

---

# Contents

<b>Preface</b>	<b>vii</b>
Description and assessment	vii
Keeping a cool head	viii
Previous attempts	ix
Structure of the book	ix
Perspective and scope	x
Analysis: instinctive, experiential, logical or empirical?	xi
Free critical inquiry	xii
Acknowledgments	xiii
<b>Contents</b>	<b>xv</b>
<b>1 OVERVIEW</b>	<b>1</b>
1.1 VALUES	2
1.2 PRINCIPLES	4
Organizational principles	5
Technical principles	6
1.3 ROLES	7
1.4 PRACTICES	8
Organizational practices	8
Technical practices	9
1.5 ARTIFACTS	10
Virtual artifacts	10
Material artifacts	11
1.6 A FIRST ASSESSMENT	12
Not new and not good	12
New and not good	13
Not new but good	14
New and good!	14

<b>2 DECONSTRUCTING AGILE TEXTS</b>	<b>17</b>
2.1 THE PLIGHT OF THE TRAVELING SEMINARIST	17
Proof by anecdote	18
When writing beats speaking	19
Discovering the gems	20
Agile texts: reader beware!	21
2.2 THE TOP SEVEN RHETORICAL TRAPS	22
Proof by anecdote	22
Slander by association	23
Intimidation	23
Catastrophism	26
All-or-nothing	27
Cover-your-behind	27
Unverifiable claims	28
Postscript: you have been ill-served by the software industry!	30
<b>3 THE ENEMY: BIG UPFRONT ANYTHING</b>	<b>31</b>
3.1 PREDICTIVE IS NOT WATERFALL	31
3.2 REQUIREMENTS ENGINEERING	32
Requirements engineering techniques	32
Agile criticism of upfront requirements	32
The waste criticism	33
The change criticism	35
The domain and the machine	36
3.3 ARCHITECTURE AND DESIGN	37
Is design separate from implementation?	37
Agile methods and design	39
3.4 LIFECYCLE MODELS	41
3.5 RATIONAL UNIFIED PROCESS	42
3.6 MATURITY MODELS	43
CMMI in plain English	44
The Personal Software Process	46
CMMI/PSP and agile methods	46
An agile maturity scale	47
<b>4 AGILE PRINCIPLES</b>	<b>49</b>
4.1 WHAT IS A PRINCIPLE?	49
4.2 THE OFFICIAL PRINCIPLES	50
4.3 A USABLE LIST	51
4.4 ORGANIZATIONAL PRINCIPLES	51

---

---

Put the customer at the center	51
Let the team self-organize	53
Maintain a sustainable pace	56
Develop minimal software	58
Accept change	68
<b>4.5 TECHNICAL PRINCIPLES</b>	<b>70</b>
Develop iteratively	70
Treat tests as a key resource	75
Do not start any new development until all tests pass	76
Test first	77
Express requirements through scenarios	77
<b>5 AGILE ROLES</b>	<b>79</b>
5.1 MANAGER	79
5.2 PRODUCT OWNER	80
5.3 TEAM	80
Self-organizing	80
Cross-functional	81
5.4 MEMBERS AND OBSERVERS	82
5.5 CUSTOMER	82
5.6 COACH, SCRUM MASTER	84
5.7 SEPARATING ROLES	86
<b>6 AGILE PRACTICES: MANAGERIAL</b>	<b>89</b>
6.1 SPRINT	89
Sprint basics	89
The closed-window rule	90
Sprint: an assessment	91
6.2 DAILY MEETING	91
6.3 PLANNING GAME	93
6.4 PLANNING POKER	94
6.5 ONSITE CUSTOMER	96
6.6 OPEN SPACE	96
6.7 PROCESS MINIATURE	97
6.8 ITERATION PLANNING	98
6.9 REVIEW MEETING	99
6.10 RETROSPECTIVE	99
6.11 SCRUM OF SCRUMS	99
6.12 COLLECTIVE CODE OWNERSHIP	100

---

---

The code ownership debate	100
Collective ownership and cross-functionality	102
<b>7 AGILE PRACTICES: TECHNICAL</b>	<b>103</b>
7.1 DAILY BUILD AND CONTINUOUS INTEGRATION	103
7.2 PAIR PROGRAMMING	105
Pair programming concepts	106
Pair programming versus mentoring	107
Mob programming	107
Pair programming: an assessment	107
7.3 CODING STANDARDS	109
7.4 REFACTORING	109
The refactoring concept	109
Benefits and limits of refactoring	110
Incidental and essential changes	112
Combining a priori and a posteriori approaches	113
7.5 TEST-FIRST AND TEST-DRIVEN DEVELOPMENT	113
The TDD method of software development	113
An assessment of TFD and TDD	115
<b>8 AGILE ARTIFACTS</b>	<b>117</b>
8.1 CODE	117
8.2 TESTS	117
8.3 USER STORIES	119
8.4 STORY POINTS	121
8.5 VELOCITY	123
8.6 DEFINITION OF DONE	125
8.7 WORKING SPACE	125
8.8 PRODUCT BACKLOG, ITERATION BACKLOG	126
8.9 STORY CARD, TASK CARD	127
8.10 TASK AND STORY BOARDS	127
8.11 BURNDOWN AND BURNUP CHARTS	128
8.12 IMPEDIMENT	129
8.13 WASTE, TECHNICAL DEBT, DEPENDENCY, DEPENDENCY CHARTS	129
<b>9 AGILE METHODS</b>	<b>133</b>
9.1 METHODS AND METHODOLOGY	133
Terminology	133
The fox and the hedgehog	133
9.2 LEAN SOFTWARE AND KANBAN	134

---

---

Lean Software's Big Idea	134
Lean Software's principles	134
Lean Software: an assessment	135
Kanban	136
<b>9.3 EXTREME PROGRAMMING</b>	<b>137</b>
XP's Big Idea	137
XP: the unadulterated source	137
Key XP techniques	138
Extreme Programming: an assessment	139
<b>9.4 SCRUM</b>	<b>139</b>
Scrum's Big Idea	139
Key Scrum practices	140
Scrum: an assessment	140
<b>9.5 CRYSTAL</b>	<b>141</b>
Crystal's Big Idea	141
Crystal principles	141
Crystal: an assessment	142
<b>10 DEALING WITH AGILE TEAMS</b>	<b>145</b>
10.1 GRAVITY STILL HOLDS	145
10.2 THE EITHER-WHAT-OR-WHEN FALLACY	146
<b>11 THE UGLY, THE HYPE AND THE GOOD: AN ASSESSMENT OF THE AGILE APPROACH</b>	<b>149</b>
11.1 THE BAD AND THE UGLY	149
Deprecation of upfront tasks	149
User stories as a basis for requirements	150
Feature-based development and ignorance of dependencies	150
Rejection of dependency tracking tools	150
Rejection of traditional manager tasks	150
Rejection of upfront generalization	151
Embedded customer	151
Coach as a separate role	151
Test-driven development	151
Deprecation of documents	151
11.2 THE HYPED	152
11.3 THE GOOD	153
11.4 THE BRILLIANT	154
<b>Bibliography</b>	<b>155</b>
<b>Index</b>	<b>163</b>