Übersicht

Lob für den *Clean Coder* 17

Vorwort 19

Einführung 25

Danksagungen 29

Über den Autor 35

Auf dem Titelbild 37

Unverzichtbare Einführung 39

1 Professionalität 45

2 Nein sagen 61

3 Ja sagen 85

4 Programmieren 97

5 Test Driven Development 115

6 Praktizieren und Üben 123

7 Akzeptanztests 133

8 Teststrategien 151

9 Zeitmanagement 159

10 Schätzungen 171

11 Druck von außen 185

12 Teamwork 191

13 Teams und Projekte 201

14 Mentoring, Lehrzeiten und die Handwerkskunst 207

A Werkzeuge und Hilfsmittel 219

Index 235

Inhalt

Lob für den *Clean Coder* 17

Vorwort 19

Einführung 25
Über dieses Buch 27
Bibliografie 28

Danksagungen 29

Über den Autor 35

Auf dem Titelbild 37

Unverzichtbare Einführung 39

1 Professionalität 45
1.1 Seien Sie vorsichtig, wonach Sie verlangen 46
1.2 Verantwortung übernehmen 46
1.3 Erstens: Richte keinen Schaden an. 48
1.3.1 Beschädige nicht die Funktion 49
1.3.2 Beschädige nicht die Struktur 51
1.4 Arbeitsethik 53
1.4.1 Sie sollten sich in Ihrem Bereich auskennen 54
1.4.2 Lebenslanges Lernen 56
1.4.3 Praxis 56
1.4.4 Teamwork 57
1.4.5 Mentorenarbeit 57
1.4.6 Sie sollten sich in Ihrem Arbeitsgebiet auskennen 58
1.4.7 Identifizieren Sie sich mit Ihrem Arbeitgeber bzw. Kunden. 58
1.4.8 Bescheidenheit 58
1.5 Bibliografie 59

2 Nein sagen ... 61
2.1 Feindliche Rollen ... 64
2.1.1 Was ist mit dem Warum? ... 66
2.2 Hoher Einsatz ... 67
2.3 Ein »Teamplayer« sein ... 68
2.3.1 Versuchen ... 70
2.3.2 Passive Aggression ... 72
2.4 Die Kosten eines Ja ... 74
2.5 Code unmöglich ... 81

3 Ja sagen ... 85
3.1 Verbindliche Sprache ... 87
3.1.1 So erkennt man mangelnde Selbstverpflichtung ... 88
3.1.2 Wie echte Selbstverpflichtung klingt ... 89
3.1.3 Zusammenfassung ... 91
3.2 Lernen, wie man »Ja« sagt ... 91
3.2.1 Die Kehrseite von »Ich versuch's mal« ... 92
3.2.2 Der Disziplin verpflichtet ... 92
3.3 Schlussfolgerung ... 95

4 Programmieren ... 97
4.1 Bereit sein ... 98
4.1.1 Code um drei Uhr früh ... 99
4.1.2 Sorgencode ... 100
4.2 Der Flow-Zustand ... 101
4.2.1 Musik ... 102
4.2.2 Unterbrechungen ... 103
4.3 Schreibblockaden ... 104
4.3.1 Kreativer Input ... 105
4.4 Debugging ... 105
4.4.1 Zeit zum Debuggen ... 108
4.5 Die eigene Energie einteilen ... 108
4.5.1 Wann man den Stift weglegen muss ... 109
4.5.2 Die Heimfahrt ... 109
4.5.3 Die Dusche ... 109

4.6 In Verzug sein ... 110
4.6.1 Hoffnung ... 110
4.6.2 Sich beeilen ... 110
4.6.3 Überstunden ... 111
4.6.4 Unlautere Ablieferung ... 111
4.6.5 Definieren Sie »fertig und erledigt« ... 112
4.7 Hilfe ... 112
4.7.1 Anderen helfen ... 112
4.7.2 Hilfe annehmen ... 113
4.7.3 Mentorenarbeit ... 114
4.8 Bibliografie ... 114

5 Test Driven Development ... 115
5.1 Die Geschworenen haben sich entschieden ... 117
5.2 Die drei Gesetze des TDD ... 117
5.2.1 Die Litanei der Vorteile ... 118
5.2.2 Die professionelle Option ... 121
5.3 Was TDD nicht ist ... 121
5.4 Bibliografie ... 122

6 Praktizieren und Üben ... 123
6.1 Etwas Hintergrund übers Üben ... 123
6.1.1 22 Nullen ... 124
6.1.2 Durchlaufzeiten ... 125
6.2 Das Coding Dojo ... 127
6.2.1 Kata ... 127
6.2.2 Waza ... 129
6.2.3 Randori ... 129
6.3 Die eigene Erfahrung ausbauen ... 130
6.3.1 Open Source ... 130
6.3.2 Ethisch handeln ... 130
6.4 Schlussfolgerung ... 131
6.5 Bibliografie ... 131

7 Akzeptanztests ... 133
7.1 Anforderungen der Kommunikation ... 133
7.1.1 Verfrühte Präzisierung ... 135
7.2 Akzeptanztests ... 138
7.2.1 Die »Definition of Done« ... 138
7.2.2 Kommunikation ... 141
7.2.3 Automatisierung ... 141
7.2.4 Zusätzliche Arbeit ... 143
7.2.5 Wer schreibt die Akzeptanztests und wann? ... 143
7.2.6 Die Rolle des Entwicklers ... 144
7.2.7 Verhandlungen über die Tests und passive Aggression ... 145
7.2.8 Akzeptanz- und Unit-Tests ... 146
7.2.9 GUIs und andere Komplikationen ... 147
7.2.10 Andauernde Integration ... 148
7.3 Schlussfolgerung ... 149

8 Teststrategien ... 151
8.1 Die Qualitätssicherung sollte keine Fehler finden ... 152
8.1.1 Die Qualitätssicherung gehört zum Team ... 152
8.2 Die Pyramide der Testautomatisierung ... 153
8.2.1 Unit-Tests ... 153
8.2.2 Komponententests ... 154
8.2.3 Integrationstests ... 155
8.2.4 Systemtests ... 156
8.2.5 Manuelle explorative Tests ... 156
8.3 Schlussfolgerung ... 157
8.4 Bibliografie ... 157

9 Zeitmanagement ... 159
9.1 Meetings ... 160
9.1.1 Absagen ... 161
9.1.2 Sich ausklinken ... 161
9.1.3 Tagesordnung und Ziel ... 162
9.1.4 Stand-up-Meetings ... 162
9.1.5 Planungstreffen zur Iteration ... 163
9.1.6 Retrospektive und Demo der Iteration ... 163
9.1.7 Auseinandersetzungen und Meinungsverschiedenheiten ... 163

9.2 Fokus-Manna ... 164
9.2.1 Schlaf ... 165
9.2.2 Koffein ... 165
9.2.3 Die Akkus aufladen ... 166
9.2.4 Muskelfokus ... 166
9.2.5 Input vs. Output ... 167
9.3 Zeitfenster und Tomaten ... 167
9.4 Vermeidung ... 168
9.4.1 Umkehrung der Prioritäten ... 168
9.5 Sackgassen ... 168
9.6 Morast, Moore, Sümpfe und andere Schlamassel ... 169
9.7 Schlussfolgerung ... 170

10 Schätzungen ... 171
10.1 Was eine Schätzung ist ... 173
10.1.1 Ein Commitment ... 173
10.1.2 Eine Schätzung ... 174
10.1.3 Implizierte Commitments ... 176
10.2 PERT ... 177
10.3 Aufgaben schätzen ... 179
10.3.1 Wideband Delphi ... 179
10.4 Das Gesetz der großen Zahlen ... 182
10.5 Schlussfolgerung ... 182
10.6 Bibliografie ... 183

11 Druck von außen ... 185
11.1 Druck vermeiden ... 187
11.1.1 Commitments ... 187
11.1.2 Sauber arbeiten ... 188
11.1.3 Verhalten in der Krise ... 188
11.2 Umgang mit Druck ... 189
11.2.1 Keine Panik ... 189
11.2.2 Kommunizieren Sie ... 189
11.2.3 Verlassen Sie sich auf Ihr Fachwissen ... 189
11.2.4 Hilfe holen ... 190
11.3 Schlussfolgerung ... 190

12 Teamwork ... 191
12.1 Programmierer kontra Menschen ... 193
12.1.1 Programmierer kontra Arbeitgeber ... 193
12.1.2 Programmierer kontra Programmierer ... 196
12.2 Kleinhirne ... 198
12.3 Schlussfolgerung ... 199

13 Teams und Projekte ... 201
13.1 Harmoniert es? ... 201
13.1.1 Das zusammengeschweißte Team ... 202
13.1.2 Aber wie managt man so etwas? ... 204
13.1.3 Das Dilemma des Product Owner ... 204
13.2 Schlussfolgerung ... 205
13.3 Bibliografie ... 205

14 Mentoring, Lehrzeiten und die Handwerkskunst ... 207
14.1 Der Grad des Versagens ... 207
14.2 Mentoring ... 208
14.2.1 Digi-Comp I – Mein erster Computer ... 208
14.2.2 Die ECP-18 in der Highschool ... 210
14.2.3 Unkonventionelles Mentoring ... 212
14.2.4 Schicksalsschläge ... 213
14.3 Die Lehrzeit ... 214
14.3.1 Die Lehrzeit bei der Software ... 215
14.3.2 Die Realität ... 216
14.4 Die Handwerkskunst ... 217
14.4.1 Menschen überzeugen ... 217
14.5 Schlussfolgerung ... 218

A Werkzeuge und Hilfsmittel ... 219
A.1 Tools ... 221
A.2 Quellcodekontrolle ... 221
A.2.1 Ein »Enterprise«-System der Quellcodekontrolle ... 221
A.2.2 Pessimistisches kontra optimistisches Locking ... 221
A.2.3 CVS/SVN ... 222
A.2.4 git ... 223

A.3 IDE/Editor ... 226
A.3.1 vi ... 226
A.3.2 Emacs ... 226
A.3.3 Eclipse/IntelliJ ... 226
A.3.4 TextMate ... 227
A.4 Issue-Tracking-Systeme ... 227
A.4.1 Bug-Zähler ... 228
A.5 Continuous Build ... 228
A.6 Tools für Unit-Tests ... 229
A.7 Tools für Komponententests ... 230
A.7.1 Die »Definition of Done« ... 230
A.7.2 FitNesse ... 230
A.7.3 Andere Tools ... 231
A.8 Tools für Integrationstests ... 231
A.9 UML/MDA ... 232
A.9.1 Die Details ... 232
A.9.2 Keine Hoffnung, keine Änderung ... 234
A.10 Schlussfolgerung ... 234

Index ... 235