

Inhalt

Vorwort zur 3. Auflage	XV
1 Einführung	1
1.1 Warum dieses Buch?	2
1.2 Struktur und Aufbau	3
1.3 Dankeschön	5
1.4 Feedback	6
2 Beispiel: Scrumcoaches.com	7
2.1 Das Projekt	8
2.2 Der Entwicklungsprozess	9
2.3 Die Beteiligten	9
2.4 Die Anforderungen	10
2.5 Priorisieren und Schätzen des Product Backlog	11
2.5.1 Priorisieren	12
2.5.2 Schätzen	13
2.6 Sprint-Planung	14
2.6.1 Sprint-Ziel	14
2.6.2 Entwicklungsgeschwindigkeit	15
2.6.3 Analyse der User Stories	15
2.6.4 Design der User Stories	15
2.7 Sprint-Durchführung	17
2.8 Messen des Sprint-Fortschritts	19
2.9 Am Ende des Sprint	20
2.9.1 Sprint-Review	20
2.9.2 Sprint-Retrospektive	20
2.10 Die Arbeit geht weiter	22
2.11 Zusammenfassung	23
2.12 Wie geht es weiter?	23

4	User Stories	49
4.1	Was sind User Stories?	50
4.1.1	Story-Karte	51
4.1.2	Konversation	52
4.1.3	Akzeptanzkriterien	52
4.2	Warum User Stories?	53
4.3	User Stories schreiben	54
4.3.1	Die Sprache des Kunden	55
4.3.2	Benutzerrollen	55
4.3.3	User-Story-Muster	57
4.3.4	Epics	57
4.3.5	Themen	59
4.3.6	Wie viel Detail?	60
4.3.7	Keine Technik	61
4.3.8	Keine Benutzeroberfläche	61
4.4	Eigenschaften guter User Stories	61
4.4.1	Independent – Unabhängige User Stories	61
4.4.2	Negotiable – Verhandelbare User Stories	62
4.4.3	Valuable – Wertvolle User Stories	62
4.4.4	Estimatable – Schätzbare User Stories	63
4.4.5	Small – Kleine User Stories	63
4.4.6	Testable – Testbare User Stories	64
4.5	Zusammenfassung	65
4.6	Wie geht es weiter?	65
5	Agiles Schätzen	67
5.1	Was ist agiles Schätzen?	68
5.1.1	Relative Größe statt Dauer	68
5.1.2	Schätzen in Story Points	69
5.1.3	Wo bleibt die Dauer?	70
5.1.4	Argumentationshilfe für Story Points	70
5.2	Schätzen von User Stories	71
5.2.1	Größenordnungen und Punktesequenzen	72
5.2.2	Planungspoker	74
5.2.2.1	Schätzen im Team	76
5.2.2.2	Referenz-Story und Triangularisierung	76
5.2.2.3	Planungspoker funktioniert	78
5.2.3	Wann schätzen?	78
5.3	Zusammenfassung	79
5.4	Wie geht es weiter?	79

6	Agiles Planen	81
6.1	Was macht Planung agil?	81
6.2	Velocity	83
6.2.1	Tatsächliche Velocity	83
6.2.2	Angenommene Velocity	84
6.2.2.1	Angenommene Velocity = Tatsächliche Velocity	85
6.2.2.2	Mittlere Velocity	86
6.2.3	Velocity-basierte Planung	87
6.2.4	Nachhaltige Velocity	88
6.3	Agile Planung funktioniert	90
6.3.1	Velocity korrigiert Schätzfehler	90
6.3.2	Neubewertung von User Stories	91
6.3.3	Urlaub, Krankheit und ähnliche Ereignisse	92
6.3.4	Der Plan entsteht	92
6.4	Zusammenfassung	93
6.5	Wie geht es weiter?	93
7	User Stories fürs Product Backlog	95
7.1	Das Product Backlog	95
7.2	Das Product Backlog füllen	98
7.2.1	Anforderungswshops	99
7.2.2	Interviews, Markt-Feedback und Abstimmungsrunden	100
7.2.3	Überarbeitung und Pflege des Product Backlog	101
7.3	User Stories priorisieren	102
7.3.1	Finanzieller Wert	102
7.3.2	Kosten	103
7.3.3	Kundenzufriedenheit nach Kano	104
7.3.4	Risiko	105
7.3.5	Abhängigkeiten	106
7.3.6	Priorisierende Faktoren abwägen	106
7.3.7	MuSCoW-Priorisierung	107
7.4	User Stories schneiden	108
7.4.1	Vertikales Schneiden	109
7.4.2	Schneiden nach Daten	110
7.4.3	Schneiden nach Aufwand	111
7.4.4	Schneiden nach Forschungsanteilen	111
7.4.5	Schneiden nach Qualität	112
7.4.6	Schneiden nach Benutzerrolle	113

7.4.7	Schneiden nach Akzeptanzkriterien	113
7.4.8	Schneiden nach technischer Voraussetzung	114
7.5	Andere Anforderungen	115
7.5.1	Anforderungen umformulieren	115
7.5.2	Constraints	116
7.5.3	Fehler	117
7.5.4	Technisches Backlog	117
7.6	Zusammenfassung	118
7.7	Wie geht es weiter?	119
8	User Story Mapping	121
8.1	User Story Maps	122
8.2	Eine Story Map erstellen	123
8.2.1	Schritt 1: User Tasks ermitteln	124
8.2.2	Schritt 2: Gruppen bilden – User Activities	125
8.2.3	Schritt 3: Ordnung schaffen	126
8.2.4	Schritt 4: User Tasks durchlaufen = Geschichten erzählen	126
8.2.5	Schritt 5: User Stories schreiben	127
8.3	Warum Story Mapping?	128
8.3.1	Basis für gute Product Backlogs	128
8.3.2	Kleinstmögliche Releases	129
8.3.3	Motivation und Einsicht für alle Stakeholder	129
8.3.4	Lückenlosigkeit	129
8.3.5	Softwarearchitektur	130
8.3.6	Multi-Team-Setups	130
8.4	Von der Story Map zum Product Backlog	130
8.4.1	User Stories schreiben	133
8.4.2	Die Story Map ersetzt das Product Backlog	133
8.5	Zusammenfassung	133
8.6	Wie geht es weiter?	134
9	Sprint-Planung	135
9.1	Überblick und Ablauf	135
9.2	Beteiligte	136
9.3	Ergebnisse	136
9.4	Vorbereitung	139
9.4.1	Sprint Velocity	139
9.4.1.1	Anpassen der Velocity	139
9.4.1.2	Bugfixing, Refactoring und andere Aufgaben	140

9.4.2	Story-Auswahl.....	141
9.4.3	Sprint-Länge	142
9.5	Sprint Planning 1.....	143
9.5.1	Ablauf	143
9.5.2	Sprint-Ziel – Warum führen wir den Sprint durch?	144
9.5.3	Vorstellung, Analyse und Commitment	144
9.5.4	Fehler und andere technische Aufgaben	146
9.6	Sprint Planning 2.....	147
9.6.1	Ablauf	148
9.6.2	Story-Design.....	149
9.6.3	Tasks schneiden	150
9.6.3.1	Taskgröße.....	151
9.6.3.2	Schneidetechniken.....	151
9.6.3.3	Ungeplante Tasks.....	152
9.6.4	Tasks schätzen?	152
9.6.4.1	Taskschätzungen sind sinnvoll	153
9.6.4.2	Taskschätzungen sind unsinnig.....	153
9.6.4.3	Keine Empfehlung	154
9.6.5	Das Sprint Backlog	155
9.6.6	Fehler und andere technischen Aufgaben verteilen	156
9.6.7	Was tun, wenn es länger wird?	156
9.7	Abschluss.....	157
9.8	Zusammenfassung.....	158
9.9	Wie geht es weiter?.....	158
10	Sprint-Durchführung	159
10.1	Die eigentliche Arbeit beginnt	159
10.2	Wer macht was?	161
10.2.1	Das Team	161
10.2.2	Der Product Owner	162
10.2.3	Der ScrumMaster	162
10.3	Story für Story Richtung Sprint-Ziel	163
10.3.1	Wie viele User Stories zurzeit?	164
10.3.2	Arbeit an einer User Story	164
10.3.3	Definition of Done	164
10.3.4	Abnahme fertiger User Stories	165
10.3.4.1	Entwicklertest.....	165
10.3.4.2	Akzeptanztest.....	166

10.3.4.3	QA-Abnahme	166
10.3.4.4	Frühestmögliche Fehlerbehebung	167
10.4	Daily Scrum	167
10.4.1	Aktualisierung des Taskboard	168
10.4.2	Ein guter Zeitpunkt	169
10.4.3	Ein guter Ort	170
10.4.4	Wer ist noch dabei?	170
10.4.5	Was macht der ScrumMaster?	171
10.5	Unterbrechungen	172
10.6	Messen und Anpassen	173
10.6.1	Bug- und technische Burndown-Charts	174
10.6.2	Was tun, wenn es eng wird?	175
10.7	Reguläres Sprint-Ende	176
10.8	Sprint-Review	177
10.8.1	Vorbereitung	177
10.8.2	Ort, Zeitpunkt und Teilnehmer	177
10.8.3	Ziel	178
10.8.4	Ablauf	178
10.9	Das Team organisiert sich	179
10.9.1	Verantwortung übernehmen	179
10.9.2	Das Team machen lassen und aus Fehlern lernen	180
10.9.3	Den Product Owner einbeziehen	180
10.9.4	Software-Pull-Systeme	181
10.9.5	Das Team bei der Arbeit mit Tasks coachen	182
10.9.6	Einzelgespräche	182
10.10	Sprint Best Practices	183
10.10.1	Source Code Management und Story-Banches	183
10.10.2	Kontinuierliches Integrieren	184
10.10.3	Automatisierung	184
10.10.4	Verständlicher Quellcode	185
10.10.5	Elektronische Sprint Backlogs und Burndown-Charts	185
10.11	Zusammenfassung	186
10.12	Wie geht es weiter?	186

11	User Stories Akzeptanztesten	187
11.1	Was ist Akzeptanztesten?	187
11.1.1	Akzeptanzkriterien	188
11.1.1.1	Akzeptanzkriterien sind Erwartungen	188
11.1.1.2	Akzeptanzkriterien sind Geschäftsregeln	189
11.1.2	Akzeptanztests	189
11.1.3	Akzeptanztesten	190
11.2	Akzeptanzkriterien schreiben	191
11.2.1	Vom Akzeptanzkriterium zum Akzeptanztest	191
11.2.2	Merkmale guter Akzeptanzkriterien	192
11.2.3	Akzeptanzkriterien auch für Epics?	194
11.3	Beispiel: Suche nach Coaches	194
11.4	Kleine Bausteine: Auf dem Weg zur DSL	195
11.5	Akzeptanztesten während des Sprint	196
11.6	Die hohe Schule: Akzeptanztest-getriebene Entwicklung	198
11.6.1	ATDD-Beispiel: Suche nach Coaches	199
11.6.2	Product Owner love writing Tests?	201
11.6.2.1	Alternative JCriteria	201
11.7	Lohnt sich das Ganze?	202
11.8	Zusammenfassung	202
11.9	Wie geht es weiter?	203
12	Sprint-Retrospektive	205
12.1	Nach dem Sprint ist vor dem Sprint	206
12.2	Ablauf von Retrospektiven	206
12.3	Retrospektiven vorbereiten	208
12.4	Retrospektiven leiten	208
12.5	Agenda und Check-in	209
12.6	Phase 1: Daten sammeln	210
12.6.1	Erstellung einer Timeline	211
12.6.2	Erweiterung der Timeline um Energiepunkte	212
12.7	Phase 2: Einsichten generieren	213
12.7.1	Positiv- / Delta-Liste	213
12.7.2	Warum-Fragen	214
12.8	Phase 3: Entscheiden, was zu tun ist	214
12.9	Phase 4: Ziele formulieren und Aktionen planen	215
12.10	Abschluss	216
12.11	Themenorientierte Retrospektiven	217
12.12	Zusammenfassung	218
12.13	Wie geht es weiter?	219

13	Agile Releaseplanung	221
13.1	Releaseplanung	221
13.1.1	Themen-Releases	221
13.1.2	Datum-Releases	222
13.1.3	Releaseplanungs-Workshop	223
13.1.4	Was macht die Planung agil?	223
13.2	Planungs-Velocity	224
13.2.1	Durchführung von Test-Sprints	224
13.2.2	Historische Daten	225
13.2.3	Das Team bestimmen lassen	225
13.2.4	Auswahl eines Verfahrens	225
13.3	Der Releaseplan	226
13.4	Sichere Planung	227
13.4.1	Sichere Velocity	227
13.4.2	Sicherheit durch weniger wichtige User Stories	228
13.5	Monitoring und Aktualisierung	229
13.6	Zusammenfassung	230
13.7	Wie geht es weiter?	230
14	Verticals – SCRUM@OTTO	231
14.1	Warum ich über diese Geschichte schreibe	231
14.2	Die Vorgeschichte	233
14.3	Das Lhotse-Projekt – Zahlen, Daten, Fakten	234
14.4	Das Team – Menschen im Mittelpunkt	235
14.5	Triaden – die Führung eines Teams	237
14.6	Die Triade – Rollenbeschreibungen	237
14.6.1	Der Projektmanager – Project-Lead	238
14.6.2	Der Produktmanager – Business-Designer	239
14.6.3	Der Team-Architekt – Technical-Designer	239
14.7	Die TD-Runde	241
14.8	Die Otto-Architektur in Vertikalen	242
14.8.1	Warum die klassische IT versagt	242
14.8.2	Warum vertikale Schnitte helfen	245
14.8.3	Was eine Vertikale ist	246
14.8.4	Wie vertikale Schnitte gefunden werden können	248
14.9	Makro- und Mikroarchitektur	250
14.9.1	Makroarchitektur	251
14.9.2	Mikroarchitektur	251

14.10	Werte und Leitplanken statt Richtlinien und Governance	252
14.11	Das klassische Management in der agiler werdenden Organisation.....	252
14.12	Scrum@Otto – 100 Sprints später	254
14.13	Fazit	256
Glossar		257
Literatur		265
Stichwortverzeichnis		267