

Inhaltsverzeichnis

	Vorwort	13
	Vorwort des Übersetzers der deutschen Ausgabe	17
	Einleitung	19
	Über den Begriff »Craftsmanship«	19
	Auf dem einzig wahren Weg	19
	Einführung in das Buch	20
	Danksagungen	25
	Über den Autor	27
1	Craftsmanship	29
	Teil I Die Praktiken	37
2	Testgetriebene Entwicklung	45
2.1	Überblick	46
	2.1.1 Software	48
	2.1.2 Die drei Gesetze des TDD	49
	2.1.3 Das vierte Gesetz	57
2.2	Die Grundlagen	59
	2.2.1 Einfache Beispiele	59
	2.2.2 Stack	60
	2.2.3 Primfaktoren	75
	2.2.4 Bowling	83
2.3	Fazit	100
3	Fortgeschrittenes TDD	101
3.1	Sort 1	101
3.2	Sort 2	106
3.3	Steckenbleiben	114
3.4	Erstellen, Ausführen, Sicherstellen	121
	3.4.1 BDD einführen	122

3.4.2	Endliche Automaten.....	122
3.4.3	Und wieder BDD	124
3.5	Test-Doubles.....	124
3.5.1	Dummy.....	127
3.5.2	Stub	130
3.5.3	Spy	133
3.5.4	Mock	135
3.5.5	Fake	138
3.5.6	Das TDD-Unsicherheitsprinzip.....	140
3.5.7	London vs. Chicago.....	152
3.6	Architektur	155
3.7	Fazit	157
4	Testdesign.....	159
4.1	Datenbanken testen.....	160
4.2	Benutzeroberflächen testen.....	161
4.2.1	Eingaben in der GUI	163
4.3	Test Pattern	164
4.3.1	Testspezifische Unterklasse.....	165
4.3.2	Self-Shunt	166
4.3.3	Humble Object	166
4.4	Testdesign.....	169
4.4.1	Das Problem der fragilen Tests	169
4.4.2	Die Eins-zu-eins-Kopplung	170
4.4.3	Durchbrechen der Kopplung.....	171
4.4.4	Die Videothek	173
4.4.5	Speziell versus allgemein.....	190
4.5	Transformationsprioritätsgrundsatz	191
4.5.1	{ } → Nil.....	193
4.5.2	Nil → Konstante	193
4.5.3	Konstante → Variable.....	194
4.5.4	Ohne Bedingung → Verzweigung	195
4.5.5	Wert → Liste	195
4.5.6	Anweisung → Rekursion.....	196
4.5.7	Verzweigung → Iteration.....	196
4.5.8	Wert → Veränderter Wert	197
4.5.9	Beispiel: Fibonacci	197
4.5.10	Die Prämisse der Priorität der Transformation	201
4.6	Fazit	202

5	Refactoring	203
5.1	Was ist Refactoring?	204
5.2	Das Basis-Toolkit	205
5.2.1	Umbenennen	205
5.2.2	Methode extrahieren	206
5.2.3	Variable extrahieren	207
5.2.4	Feld extrahieren	209
5.2.5	Rubiks Würfel.	221
5.3	Die Praktiken	221
5.3.1	Tests	221
5.3.2	Schnelle Tests	222
5.3.3	Aufbrechen tieferreichender Eins-zu-eins-Kopplungen	222
5.3.4	Kontinuierliches Refactoring.	222
5.3.5	Gnadenloses Refactoring.	223
5.3.6	Lassen Sie die Tests bestehen!	223
5.3.7	Lassen Sie sich einen Ausweg offen.	224
5.4	Fazit	224
6	Einfaches Design	225
6.1	YAGNI	227
6.2	Abgedeckt durch Tests	229
6.2.1	Abdeckung	230
6.2.2	Ein asymptotisches Ziel.	231
6.2.3	Design?	232
6.2.4	Aber da ist mehr.	232
6.3	Aussagekraft maximieren	233
6.3.1	Die zugrunde liegende Abstraktion	234
6.3.2	Tests: Die zweite Hälfte des Problems	235
6.4	Duplikate minimieren.	236
6.5	Zufällige Duplizierung	237
6.6	Größe minimieren	238
6.7	Einfaches Design.	238
7	Kollaborative Programmierung	239
8	Akzeptanztests	243
8.1	Die Praktiken	245
8.2	Der kontinuierliche Build	246

Teil II Die Standards	247
<hr/>	
9 Produktivität	249
9.1 Wir werden nie Sch**** ausliefern	249
9.2 Leichte Anpassbarkeit	251
9.3 Wir werden immer bereit sein	252
9.4 Stabile Produktivität	254
10 Qualität	255
10.1 Kontinuierliche Verbesserung	255
10.2 Furchtlose Kompetenz	256
10.3 Extreme Qualität	257
10.4 Wir werfen die QS nicht über Bord	258
10.4.1 Die QS-Krankheit	259
10.5 Die QS wird nichts finden	259
10.6 Testautomatisierung	260
10.7 Automatisiertes Testen und Benutzeroberflächen	260
10.8 Testen der Benutzeroberfläche	261
11 Mut	263
11.1 Wir stehen füreinander ein	263
11.2 Ehrliche Schätzungen	264
11.3 Sie müssen NEIN sagen	266
11.4 Ständiges aggressives Lernen	267
11.5 Mentoring	268
Teil III Die Ethik	269
<hr/>	
12 Schaden	283
12.1 Erstens, keinen Schaden anrichten	284
12.1.1 Gesellschaftlicher Schaden	285
12.1.2 Funktionsbeeinträchtigung	286
12.1.3 Keine Schädigung der Struktur	288
12.1.4 Soft	290
12.1.5 Tests	291
12.2 Beste Arbeit	292
12.2.1 Es richtig machen	293
12.2.2 Was ist eine gute Struktur?	294

12.2.3	Eisenhower-Matrix	295
12.2.4	Programmierer sind Stakeholder	297
12.2.5	Ihr Bestes	298
12.3	Reproduzierbarer Beweis	300
12.3.1	Dijkstra	300
12.3.2	Beweis der Korrektheit.	301
12.3.3	Strukturierte Programmierung.	303
12.3.4	Funktionale Dekomposition	305
12.3.5	Testgetriebene Entwicklung	306
13	Integrität.	309
13.1	Kleine Zyklen.	309
13.1.1	Die Geschichte der Versionsverwaltung.	310
13.1.2	Git	314
13.1.3	Kurze Zyklen	315
13.1.4	Kontinuierliche Integration	316
13.1.5	Branches versus Toggles	317
13.1.6	Kontinuierliches Deployment	319
13.1.7	Kontinuierlicher Build.	320
13.2	Unerbittliche Verbesserung	321
13.2.1	Testabdeckung	321
13.2.2	Mutationstests	322
13.2.3	Semantische Stabilität	322
13.2.4	Aufräumen	323
13.2.5	Kreationen.	323
13.3	Hohe Produktivität beibehalten	324
13.3.1	Viskosität.	324
13.3.2	Umgang mit Ablenkungen	327
13.3.3	Zeitmanagement	329
14	Teamarbeit	331
14.1	Arbeit im Team	331
14.1.1	Offenes/virtuelles Büro	332
14.2	Ehrliche und faire Schätzungen.	333
14.2.1	Lügen.	334
14.2.2	Ehrlichkeit, Genauigkeit, Präzision	334
14.2.3	Genauigkeit.	339
14.2.4	Präzision	340

Inhaltsverzeichnis

14.2.5	Aggregation.....	342
14.2.6	Ehrlichkeit.....	343
14.3	Respekt.....	345
14.4	Niemals aufhören zu lernen.....	345
	Stichwortverzeichnis	347