# Inhaltsverzeichnis

**Geleitwort** ..... **11**

**Einleitung** ..... **13**

**1 Einführung** ..... **25**

1.1 Setup ..... 25
1.2 Die Beispiele ..... 25
1.3 C++-Compiler ..... 27
1.4 CMake ..... 30
1.5 Google Mock ..... 30
1.6 CppUTest ..... 33
1.7 libcurl ..... 34
1.8 JsonCpp ..... 34
1.9 rlog ..... 35
1.10 Boost ..... 36
1.11 Beispiele erstellen und Tests ausführen ..... 37
1.12 Teardown ..... 38

**2 Testgetriebene Entwicklung: Ein erstes Beispiel** ..... **39**

2.1 Setup ..... 39
2.2 Der Soundex-Algorithmus ..... 39
2.3 Erste Schritte ..... 40
2.4 Unsauberen Code korrigieren ..... 48
2.5 Schrittweises Vorgehen ..... 50
2.6 Fixtures ..... 53
2.7 Denkprozesse bei TDD ..... 56
2.8 Testen und testgetriebene Entwicklung im Vergleich ..... 59

2.9 Was wäre, wenn ...? . . . . . 62
2.10 Eins nach dem anderen! . . . . . 63
2.11 Die Länge einschränken . . . . . 65
2.12 Vokale fallen lassen . . . . . 67
2.13 Tests übersichtlich gestalten . . . . . 67
2.14 Querdenken beim Testen . . . . . 69
2.15 Zurück zum Thema . . . . . 72
2.16 Refactoring zu Funktionen mit nur je einer Aufgabe . . . . . 73
2.17 Der letzte Schliff . . . . . 75
2.18 Welche Tests fehlen noch? . . . . . 76
2.19 Unsere Lösung . . . . . 77
2.20 Die Soundex-Klasse . . . . . 78
2.21 Teardown . . . . . 81

**3 Testgetriebene Entwicklung: Grundlagen . . . . . 83**

3.1 Setup . . . . . 83
3.2 Unit Tests und Grundlagen von TDD . . . . . 83
3.3 Der TDD-Zyklus: Rot – Grün – Refactoring . . . . . 86
3.4 Die drei Regeln von TDD . . . . . 88
3.5 Verfrühtes Bestehen von Tests . . . . . 89
3.6 Die richtige Einstellung für den erfolgreichen Einsatz von TDD . . . . . 98
3.7 Techniken für den Erfolg . . . . . 102
3.8 Teardown . . . . . 107

**4 Tests konstruieren . . . . . 109**

4.1 Setup . . . . . 109
4.2 Aufbau . . . . . 109
4.3 Schnelle Tests, langsame Tests, Filter und Suiten . . . . . 116
4.4 Assertions (Zusicherungen) . . . . . 120
4.5 Private Daten untersuchen . . . . . 127
4.6 Testen und testgetriebene Entwicklung im Vergleich: Parametrisierte Tests und andere Spielereien . . . . . 132
4.7 Teardown . . . . . 136

**5 Testdoubles ... 137**

5.1 Setup ... 137
5.2 Herausforderungen durch Abhängigkeiten ... 137
5.3 Testdoubles ... 138
5.4 Ein selbst gebautes Testdouble ... 139
5.5 Die Testabstraktion bei der Verwendung von Testdoubles verbessern ... 144
5.6 Mock-Frameworks verwenden ... 146
5.7 Testdoubles platzieren ... 157
5.8 Ein anderes Vorgehen beim Design ... 164
5.9 Strategien zur Verwendung von Testdoubles ... 166
5.10 Verschiedenes zum Thema Testdoubles ... 171
5.11 Teardown ... 174

**6 Inkrementelles Design ... 175**

6.1 Setup ... 175
6.2 Einfaches Design ... 175
6.3 Was ist mit dem Design im Voraus? ... 201
6.4 Hindernisse für das Refactoring ... 205
6.5 Teardown ... 208

**7 Qualitativ hochwertige Tests ... 209**

7.1 Setup ... 209
7.2 Tests nach dem FIRST-Prinzip ... 209
7.3 Eine Zusicherung pro Test ... 215
7.4 Testabstraktion ... 218
7.5 Teardown ... 232

**8 Herausforderungen durch Legacy-Code ... 233**

8.1 Setup ... 233
8.2 Legacy-Code ... 234
8.3 Kernsätze ... 234
8.4 Die Altanwendung ... 236
8.5 Die Denkweise der testgetriebenen Entwicklung ... 239
8.6 Sicheres Refactoring zur Unterstützung von Tests ... 240

8.7 Tests zur Beschreibung des vorhandenen Verhaltens hinzufügen . . . . . 244
8.8 Ablenkungen durch die Realitäten des vorhandenen Codes . . 246
8.9 Ein Testdouble für rlog erstellen . . . . . 246
8.10 Testgetriebene Änderungen . . . . . 251
8.11 Eine neue Story . . . . . 253
8.12 Schnellere Tests finden . . . . . 255
8.13 Mondo Extracto . . . . . 256
8.14 Verwendung einer Membervariablen . . . . . 259
8.15 Verwendung eines Mocks . . . . . 260
8.16 Alternative Injektionstechniken . . . . . 265
8.17 Umfassende Änderungen mit der Mikado-Methode . . . . . 265
8.18 Die Mikado-Methode im Überblick . . . . . 266
8.19 Methoden mit Mikado verschieben . . . . . 267
8.20 Weitere Überlegungen zur Mikado-Methode . . . . . 279
8.21 Lohnt sich der Aufwand? . . . . . 280
8.22 Teardown . . . . . 281

**9 TDD für Threads . . . . . 283**

9.1 Setup . . . . . 283
9.2 Grundprinzipien für die testgetriebene Thread-Entwicklung . . . . . 283
9.3 GeoServer . . . . . 284
9.4 Performance-Anforderungen . . . . . 291
9.5 Eine asynchrone Lösung gestalten . . . . . 293
9.6 Dies ist immer noch einfaches TDD . . . . . 297
9.7 »Einfädeln« . . . . . 299
9.8 Probleme der Parallelverarbeitung offenlegen . . . . . 301
9.9 Client-Threads im Test erstellen . . . . . 304
9.10 Mehrere Threads in ThreadPool erstellen . . . . . 306
9.11 Zurück zu GeoServer . . . . . 308
9.12 Teardown . . . . . 313

**10 Weitere Aspekte von TDD ..... 315**

10.1 Setup ..... 315
10.2 TDD und die Performance ..... 315
10.3 Unit Tests, Integrationstests und Akzeptanztests ..... 325
10.4 Transformation Priority Premise (TPP) ..... 328
10.5 Zusicherungen zuerst schreiben ..... 343
10.6 Teardown ..... 347

**11 Wachstum und Pflege von TDD ..... 349**

11.1 Setup ..... 349
11.2 Laien TDD erklären ..... 350
11.3 Die Todesspirale schlechter Tests (»SCUMmy-Kreislauf«) ..... 355
11.4 Pair Programming ..... 358
11.5 Katas und Dojos ..... 361
11.6 Metriken zur Codeabdeckung wirkungsvoll anwenden ..... 365
11.7 Kontinuierliche Integration ..... 366
11.8 Teamstandards für TDD aufstellen ..... 367
11.9 Mit der Community auf dem Laufenden bleiben ..... 368
11.10 Teardown ..... 369

**A Unit-Test-Frameworks im Vergleich ..... 371**

A.1 Setup ..... 371
A.2 Funktionen von TDD-Unit-Test-Frameworks ..... 371
A.3 Hinweise zu Google Mock ..... 373
A.4 Hinweise zu CppUTest ..... 373
A.5 Andere Frameworks für Unit Tests ..... 374
A.6 Teardown ..... 374

**B Code-Kata: Umrechner für römische Zahlen ..... 375**

B.1 Setup ..... 375
B.2 Los geht's! ..... 375
B.3 Übung macht den Meister ..... 384
B.4 Teardown ..... 384

**Literatur ..... 385**

**Index ..... 387**