Continuous Delivery

Der pragmatische Einstieg

Bearbeitet von
Eberhard Wolff

2., aktualisierte und erweiterte Auflage 2016. Taschenbuch. XL, 272 S. Softcover
ISBN 978 3 86490 371 7
Format (B x L): 16,5 x 24 cm

Inhaltsverzeichnis

1 Einleitung 1

1.1 Motivation 1

1.2 Abgrenzung des Buchs zu ISTQB-Lehrplänen 1

1.3 Zur Gliederung dieses Buchs 2

1.4 Die wichtigsten Begriffe kurz erklärt 3
1.4.1 Definition von Fachbegriffen 3
1.4.2 Zu Definitionen und TesterInnen 5

1.5 Ein Überblick über das Umfeld des Software-Testing 5
1.5.1 Ursachen von Software-Fehlern 6
1.5.2 Warum Programmfehler nicht entdeckt werden 7
1.5.3 Angebrachter Testaufwand 8
1.5.4 Der Tester und der Testprozess 9
1.5.5 Modellieren der Software-Umgebung 10
1.5.6 Erstellen von Testfällen 12
1.5.7 Ausführen und Evaluieren der Tests 14
1.5.8 Messen des Testfortschritts 15
1.5.9 Testdesign und Testdokumentation im Software-Entwicklungsprozess 15
1.5.10 Verschiedene Teststufen und deren Zusammenspiel 16
1.5.11 Andere Verifikationsmethoden als Ergänzung zum Test 19
1.5.12 Agile Prozessmodelle 21
1.5.13 Der Software-Test in agilen Vorgehensmodellen 22
1.5.14 Wer testet die Tester? 24

2 Anforderungen und Test 25

2.1 Die Bedeutung textueller Anforderungen 25

2.2 Requirements Engineering im Projekt 26

2.3 Arten und Quellen von Anforderungen 27

2.4 Warum Anforderungen dokumentiert werden sollen 28

2.5 Die Review von Anforderungen 29
2.5.1 Testbarkeit von Anforderungen 30
2.5.2 Modifizierbarkeit und Erweiterbarkeit 31
2.5.3 Relevanz von Anforderungen 32

2.6 Der Umgang mit natürlicher Sprache 32
2.6.1 Einfache Sprache gegen Missverständnisse 32
2.6.2 Gelenkte Sprache 34

2.7 Hinweise zur Dokumentenform 35

2.8 Die Spezifikation an der Schnittstelle zum Testteam 38
2.8.1 Konfiguration von Testdesigns 38
2.8.2 Vollständigkeit von Spezifikationen 39

2.9 Werkzeuge zur Review von Anforderungen 40

2.10 Diskussion 41
2.10.1 Verifikation beim Requirements Engineering mit Augenmaß 41
2.10.2 Bewertung der Rolle des Requirements Engineering für den Testprozess 42

2.11 Fragen und Übungsaufgaben 43

3 Review des Designs 45

3.1 Ziele der Review des Architekturdesigns 45

3.2 Ziele der Review des Detaildesigns 46

3.3 Eigenschaften von gutem Software-Design 47

3.4 Hinweise zur Architektur-Design-Review 47

3.5 Embedded Design 51
3.5.1 Sicherheit, Verfügbarkeit & Co 51
3.5.2 Wartbarkeit des Geräts 51
3.5.3 Ressourcenverbrauch 52
3.5.4 Design von Echtzeitsystemen 52

3.6 Diskussion 52

3.7 Fragen und Übungsaufgaben 53

4 Automatische statische Code-Analyse 55

4.1 Motivation zum Einsatz von Analysewerkzeugen 55

4.2 Techniken von Analysewerkzeugen im unteren Preissegment 56
4.2.1 Sprachspezifische Fallstricke 58
4.2.2 Kontrollflussanalyse 59
4.2.3 Datenflussanalyse, Initialisation Tracking 60
4.2.4 Datenflussanalyse, Value Tracking 61
4.2.5 Semantische Analyse 62
4.2.6 Starke Typenprüfung 64

4.3 Techniken von Analysewerkzeugen im oberen Preissegment 64
4.3.1 Größerer Komfort für den Benutzer 65
4.3.2 Concurrency Checks 66
4.3.3 Stack-Analyse und erweiterte Kontrollflussanalyse 66
4.3.4 Erschöpfende Analyse des Zustandsbaums 67

4.4 Statische Security-Analyse (SSA) 67

4.5 Code-Metriken 69

4.6 Werkzeuge für die Automatische Code-Analyse 71

4.7 Diskussion 73

4.8 Fragen und Übungsaufgaben 74

5 Code-Reviews 75

5.1 Review-Arten 75
5.1.1 Code-Inspektionen 75
5.1.2 Walkthrough 76
5.1.3 Peer-Review 77

5.2 Pair Programming 78

5.3 Werkzeuge zur Code-Review 78

5.4 Diskussion 81

5.5 Fragen und Übungsaufgaben 84

6 Unit-Tests 85

6.1 Der Unit-Test im Entwicklungsprozess 85

6.2 Zur Definition von Unit-Test und Modultest 86

6.3 Black-Box-Testfälle beim White-Box-Test 86
6.3.1 Äquivalenzklassenbildung 87
6.3.2 Grenzwertanalyse 88
6.3.3 Andere Methoden 89

6.4 Stubs und Treiber 90

6.5 Verschiedene Typen von Werkzeugen beim White-Box-Test 98
6.5.1 Unit-Test-Frameworks 98
6.5.2 Werkzeuge zur Testerstellung 99
6.5.3 Werkzeuge zur Messung der Testabdeckung 101

6.6 Testabdeckung 102
6.6.1 Statement Coverage 102
6.6.2 Branch Coverage und Decision Coverage 103
6.6.3 Decision/Condition Coverage 104
6.6.4 Modified Condition/Decision Coverage 104
6.6.5 Andere Testabdeckungen 105
6.6.6 Testabdeckung bei modellbasierter Entwicklung 105
6.6.7 Messung der Testabdeckung 105

6.7 Basis Path Testing 107

6.8 Host oder Target Testing? 109

6.9 Den Code immer unverändert testen? 110

6.10 Unit-Tests bei objektorientierten Sprachen 111

6.11 Grenzen des Unit-Tests 112

6.12 Werkzeuge für den Unit-Test 113
6.12.1 Unit-Test-Frameworks 113
6.12.2 Werkzeuge zur Testerstellung 113
6.12.3 Coverage-Analyse 116

6.13 Diskussion 116
6.13.1 Testabdeckung 116
6.13.2 Organisation von Unit-Tests 118

6.14 Fragen und Übungsaufgaben 119

7 Integrationstests 123

7.1 Software/Software-Integrationstest 123
7.1.1 Bottom-up-Unit-Tests als Integrationstest 123
7.1.2 Strukturierter Integrationstest 126
7.1.3 Testabdeckung der Aufrufe von Unterprogrammen 128
7.1.4 Vergleich der Teststrategien 130
7.1.5 Grenzen des Software/Software-Integrationstests 132
7.1.6 Diskussion des Software/Software-Integrationstests 133

7.2 Ressourcentests 135
7.2.1 Statischer Ressourcentest 135
7.2.2 Dynamischer Ressourcentest 135

7.3 Hardware/Software-Integrationstest 138
7.3.1 Bottom-up-Verfahren 139
7.3.2 Regressionsverfahren 139
7.3.3 Black-Box-Verfahren 139
7.3.4 Test und Analysen bei Sicherheitsrelevanz 140
7.3.5 Diskussion des Hardware/Software-Integrationstests 140

7.4 Systemintegrationstest 141

7.5 Werkzeuge für den Integrationstest 142

7.6 Fragen und Übungsaufgaben 142

8 Systemtests 145

8.1 Funktionale Systemtests 145
8.1.1 Zuordnung funktionaler Systemtests zu Anforderungen 145
8.1.2 Äquivalenzklassen und Grenzwerte im Black-Box-Test 146
8.1.3 Zustandsbasierter Test 149
8.1.4 Ursache-Wirkungs-Analyse 153
8.1.5 CECIL-Methode 162
8.1.6 Entscheidungstabellentechnik 163
8.1.7 Paarweises Testen und Klassifikationsbaum-Methode 163
8.1.8 Back To Back Testing 165
8.1.9 Erfahrungsbasierter Test 165
8.1.10 Diskussion des Black-Box-Tests 167
8.1.11 Auswahl eines Black-Box-Testverfahrens für eine Aufgabe 167
8.1.12 Werkzeuge für Funktionstests 168

8.2 Test der Benutzerschnittstelle 169
8.2.1 Grafische Benutzerschnittstelle 169
8.2.2 Werkzeuge für GUI-Tests 169
8.2.3 Eingebettete Benutzerschnittstellen 171
8.2.4 Werkzeuge für den Test von eingebetteten Benutzerschnittstellen 172

8.3 Performanztest und Lasttest 173
8.4 Stresstest 174
8.5 Volumentest 175
8.6 Failover und Recovery Testing 175
8.7 Ressourcentests 177
8.8 Installationstests 178
8.9 Konfigurationstests 179
8.10 Security-Tests 180
8.11 Dokumententests 182
8.12 Testumgebung und Testdaten 183
8.13 Formale Methoden 183
8.13.1 Symbolischer Test 184
8.13.2 Deduktive Verifikation von funktionalen Anforderungen 184
8.13.3 Model Checking 185
8.14 Automation von Systemtests 187
8.14.1 Vor- und Nachteile der Testautomation 188
8.14.2 Tipps zur Automation von Systemtests 189
8.15 Dokumentation des Testdesigns und der Testergebnisse 194
8.16 Grenzen des Systemtests 195
8.17 Fragen und Übungsaufgaben 195

9 Testen von RTOS und Middleware 199

9.1 Definition und Motivation 199
9.2 White-Box-Requirements-Test 200
9.3 Test eines Interrupt-Managers 201
9.4 Test eines Schedulers 202
9.5 Fragen und Übungsaufgaben 204

10 Race Conditions 205

10.1 Definition von Data Races 205
10.2 Dynamische Data-Race-Analyse 210
10.2.1 Eraser 210
10.2.2 Lamports Happens-Before-Relation 213

10.3 Statische Data-Race-Analyse 216
10.3.1 Ansätze zur statischen Data-Race-Analyse 216
10.3.2 Vergleich zur dynamischen Data-Race-Analyse 218
10.4 Werkzeuge für die Data-Race-Analyse 218
10.5 Diskussion 219
10.6 Fragen und Übungsaufgaben 221

11 Deadlocks 223

11.1 Über die Entstehung von Deadlocks 223
11.2 Verschiedene Arten der Deadlock-Analyse 224
11.3 Dynamische Deadlock-Analyse 225
11.4 Statische Deadlock-Analyse 225
11.5 Werkzeuge zur Deadlock-Detektion 226
11.6 Diskussion 227
11.7 Fragen und Übungsaufgaben 227

12 Echtzeit-Verifikation 229

12.1 Antwortzeiten bei funktionalen Tests 229
12.2 WCET-Analyse 230
12.2.1 Problemstellung 230
12.2.2 Laufzeitanalyse 232
12.3 Werkzeuge für die WCET-Analyse 235
12.4 Diskussion 236
12.5 Fragen und Übungsaufgaben 237

13 Schedulability-Analyse 239

13.1 Aufgaben der Schedulability-Analyse 239
13.2 Definitionen 240
13.3 Diskussion der Scheduling-Strategien 241
13.3.1 Statisches Scheduling 242
13.3.2 Dynamisches Scheduling 243

13.4 Analyse bei Fixed-Priority-Single-CPU-Systemen 245
13.4.1 Optimale Prioritätsvergabe 245
13.4.2 Rate Monotonic Analysis 246
13.4.3 Exakte Antwortzeitenanalyse 247
13.4.4 Gegenseitiger Ausschluss 252
13.4.5 Aperiodische Aufgaben 254
13.4.6 Kontextwechsel 255
13.4.7 Cache und Out Of Order Execution 255
13.4.8 Input-Jitter 255
13.4.9 Interrupts 256
13.5 Multi-CPU-Systeme 256
13.5.1 Multicore- und Multiprozessor-Systeme 257
13.5.2 Verteilte Systeme 257
13.6 Scheduling-Analyse für CAN 259
13.7 Werkzeuge 261
13.8 Diskussion 262
13.9 Fragen und Übungsaufgaben 263

14 Hardware/Software-Interaktionsanalyse 265

14.1 Die FMEA als Grundlage der HSIA 265
14.2 Die HSIA als Quelle für Software-Anforderungen 269
14.3 Software-Kritikalitätsanalyse 270
14.4 Software-FMEA 271
14.5 Werkzeuge 272
14.6 Diskussion 273
14.7 Fragen und Übungsaufgaben 273

15 Modellbasierter Test 275

15.1 Begriffsdefinition 275
15.2 MBT und Testautomation 276
15.3 Modelle 276
15.3.1 Statecharts 276
15.3.2 SDL 276
15.3.3 Message Sequence Charts 277
15.3.4 UML Version 2 277
15.3.5 SysML 278
15.3.6 Funktionsmodellierung 278

15.4 Testmodell vs. Implementierungsmodell 278
15.5 Werkzeuge 279
15.6 Diskussion 280
15.7 Fragen und Übungsaufgaben 280

16 Trace-Daten im Testumfeld 281

16.1 Das Dilemma mit instrumentiertem Code 281
16.2 Embedded-Trace-Schnittstellen 282
16.3 Werkzeuge 283
16.4 Diskussion 283

17 Testmanagement 287

17.1 Testplanung 287
17.2 Teststeuerung 289
17.3 Abweichungsmanagement 291
17.4 Bewertung und Anpassung des Testprozesses 293
17.4.1 Formale Reifegradmodelle für den Software-Test 293
17.4.2 Prozessbewertung in agilen Projekten 294
17.4.3 Mit Augenmaß ins Kostenoptimum 294
17.5 Risikobasierter Test 297
17.6 Werkzeuge 300
17.7 Diskussion 300
17.8 Fragen und Übungsaufgaben 302

18 Qualitätsmanagement 303

18.1 Definition 303
18.2 Qualitätsmanagement-Standards 304
18.3 Kosten und Haftungsrelevanz des QM 307
18.4 Umsetzung des Qualitätsmanagements 308
18.5 Die Rolle des Qualitätsmanagers 309
18.6 Mit Metriken die Qualität steuern 310
18.7 Die Wirtschaftlichkeit von QM 313

18.8 Werkzeuge 314
18.9 Diskussion 314
18.10 Fragen und Übungsaufgaben 315

19 Software-Test und Haftungsrisiko 317

19.1 Ein Software-Fehler im Sinne des Gesetzes 317
19.2 Vertragliche Gewährleistung und Haftung 318
19.3 Vertragliche Beschränkung der Haftung 319
19.4 Produzentenhaftung bei Software 320
19.5 Produkthaftung 320
19.6 Sorgfaltspflicht des Software-Herstellers 321
19.7 Technische Normen mit Bezug zum Software-Test 324
19.7.1 DIN IEC 56/575/CD 324
19.7.2 IEEE Std 1012 324
19.7.3 IEEE Std 829 325
19.7.4 IEEE Std 1008-1987 325
19.7.5 ISO/IEC 29119 326
19.7.6 IEC/EN 61508 330
19.7.7 ISO 26262 333
19.7.8 Normenreihe 250XX 333
19.8 Tipps vom Rechtsanwalt und vom Techniker 336
19.9 Fragen und Übungsaufgaben 338

Nachwort 339

Anhang

Anhang A – Lösungen zu den Übungsaufgaben 343

Anhang B – Dokumentation des Testdesigns 369

Anhang C – Software-Verifikationsplan 371

Anhang D –Software-Verifikationsreport 375

Quellenverzeichnis 377

Index 387