

Inhaltsverzeichnis

1	Einleitung	1
1.1	Was ist Testgetriebene Entwicklung?	2
1.2	Warum Testgetriebene Entwicklung?	4
1.3	Über dieses Buch	7
1.4	Merci beaucoup	8
2	Testgetriebene Entwicklung, über die Schulter geschaut	9
2.1	Eine Programmierepisode	10
2.2	Testgetriebenes Programmieren	11
2.3	Möglichst einfaches Design	12
2.4	Ein wenig Testen, ein wenig Programmieren	13
2.5	Evolutionäres Design	14
2.6	Natürlicher Abschluss einer Programmierepisode	16
2.7	Refactoring	17
2.8	Abschließende Reflexion	19
2.9	Häufige Integration	20
2.10	Rückblende	20
3	Unit Tests mit JUnit	21
3.1	Download und Installation	21
3.2	Ein erstes Beispiel	22
3.3	Anatomie eines Testfalls	23
3.4	Test-First	24
3.5	JUnit in Eclipse	26
3.6	Das JUnit-Framework von innen	27
3.7	»Assert«	27
3.8	»AssertionFailedError«	29

3.9	»TestCase«	31
3.10	Lebenszyklus eines Testfalls	34
3.11	»TestSuite«	36
3.12	»TestRunner«	38
3.13	Zwei Methoden, die das Testen vereinfachen	39
3.14	Testen von Exceptions	42
3.15	Unerwartete Exceptions	43
3.16	JUnit 4	44
4	Testgetriebene Programmierung	51
4.1	Die erste Direktive	51
4.2	Der Testgetriebene Entwicklungszyklus	52
4.3	Die Programmierzüge	53
4.4	Beginn einer Testepisode	54
4.5	Ein einfacher Testplan	55
4.6	Erst ein neuer Test ...	56
4.7	... dann den Test fehlschlagen sehen	57
4.8	... schließlich den Test erfüllen	59
4.9	Zusammenspiel von Test- und Programmcode	60
4.10	Ausnahmebehandlung	61
4.11	Ein unerwarteter Erfolg	62
4.12	Ein unerwarteter Fehlschlag	64
4.13	Vorprogrammierte Schwierigkeiten	68
4.14	Kleine Schritte gehen	72
5	Refactoring	73
5.1	Die zweite Direktive	73
5.2	Die Refactoringzüge	74
5.3	Von übel riechendem Code ...	75
5.4	... über den Refactoringkatalog	76
5.5	... zur Einfachen Form	77
5.6	Überlegungen zur Refactoringroute	78
5.7	Substitution einer Implementierung	79
5.8	Evolution einer Schnittstelle	80
5.9	Teilen von Klassen	83
5.10	Verschieben von Tests	85
5.11	Abstraktion statt Duplikation	86
5.12	Die letzte Durchsicht	90
5.13	Ist Design tot?	91

5.14	Richtungswechsel ...	95
5.15	... und der wegweisende Test	96
5.16	Fake it ('til you make it)	98
5.17	Vom Bekannten zum Unbekannten	99
5.18	Retrospektive	106
5.19	Tour de Design évolutionnaire	107
5.20	Durchbrüche erleben	108
6	Häufige Integration	109
6.1	Die dritte Direktive	109
6.2	Die Integrationszüge	110
6.3	Änderungen mehrmals täglich zusammenführen ...	111
6.4	... das System von Grund auf neu bauen	113
6.5	... und ausliefern	114
6.6	Versionsverwaltung (mit CVS oder Subversion)	116
6.7	Build-Skript mit Ant	116
6.8	Build-Prozess-Tuning	121
6.9	Integrationsserver mit CruiseControl	122
6.10	Aufbau einer Staging-Umgebung	124
6.11	Teamübergreifende Integration	124
6.12	Gesund bleiben	125
7	Testfälle schreiben von A bis Z	127
7.1	Aufbau von Testfällen	127
7.2	Benennung von Testfällen	129
7.3	Buchführung auf dem Notizblock	129
7.4	Der erste Testfall	130
7.5	Der nächste Testfall	130
7.6	Erinnerungstests	131
7.7	Ergebnisse im Test festschreiben, nicht berechnen	131
7.8	Erst die Zusicherung schreiben	132
7.9	Features testen, nicht Methoden	133
7.10	Finden von Testfällen	134
7.11	Generierung von Testdaten	134
7.12	Implementierungsunabhängige Tests	136
7.13	Kostspielige Setups	137
7.14	Lange Assert-Ketten oder mehrere Testfälle?	138
7.15	Lerntests	139

7.16	Minimale Fixture!	140
7.17	Negativtests	142
7.18	Organisation von Testfällen	143
7.19	Orthogonale Testfälle	144
7.20	Parameterisierbare Testfälle	150
7.21	Qualität der Testsuite	153
7.22	Refactoring von Testcode	160
7.23	Reihenfolgeunabhängigkeit der Tests	162
7.24	Selbsterklärende Testfälle	164
7.25	String-Parameter von Zusicherungen	165
7.26	Szenarietests	166
7.27	Testexemplare	166
7.28	Testsprachen	168
7.29	Umgang mit Defekten	169
7.30	Umgang mit externem Code	172
7.31	Was wird getestet? Was nicht?	173
7.32	Zufälle und Zeitabhängigkeiten	173
8	Isoliertes Testen	
	durch Stubs und Mocks	175
8.1	Verflichte Abhängigkeiten!	176
8.2	Was ist die Unit im Unit Test?	177
8.3	Mikrointegrationstest versus strikter Unit Test	178
8.4	Vertrauenswürdige Komponenten	179
8.5	Austauschbarkeit von Objekten	182
8.6	Stub-Objekte	183
8.7	Größere Unabhängigkeit	184
8.8	Testen durch Indirektion	185
8.9	Stub-Variationen	189
8.10	Testen von Mittelsmännern	190
8.11	Self-Shunt	191
8.12	Testen von innen	193
8.13	Möglichst frühzeitiger Fehlschlag	194
8.14	Erwartungen entwickeln	195
8.15	Gebrauchsfertige Erwartungsklassen	197
8.16	Testen von Protokollen	198
8.17	Mock-Objekte	201
8.18	Wann verwende ich welches Testmuster?	204
8.19	Crashtest-Dummies	207

8.20	Dynamische Mocks mit EasyMock	208
8.21	Stubs via Record/Replay	210
8.22	Überspezifizierte Tests	211
8.23	Überstrapazierte Mocks	211
8.24	Systemgrenzen im Test	212
9	Entwicklung mit Mock-Objekten	215
9.1	Tell, don't ask	215
9.2	Von außen nach innen	216
9.3	Wer verifiziert wen?	217
9.4	Schnittstellen finden auf natürlichem Weg	218
9.5	Komponierte Methoden	220
9.6	Vom Mock lernen für die Implementierung	221
9.7	Viele schmale Schnittstellen	223
9.8	Kleine fokussierte Klassen	224
9.9	Tell und Ask unterscheiden	225
9.10	neremargorP sträwkcüR	226
9.11	Schüchterner Code und das Gesetz von Demeter	228
9.12	Fassaden und Mediatoren als Abstraktionsebene	229
9.13	Rekonstruktion	230
10	Akzeptanztests mit FIT	233
10.1	Von einer ausführbaren Spezifikation	234
10.2	Download Now	234
10.3	Schritt für Schritt für Schritt	235
10.4	... zum ausführbaren Anforderungsdokument	242
10.5	Die drei Basis-Fixtures	243
10.6	»ActionFixture«	243
10.7	Richtung peilen, Fortschritt erzielen	247
10.8	Fixture wachsen lassen, dann Struktur extrahieren	249
10.9	Nichts als Fassade	251
10.10	Die Fixture als zusätzlicher Klient	253
10.11	Aktion: Neue Aktion	254
10.12	»ColumnFixture«	256
10.13	Fixture-Interkommunikation	258
10.14	Negativbeispiele	259
10.15	Transformation: Action \Rightarrow Column	261
10.16	»RowFixture«	264
10.17	Einfacher Schlüssel	266

10.18	Mehrfacher Schlüssel	269
10.19	Abfragemethoden einspannen	270
10.20	»Summary«	271
10.21	»ExampleTests«	274
10.22	»AllFiles«	276
10.23	Setup- und Teardown-Fixtures	278
10.24	Das FIT-Framework von innen	278
10.25	»FileRunner«	279
10.26	»Parse«	279
10.27	»Fixture«	282
10.28	Annotationsmöglichkeiten in Dokumenten	284
10.29	»TypeAdapter«	289
10.30	»ScientificDouble«	293
10.31	Domänenspezifische Grammatiken	293
10.32	»ArrayAdapter«	294
10.33	»PrimitiveFixture«	295
10.34	Domänenspezifische Fixtures	297
10.35	Anschluss finden	299
10.36	Stichproben reiner Geschäftslogik	300
10.37	Integrationstests gegen Fassaden und Services	300
10.38	Oberflächentests	302
10.39	Kundenfreundliche Namen	304
10.40	FitNesse	304
10.41	FitLibrary	306
10.42	Akzeptanztesten aus Projektsicht	307
11	Änderbare Software	309
11.1	Konstantes Entwicklungstempo	310
11.2	Kurze Zykluszeiten	315
11.3	Neue Geschäftsmodelle	316
	Literatur	321
	Werkzeugkasten	323
	Index	325