

Inhaltsverzeichnis

1 Software-Prozessmodelle	1
1.1 Ein Prozessmodell – was ist das?	1
1.1.1 Einteilung in Projekttypen	2
1.1.2 Schwergewichtige und leichtgewichtige Prozessmodelle	2
1.1.3 Prozessmodell vs. Vorgehensmodell	3
1.2 Das Phasenmodell	3
1.3 Das Spiralmodell	4
1.4 Moderne iterativ-inkrementelle Prozessmodelle	6
1.4.1 V-Modell	6
1.4.2 Unified Software Development Process (UP)	6
1.4.3 Agile Prozessmodelle	7
Literatur	7
2 Das Agile Manifest	9
Literatur	11
3 Extreme Programming (XP)	13
3.1 Die fünf Werte	14
3.1.1 Kommunikation	14
3.1.2 Einfachheit	14
3.1.3 Feedback	15
3.1.4 Mut	16
3.1.5 Respekt	16
3.2 Die 14 Prinzipien	17
3.2.1 Menschlichkeit	17
3.2.2 Wirtschaftlichkeit	17
3.2.3 Wechselseitiger Vorteil	18
3.2.4 Selbstähnlichkeit	18
3.2.5 Verbesserung	18
3.2.6 Vielfältigkeit	18
3.2.7 Reflexion	19
3.2.8 Fluss	19
3.2.9 Gelegenheit	19
3.2.10 Redundanz	19
3.2.11 Fehlschlag	20

3.2.12	Qualität	20
3.2.13	Kleine Schritte	20
3.2.14	Akzeptierte Verantwortung	20
3.3	Prozessschritte und traditionelle XP-Praktiken	20
3.3.1	Planung	21
3.3.2	Design der Software	27
3.3.3	Kodieren	30
3.3.4	Testen der Software	36
3.4	Erweiterte XP-Praktiken	37
3.4.1	Primärpraktiken	38
3.4.2	Folgepraktiken	40
3.4.3	Unterschiede zwischen erweiterten und traditionellen XP-Praktiken	42
3.5	Berühmte XP-Praktiken	43
3.5.1	Erstellen von User Stories	44
3.5.2	Paarweises Programmieren	44
3.5.3	Collective Code Ownership, Shared Code	44
3.5.4	Kontinuierliche Code-Integration, eine Codebasis	45
3.5.5	Kunde im Team, Einbeziehung des Kunden	45
Literatur	45
4	Crystal und Crystal Clear	47
4.1	Teamgröße und Risiko – die Crystal-Familie	47
4.2	Die sieben Crystal-Eigenschaften	49
4.2.1	Regelmäßige Lieferung	49
4.2.2	Reflektierte Verbesserung	49
4.2.3	Verdichtete oder osmotische Kommunikation	50
4.2.4	Persönliche Sicherheit	51
4.2.5	Schwerpunkte bilden	51
4.2.6	Einfache Kontaktaufnahme mit Endanwendern	52
4.2.7	Technische Umgebung mit automatisierten Tests, Konfigurationsmanagement und regelmäßige Integrationen	52
4.3	Crystal Clear	53
4.3.1	Eigenschaften und Praktiken	53
4.3.2	Rollen im Projektteam	56
Literatur	59
5	Scrum	61
5.1	Scrum – die Projektrollen	61
5.1.1	Product Owner	62
5.1.2	Team	63
5.1.3	ScrumMaster	65
5.1.4	Weitere Scrum-Rollen	67
5.1.5	Gefahr durch Rollenmissbrauch	67
5.2	Scrum – der Prozess	68
5.2.1	Scrum-Flow – Überblick	68
5.2.2	Sprint – Details	69

5.3	Scrum-Artefakte	73
5.3.1	Product Backlog	73
5.3.2	Sprint Backlog	75
5.3.3	Releaseplan und Burndown Chart	75
Literatur		77
6	Experimentelles Software-Engineering im studentischen Labor	79
6.1	Die Projekte im studentischen Labor	80
6.2	Randbedingungen der studentischen Sessions	82
6.3	Veränderung der klassischen Projektrollen in agilen Projekten	84
6.3.1	Der Projektmanager im agilen Projekt	84
6.3.2	Der Qualitätsmanager im agilen Projekt	88
6.4	Neue Teamrolle – der Integrationsingenieur	92
6.5	Veränderung agiler Praktiken und Prozesse in der Praxis	93
6.5.1	Design von User Stories	93
6.5.2	Collective Code Ownership	94
6.5.3	Mini-Team-Größe: Sind XP-Paare erfolgreich?	94
6.5.4	Der Weg zur erfolgreichen Software-Integration	100
6.5.5	Crystal und die reflektierte Verbesserung	103
6.5.6	Scrum im studentischen Labor	107
6.5.7	Hierarchische Prozesse unter dem agilen Deckmantel	115
Literatur		117
7	MAP – Meta Agile Process Model	119
7.1	Team-Psychologie – Landkarte der Verhaltensweisen im Team	119
7.2	Das Super-Team	122
7.3	Was ist MAP?	123
7.4	MAP – die Projektrollen im Team	124
7.4.1	Kunde	125
7.4.2	Kommunikationsmanager	125
7.4.3	Integrationsingenieur	127
7.4.4	Team	128
7.4.5	Prozessverantwortlicher – der MAP-Beobachter	129
7.4.6	Projektrollen in der Landkarte der Verhaltensweisen	130
7.5	MAP Cycle – der Referenzprozess	131
7.6	MAP und Scrum	133
7.6.1	Vergleich der Rollen	133
7.6.2	Bestimmung des Product Owners	136
7.6.3	Prozess	139
7.6.4	Artefakte	141
7.6.5	MAP und Scrum – geht das?	142
7.6.6	Vorteile für Scrum-Anwender	143
7.7	MAP im regulierten Umfeld	143
7.8	MAP-Projekt in der Industrie	146
Literatur		147