

Inhaltsverzeichnis

	Vorwort	11
	Einleitung	13
	Danksagungen	17
	Über den Autor	19
1	Einführung in agile Softwareentwicklung	21
1.1	Ursprung der agilen Softwareentwicklung	22
1.2	Snowbird	29
1.3	Nach Snowbird	32
1.4	Überblick über agile Softwareentwicklung	32
	1.4.1 Das Eiserne Kreuz	33
	1.4.2 Diagramme an der Wand	33
	1.4.3 Das Erste, was man weiß	36
	1.4.4 Das Meeting	36
	1.4.5 Die Analysephase	37
	1.4.6 Die Designphase	38
	1.4.7 Die Implementierungsphase	38
	1.4.8 Die Todesmarsch-Phase	39
	1.4.9 Übertreibung?	40
	1.4.10 Ein besseres Verfahren	40
	1.4.11 Iteration Null	41
	1.4.12 Agile Softwareentwicklung liefert Daten	42
	1.4.13 Hoffnung vs. Management	43
	1.4.14 Das Eiserne Kreuz managen	44
	1.4.15 Kreis des Lebens	48
1.5	Zusammenfassung	50
2	Die Bedeutung agiler Softwareentwicklung	51
2.1	Professionalität	52
	2.1.1 Software ist allgegenwärtig	52
	2.1.2 Wir beherrschen die Welt	54
	2.1.3 Die Katastrophe	55

2.2	Berechtigte Erwartungen.	56
2.2.1	Wir werden keinen Mist abliefern!	56
2.2.2	Kontinuierliche technische Einsatzfähigkeit	58
2.2.3	Beständige Produktivität	59
2.2.4	Kostengünstige Anpassungsfähigkeit	61
2.2.5	Kontinuierliche Verbesserungen	62
2.2.6	Unerschrockene Kompetenz	63
2.2.7	Die Qualitätssicherung sollte nichts zu bemängeln haben . . .	64
2.2.8	Testautomatisierung.	64
2.2.9	Wir springen füreinander ein	65
2.2.10	Ehrliche Schätzungen.	66
2.2.11	Man muss auch »Nein« sagen können.	67
2.2.12	Kontinuierliches aggressives Lernen.	67
2.2.13	Mentoring	67
2.3	Bill of Rights.	68
2.3.1	Bill of Rights für Kunden.	68
2.3.2	Bill of Rights für Entwickler.	68
2.3.3	Kunden	69
2.3.4	Entwickler	70
2.4	Zusammenfassung	72
3	Unternehmensbezogene Praktiken	73
3.1	Planung	73
3.1.1	Trivariate Analysemethoden	74
3.1.2	Stories und Points.	75
3.1.3	Stories und Geldautomaten	76
3.1.4	Stories	82
3.1.5	Aufwandsschätzung von Stories	84
3.1.6	Handhabung der Iteration	86
3.1.7	Die Demo	88
3.1.8	Velocity	88
3.2	Kleine Releases.	90
3.2.1	Eine kurze Geschichte der Versionsverwaltung	90
3.2.2	Magnetbänder.	92
3.2.3	Festplatten und SCCS	92
3.2.4	Subversion.	93
3.2.5	Git und Tests.	93

3.3	Akzeptanztests	94
3.3.1	Werkzeuge und Methoden	96
3.3.2	Verhaltensgetriebene Entwicklung	96
3.3.3	Vorgehensweise	97
3.4	Gesamtes Team	99
3.4.1	Zusammenarbeit an einem Ort.	100
3.5	Zusammenfassung	102
4	Teambezogene Praktiken	103
4.1	Metapher	103
4.1.1	Domain-Driven Design	105
4.2	Nachhaltiges Tempo	106
4.2.1	Überstunden	106
4.2.2	Marathon.	107
4.2.3	Engagement	108
4.2.4	Schlaf.	109
4.3	Gemeinsame Eigentümerschaft.	109
4.3.1	Akte X	110
4.4	Kontinuierliche Integration	111
4.4.1	Kontinuierliche Builds.	112
4.4.2	Disziplin	113
4.5	Stand-up-Meetings	114
4.5.1	Schweine und Hühner?	115
4.5.2	Ausrufen	115
4.6	Zusammenfassung	115
5	Technische Praktiken	117
5.1	Testgetriebene Entwicklung	117
5.1.1	Doppelte Buchführung	118
5.1.2	Die drei Regeln testgetriebener Entwicklung.	119
5.1.3	Debugging.	120
5.1.4	Dokumentation.	120
5.1.5	Spaß.	121
5.1.6	Vollständigkeit	121
5.1.7	Design	123
5.1.8	Mut	123
5.2	Refactoring.	125
5.2.1	Rot/Grün/Refactoring	126
5.2.2	Umfangreichere Refactorings	127

5.3	Einfaches Design	127
5.3.1	Umfang des Designs	128
5.4	Pair Programming	129
5.4.1	Was ist Pair Programming?	129
5.4.2	Warum Paarbildung?	130
5.4.3	Pair Programming als Code-Review	130
5.4.4	Was ist mit den Kosten?	131
5.4.5	Nur zwei?	131
5.4.6	Management	131
5.5	Zusammenfassung	132
6	Agil werden	133
6.1	Agile Werte	133
6.1.1	Mut	133
6.1.2	Kommunikation	134
6.1.3	Feedback	134
6.1.4	Einfachheit	134
6.2	Menagerie	135
6.3	Transformation	136
6.3.1	Täuschungsmanöver	136
6.3.2	Löwenbabys	137
6.3.3	Weinen	137
6.3.4	Moral	138
6.3.5	Vortäuschen	138
6.3.6	Erfolg in kleineren Organisationen	139
6.3.7	Individueller Erfolg und Migration	139
6.3.8	Agile Organisationen aufbauen	139
6.4	Coaching	140
6.4.1	Scrum Master	141
6.5	Zertifizierung	141
6.5.1	Ernst zu nehmende Zertifizierung	142
6.6	Agile Softwareentwicklung und große Teams	142
6.7	Werkzeuge der agilen Softwareentwicklung	145
6.7.1	Software als Werkzeug	145
6.7.2	Was zeichnet ein effektives Werkzeug aus?	146
6.7.3	Physische Hilfsmittel	148
6.7.4	Automatisierungszwang	148
6.7.5	ALMs für Besserverdiener	149

6.8	Coaching – eine alternative Sichtweise	152
6.8.1	Viele Wege führen zur agilen Softwareentwicklung	152
6.8.2	Vom Prozess-Experten zum Experten für agile Software- entwicklung.	152
6.8.3	Bedarf für Coaching	153
6.8.4	Coaching ist nicht gleich Coaching.	154
6.8.5	Jenseits von ICP-ACC	154
6.8.6	Coaching-Tools.	155
6.8.7	Professionelle Coaching-Fähigkeiten reichen nicht	155
6.8.8	Coaching in Umgebungen mit mehreren Teams	156
6.8.9	Agile Softwareentwicklung mit mehreren Teams	156
6.8.10	Agile Verfahrensweisen und Coaching nutzen, um agil zu werden	157
6.8.11	Weiterentwicklung der Adaption	158
6.8.12	Auf Kleines konzentrieren und Großes erreichen.	160
6.8.13	Die Zukunft des Coachings.	161
6.9	Zusammenfassung (wieder von Bob)	161
7	Craftsmanship	163
7.1	Agile Katerstimmung	164
7.2	Falsche Erwartungen.	166
7.3	Auseinanderrücken	167
7.4	Software Craftsmanship	168
7.5	Ideologie vs. Methodologie	170
7.6	Gibt es bei der Software Craftsmanship feste Praktiken?	171
7.7	Der Mehrwert ist wichtig, nicht die Praktik.	171
7.8	Praktiken erörtern	172
7.9	Auswirkungen auf Einzelpersonen	173
7.10	Auswirkungen auf unsere Branche	174
7.11	Auswirkungen auf Unternehmen	174
7.12	Craftsmanship und agile Softwareentwicklung	175
7.13	Zusammenfassung	176
8	Schlussbemerkung	177
	Nachwort	179
	Stichwortverzeichnis	183