, Testspezifikationswerkzeug, Díaz & Hilterscheid Unternehmensberatung GmbH

[URL: CASE-Tools]
http://de.wikipedia.org/wiki/Computer-Aided_Software_Engineering
CASE-Tool-Übersicht, Wikipedia – Stichwort „Computer-Aided Software Engineering"

[URL: Certiorina]
http://www.certiorina.de/
Formular zur Erstellung eines Web-Impressums, Herausgeber: Abmahnwelle.de (siehe [URL: Abmahnwelle])

[URL: Compuware]
http://www.compuware.com/products/qacenter/qaload.htm
QALoad, Load Test Tool, Compuware Corporation

[URL: Ergo]
http://www.sozialnetz-hessen.de/ca/ph/het/hauptpunkt/aaaaaaaaaaahfi/hauptframeid/aaaaaaaaaaahfi/hauptframetemplate/aaaaaaaaaaaapk/
Informationsdienst Arbeit und Gesundheit,
Schwerpunkt Bildschirmarbeit – Software

[URL: GesetzeBMJ]
http://www.gesetze-im-internet.de/
Gesetze in Internet, Das Bundesministerium der Justiz mit der juris GmbH

[URL: GesetzeBMWI]
http://www.bmwi.de/BMWi/Navigation/Service/gesetze.html
Gesetze und Verordnungen, Bundesministerium für Wirtschaft und Technologie

[URL: GesetzeIID]
http://www.iid.de/
Informationen zur IuKT-Förderung des Bundesministeriums für Bildung und Forschung (BMBF)

[URL: HTMLkit]
http://www.chami.com/html-kit/
HTML-Kit, freier HTML Editor, chami.com

[URL: JMeter]
http://jakarta.apache.org/jmeter/
Apache JMeter, Load Test Tool, The Apache Jakarta Projekt - The Apache Software Foundation

[URL: JUnit01]
http://www.junit.org
JUnit, Testframework, JUnit.org

[URL: JUnit02]
http://www.frankwestphal.de/UnitTestingmitJUnit.html
Deutsche Dokumentation zu JUnit von Frank Westphal

[URL: LFDN]
http://www.lfd.niedersachsen.de/master/C27772_N13184_L20_D0_I560.html, Der Landesbeauftragte für den Datenschutz Niedersachsen – Service-Angebote – Checklisten – Firewall; mit Link auf die Checkliste: Grundschutz durch Firewall – Orientierungshilfe und Checkliste, 1999: http://cdl.niedersachsen.de/blob/images/C467769_L20.pdf

[URL: Lynx]
http://lynx.browser.org/
Lynx, textbasierter Webbrowser

[URL: Mercury]
http://www.mercury.com/de/products/
Produktseite von Mercury Deutschland, Mercury Interactive Corporation

[URL: OWASP]
http://www.owasp.org
The Open Web Application Security Project (OWASP)

[URL: OWASP-CL]
https://sourceforge.net/project/showfiles.php?group_id=64424&package_id=62285
OWASP Web Application Penetration Checklist, Version 1.1

[URL: phpOpenTracker]
http://www.phpopentracker.de/
phpOpenTracker, User Tracking Tool, Sebastian Bergmann

[URL: PHPUnit]
http://www.phpunit.de
PHPUnit, Testframework, Sebastian Bergmann

[URL: PluginDoc]
http://plugindoc.mozdev.org/de-DE/
Firefox-Plugins, Deutsche Seite von PluginDoc, mozdev.org

[URL: Plugin-Java]
http://java.sun.com/products/plugin/
Java Plug-in Technology, Sun Microsystems, Inc.

[URL: Plugin-SAGA]
http://www.kbst.bund.de/cln_006/nn_838698/Content/Standards/Saga/Plugins/plugins.html__nnn=true
Positivliste Plugins, Standards und Architekturen für E-Government-Anwendungen (SAGA), Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern (KBSt).

[URL: QCHESS]
http://www.q-chess.de
Q-Chess, web-basiertes Checklistenverwaltungssystem, G. Muth Partners GmbH

[URL: Rational]
http://www-306.ibm.com/software/de/rational/design.html
Rational Design- und Entwicklungswerkzeuge, Rational Software, IBM Deutschland GmbH

[URL: SecurityTools]
http://ciac.llnl.gov/ciac/tools.html
CIAC (Computer Incident Advisory Capability) Homepage des U.S. Department of Energy

[URL: Selenium]
http://www.openqa.org/selenium/
Selenium, Capture Replay Tool für Web-Applikationen, OpenQA

[URL: SEO-Google]
http://www.google.com/intl/de/webmasters/guidelines.html
Google-Informationen für Webmaster, Google

[URL: SEO-MSN]
http://search.msn.de/docs/siteowner.aspx?FORM=WMDD2
Hilfe für Seiteneigentümer, MSN Deutschland & Windows Live – Microsoft Corporation

[URL: Sitewalker]
http://www.jstudio.de/
Jstudio SiteWalker Professional, Capture Replay Tool für Web-Applikationen, Jstudio – Jarsch Software Studio

[URL: SZ]
http://www.sueddeutsche.de/wirtschaft/artikel/400/68332/
sueddeutsche.de GmbH / Süddeutsche Zeitung GmbH, 18.01.2006 09:32 Uhr

[URL: TestToolsFAQ]
http://www.testingfaqs.org/
testingfaqs.org, Informationsseite für Software-Tester, Danny Faught

[URL: TestToolsJava]
http://java-source.net/
Open Source Software in Java

[URL: TestToolsOPENSOURCE]
http://www.opensourcetesting.org/
Open source tools for software testing professionals

[URL: TestToolsSQA]
http://www.softwareqatest.com/qatweb1.html
Software QA/Test Resource Center, Rick Hower

[URL: Tidy]
http://tidy.sourceforge.net/
HTML Tidy Library Project, SourceForge.net

[URL: UML-OMG]
http://www.uml.org/
UML Resource Page, Objekt Management Group

[URL: V-Modell]
http://www.v-modell-xt.de/
Aktuelles V-Modell XT der Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern (KBSt)

[URL: W3C]
http://www.w3.org
World Wide Web Consortium

[URL: W3C-checklink]
http://validator.w3.org/checklink
W3C Link Checker

[URL: W3C-validatorCSS]
http://jigsaw.w3.org/css-validator/
W3C CSS-Validierungsservice

[URL: W3C-validatorMarkup]
http://validator.w3.org/
W3C Markup Validation Service

[URL: W3C-validatorXML]
http://www.w3.org/2001/03/webdata/xsv
W3C Validator for XML Schema

[URL: WAS]
http://www.microsoft.com/technet/archive/itsolutions/intranet/downloads/webstres.mspx
Web Application Stress Tool, Microsoft Corporation

[URL: WAI-CL]
http://www.w3.org/TR/WCAG10/full-checklist.html
Checklist of Checkpoints for Web Content Accessibility Guidelines 1.0, Checkliste Zugänglichkeitsrichtlinien der Web Accessibility Initiative (WAI) des W3C

[URL: WAI-CLdt]
http://www.w3c.de/Trans/WAI/checkliste.html
Checkpunkte zu den Zugänglichkeitsrichtlinien für Web-Inhalte 1.0, Deutsche Übersetzung von René Hartmann, W3C Deutsch-Österreichisches Büro

[URL: WAI-Tools]
http://www.w3.org/WAI/ER/tools/complete
Complete List of Web Accessibility Evaluation Tools, Web Accessibility Initiative (WAI) des W3C

[URL: WAVE]
http://wave.webaim.org/
WAVE Accessibility Tool, Temple University Institute on Disabilities

[URL: wiki]
http://de.wikipedia.org/wiki/
Wikipedia - Die freie Enzyklopädie

[URL: wiki-Blick]
http://de.wikipedia.org/wiki/Blickbewegung
Wikipedia, Stichwort „Blickbewegung“

[URL: XenuLink]
http://home.snafu.de/tilman/xenulink.html
Xenu's Link Sleuth, Link Checker von Tilman Hausherr

Sachverzeichnis

.

.NET 263

A

ABC-Risikoanalyse 65
Abnahmekriterien 234
Abnahmetest 233
Abstandsklausel 130
Accessibility 18
ActiveX 263
AJaX 263
Akzeptanz 123
Alpha-Test 233
Änderbarkeit 14, 172
Anforderung
- funktionale 25
- nicht-funktionale 25
Anforderungsabdeckung 49
Anforderungsdokument
- fachlich 76
- technisch 76
Anforderungsmanagement-werkzeug 225
Anwendungsfall 46
Anwendungsfalldiagramm 46
Äquivalenzklasse 33
- Kombinationen 38
- unzulässige 33
- zulässige 33
Äquivalenzklassenanalyse 33
Attribute Coverage 56
Auffindbarkeit 18, **160**
Auffindbarkeitstest 160
Aufgabenangemessenheit 15
Aufklärungspflicht 124, 128
Ausfalleffektanalyse 64
Ausfallrate 201
Ausfallrisikoanalyse 200
Ausfallsicherheit 195
Ausfallsicherheitstest **197**, 222

B

Backup-Firewall 200
Backup-Komponente 195
barrierefrei 155
Barrierefreie Informationstechnik-Verordnung 156
Barrierefreiheit 18
Bedingungsabdeckung 52
- einfache 52
- mehrfache 52
- minimal mehrfache 56
Befragung 150
Behindertengleichstellungsgesetz 18
Benutzbarkeit 14
Benutzerverhalten 149
Benutzerwartungen 124
Beta-Test 233
Bewertungsbogen zur Usability 150, **243**

BITV 156
Blackbox-Test **28**, 93
Blackbox-Verfahren 28, **31**, 167
Blickpfad 153
Blickregistrierung 152
bottle necks 183
Branch Coverage 52
Browser/Betriebssystem-Risikomatrix 140
Browser-Archiv 146
Browser-Statistik 140, 147
Browser-Test 139
Brute Force Angriff 115

C

Capture Replay Tool 94, 101, 112, 137, 146, 177, **213**
Cascading Style Sheet 137, **264**
CASE-Tool 84
Checkfragen 71
Checkliste 69
- Aufklärungspflicht 128
- Ausfallsicherheitstest 198
- BITV Priorität I 245
- BITV Priorität II 252
- Browser-Einstellungen 144
- Browser-Test - Stichprobe 145
- Code-Inspektion 167
- Cookie-Test 107
- Datenmodell 79
- Dokument 78
- Fehlererwartung 60
- Firewall (extern) 119
- Installationstest 175
- Link-Test 105
- Oberflächentest 132
- Penetrationstest (extern) 119
- Performanz-/Lasttests - Planung 190
- Pilotbetrieb - Planung 235
- Plugin-Test 111
- Sicherheitstest 115
- Web-Adresse 160
- Web-Content 125
- Web-Recht 127
- Zugänglichkeit des BITV (extern) 156
- Zugänglichkeit des WAI (extern) 156
Checklisten
- Bereitstellung 223
Clustered Server 263
Code Analyzer 172
Code Coverage Tool **57**, 94
Code-Analyse 83, **165**, 222
- statische 171
Code-Inspektion 83, **167**
Code-Walkthrough 83, **166**
Common Object Model 264
Compiler 171
Condition Coverage 52
Content 124
Content-Test 83, 106, **124**
Cookie 106, **263**
Cookie Viewer 108
Cookie-Test 106
CSS 137, **264**

D

Daemon 264
Datendurchsatz 189
Datenmanipulationswerkzeug 225
DDoS 115
Decision and Branch Coverage 51
Deep Link 104
Deinstallierbarkeit 174
Denial of Service Angriff 115
DIN 66272 14
Distributed Denial of Service Angriff 115
Dokument
- produktbezogen **76**, 83
- prozessbezogen 76
Dokumententest **75**, 165, 221
Domain Name System 264
DoS 115
Download-Absicherung 130

E

Effektivität 17
Effizienz 14, 17

Ein-Personen-Code-Inspektion 171
Ein-Personen-Walkthrough 171
Entdeckungswahrscheinlichkeit 64
Entscheidungstabelle 38
Entwicklertest 88
Erfüllungsgrad einer Checkliste 70
error 24
Error Guessing 93
Erwartungskonformität 15
Eye Tracker 153
Eye Tracking 152

F

Failback 197
Failback-Test 197
Failover 195
Failover-Test 197
Failover-Verfahren 196
failure 24
Failure Mode and Effects Analysis 64
Farbkontrast-Analyzer 158
fault 24
Fehler
- äußerer 24
- innerer 24
Fehlererwartung 60
Fehlererwartungstest 93
Fehlermanagementwerkzeug 225
Fehlermaskierung **24**, 34
Fehlermöglichkeits- und Einflussanalyse 64
Fehlernachtest 208
Fehlertoleranz 16
Fehlerursache 24
Fehlerwirkung 24
Fehlhandlung 24
Fixation 154
FMEA 64
Formale Dokumentenprüfung 78
Fragebogen zum Nutzungsverhalten 150, **242**
Framing 104
Funktionalität 14

G

Gateway 264
Gebrauchstauglichkeit 15, **17**, 18, 148
Gesetze, im Web zu beachtende 127
Grenzwert 36
Grenzwertanalyse 36
Grenzwerttestfälle 38
Greybox 29
Greybox-Test **28**, 102
Grundsätze der Dialoggestaltung 15, 132
Gruppensitzung 80

H

Hauptqualitätsmerkmale 14, 22
Homepage 25
Hyperlink 104, **264**

I

Inbound Link 162
Individualisierbarkeit 16, 124
Inline Link 104
Inspektionssitzung 167
Installationsroutine 174
Installationstest 174
Installierbarkeit 174
Integration
- Ad-hoc 95
- Anwendungsfallorientiert 95
- Backbone 95
- Big Bang 95
- Bottom-up 95
- Top-down 95
Integrationstest 95, 222
Integrationsverfahren 95
Interface Coverage 57
Interoperabilität 112, **264**
ISO 8402 13
ISO 9241
- Teil 10 15, 124
- Teil 11 17
ISO/IEC 9126 14

J

Java-Applet 265
Java-Archiv 265
Java-Swing 266
JDBC 265
JUnit 91

K

Klasseneinzeltest 88
Klassenintegrationstest 89
Klassentest 88
Komparator 225
Komponente 92
Komponententest **92**, 222
Konfigurationsmanagement-werkzeug 225

L

Lasttest 182
Lasttestwerkzeug 184, **192**, 212
Lernförderlichkeit 16, 124
Link 104
- deep 104
- extern 104
- inline 104
- intern 104

Link-Checker 105
Link-Test 104
Load Balancer 196, 265
Load Balancing 189
Load Test Tool 184, **192**, 212
Logic Coverage Test 51
Loop Coverage 56

M

Man-in-the-Middle-Angriff 115
Maß 27
Massentest 183
Mean Time Between Failure 201
Mean Time To Repair 201
memory leak 184
Mengenverarbeitung 180
Metatag 104
Method Coverage 56
Methodenabdeckung 91
Metrik 27
Metriken 186
Moderator 81
Modultest 92
Monitore 101
MTBF 201
MTTR 201

N

Nutzergruppen 148, 181
Nutzungsverhalten, Fragen zum 242

O

Oberflächentest 131
Online-Fragebogen 152
Online-Umfrage 152
Open Web Application Security Project 118, 119

P

Page Ranking Tool 162
Pairwise-Methode 40
Parameter Coverage 57
Path Coverage 56
PDF 265
Penetrationstest 114
Performanz 180
Performanz-/Lasttests 180
Performanztest 180
Pfadabdeckung 56
Pilotanwender 235
Pilotbetrieb 235
Pilotphase 235
Planung der Testzyklen 237
Platzhalter 94
Plugin 109
Plugin-Test 109
Produktrisiko 64
Profiler 193
Programmcode 165
Projektmanagementwerkzeug 225
Projektrisiko 64
Proxy-Server 265
Prozessorauslastung 186, 187

Prüfung 23
- dokumentenbezogen 77
- formal 78
- inhaltsbezogen 77
Pseudocode 165

Q

Qualitätsanforderung 19
Qualitätsmaß 19
Qualitätsmerkmal 19
Qualitätsmerkmale
- funktionale 14
- für Web-Applikationen 17
- nicht funktionale 20
Qualitätssicherungsmaßnahme
- analytisch 21
- konstruktiv 21
Queued Requests 186

R

Ranking 161
Rechtmäßigkeit 124
Rechtskonformität 18, 127
Redundanzen 195
Regressionstest 208
- nach Fehlerbehebung 209
- nach Funktionsveränderungen 209
- nach Systemveränderungen 210
Regressionstest-Tool 213
Relay-Host 266
Relevanz 161
Reliability 201
Repräsentant (Testdaten) 33
Request 188
Requests Per Second (RPS) 186
Ressourcenverbrauch 180
Restart-/Recovery-Test 198
Restart-/Recovery-Verfahren 198
Review 76
Reviewmanagement-Werkzeug 70
Review-Protokoll 81
Review-Regeln 80
Review-Sitzung 80
Richtlinien zur Namensgebung 162
Risiko 64
Risikoklasse 65
Risikoklassen
- Browser-Test 141
- Dokumente 65
- Programme 66
- Testfallabdeckung 49
Risikoprioritätszahl (RPZ) 64
Risikostufe 65
Rückfalllösung 212

S

SAGA-Plugins 113
Sakkade 154
Schleifenabdeckung 56
Schreibtischtest 171
schriftliche Stellungnahme 82
Schwachstellen 183
Search Engine Optimization 161
Security Test Tool 119
Selbstbeschreibungsfähigkeit 16
Selenium 214
SEO 161
SEO-Tool 162
Session-Cookies 264
Session-Hijacking 115
Sicherheit 113
Sicherheitsfunktionen 113
Sicherheitsmechanismen 113
Sicherheitstest 113
Sicherheitsvorkehrungen 113
Sitemap 266
Skalierbarkeit 183
Skalierbarkeitstest 183
SOAP 96
Software-Metrik 27
Software-Qualität 14
Speicherleck 184
Speicherlecktest 184
SQL-Injektion 115
Standardfunktionalitäten 131
State Transition Diagram 42
Statement Coverage 51
statische Code-Analyse 171
statische Prüfung 165
Steuerbarkeit 16

Stichprobentest 145
Stresstest 182
Strukturtest 28
Suchmaschinen 161
Suchmaschinenoptimierung 161
Systemprotokolle 101
Systemtest
- funktionaler 102, 232
- nicht-funktionaler 232

T

Test 24
- Deinstallation 175
- dynamisch 25
- Erstinstallation 174
- funktional 25
- kontrollflussbasiert 56
- nicht funktional 25
- statisch 25
- Update-Installation 175

Test Driven Development 89
Testabdeckungsgrad 51, 94
- C0, Anweisungsabdeckung 51
- C1, Zweigabdeckung 51
- C2, einfache Bedingungsabdeckung 52
- C3, mehrfache Bedingungsabdeckung 52

Testart 25
Testautomatisierung 177
Testdatengenerator 225
Testeinheit 24
Testen 24
Testendekriterium **27**, 58
Tester 226
- Auftraggeber als 227
- ausgebildeter 226
- Experte 226
- neutraler 227

Testexperte 226
Testfall 26
- konkreter 42
- logischer 42

Testfallabdeckung 49
Testfallabdeckungsgrad 49, 94
Testfallermittlung
- anforderungsbasiert 32
- anwendungsfallbasiert 46
- zustandsbasiert 43

Testframework 91
Testlauf 27
Testmanagement 233
Testmandant 212
Testmethode 26
Testmetrik 27
Testmittel 222
Testobjekt 24
Testphase 26
Testplanung 207, 219, 229
Testroboter 213
Testspezifikation 27
Testspezifikationswerkzeuge 48
Teststufe 26, 229
- Abnahmetest 233
- Betrieb 235
- Entwicklertest 230
- Integrationstest 231
- Komponententest 230
- Systemtest 232

Testsuite 226
Testszenario 26
Testteam 226
Testtreiber 94
Testtyp **25**
- Bewertung 219

Testüberdeckungsgrad 51
Testverfahren 26
Testwerkzeuge 5
- allgemeine 225
- Bereitstellung 223
- prozessunterstützende 225
- spezielle 223

Testwiederholungen 207
Testzyklen 237
Thread 186, **266**
Transaktion 188, **266**
Transaktionsdauer 188

U

Übertragbarkeit 15, 172
UML 47
Umlaufzeit 188
Unified Modeling Language 47
Unit-Test 92
Unverfügbarkeit 201

Ursache-Wirkungs-Analyse 38
Ursache-Wirkungs-Graph 38
Usability **17**, 18, 148
Usability-Test **148**, 177, 222
Use Case 46
Use Case Diagram 46
User Tracking Tool 112, 146, 152

V

Validatoren - W3C 172
Verbindungen 189
Verfügbarkeitstest 202
visuelle Überprüfung 166, 167
Volumentest 183

W

Wartbarkeit 202
Wartung 211
Wartungstest 211
Web Accessibility Initiative 156
Web-Adresse 160
Web-Anwendung 1
Web-Applikation 1
Web-Archive 267
Web-Auftritt 25
Web-Impressum 130
Webpage 25
Webpräsenz 25
Web-Recht 127
Web-Seite 25
Website **25**, 124
Whitebox-Test **28**, 93
Whitebox-Verfahren 28
Wiederherstellbarkeit 198
Wiederherstellbarkeitstest 198
Wiederverwendbarkeit 173
World Wide Web Consortium 155

X

XHTML 267
XML 267
X-Windows 267

Z

Zeitverhalten 180
Zufriedenstellung 17
Zugänglichkeitsrichtlinien 156
Zugänglichkeitstest **156**, 222
Zustandsdiagramm 43
Zustandsübergangsdiagramm 42
Zuverlässigkeit **14**, 201