

Inhaltsübersicht

1	Einleitung	1
2	Überblick über SAFe	7
3	Agile Teams in SAFe	23
4	Die Programmebene	37
5	Rollen auf der Programmebene	79
6	Portfoliomanagement	89
7	Portfolio-Rollen	107
8	Metriken und Praktiken	113
9	Leadership	133
10	SAFe erfolgreich einführen	147
11	Agile Architektur skalieren	159
12	SAFe im Kontext	167
	Literaturverzeichnis	195
	Index	197

Inhaltsverzeichnis

1	Einleitung	1
1.1	Was macht Agilität erfolgreich	1
1.2	Warum skalieren	2
1.3	Die neuen Herausforderungen an Agilität	4
2	Überblick über SAFe	7
2.1	Woher kommt SAFe	8
2.2	Die Struktur von SAFe	9
2.2.1	Teamebene	10
2.2.2	Programmebene (Agile Release Train)	10
2.2.3	Portfolioebene	12
2.3	Strukturierung von Anforderungen	13
2.4	Rollen	13
2.5	Schlüsselwerte	14
2.6	Agile Architektur und Software Engineering	15
2.7	Integrierte Qualitätskultur	15
2.8	SAFe-Prinzipien für einen effektiven Entwicklungsfluss	17
2.9	Leadership	21
3	Agile Teams in SAFe	23
3.1	Zwei Wurzeln agiler Entwicklung	23
3.2	Vier Vorteile agiler Teams	24
3.3	Beschleunigtes Lernen ist Team-Lernen	25
3.4	Vier Vorteile agiler Softwaretechniken	28
3.5	Herausforderungen in skalierten Umgebungen	29
3.6	Teams bei SAFe	33

4	Die Programmebene	37
4.1	Der Agile Release Train	38
4.1.1	Arbeitsprinzipien	39
4.1.2	Rollen	39
4.1.3	Release Trains aufsetzen	40
4.1.4	Arbeiten im Programminkrement	41
4.2	Das Programminkrement	42
4.2.1	Das PI als Super-Sprint	42
4.3	Entwickeln im Takt – liefern nach Bedarf	44
4.3.1	Entwickeln im Takt	44
4.3.2	Liefern nach Bedarf	46
4.4	Artefakte des Release Train	49
4.4.1	Die Programmvision	49
4.4.2	Die Roadmap	50
4.4.3	Das ART-Backlog	51
4.4.4	Programm-Epics	52
4.4.5	Features	52
4.4.6	Enablers oder Architektur-Features	53
4.5	Planen für den Agile Release Train	54
4.5.1	Das ART-Backlog vorbereiten	54
4.5.2	Features aufsplitten	55
4.5.3	Features zwischen Teams aufteilen	55
4.5.4	Schätzen von Features	55
4.5.5	Kapazitäten allokalieren	56
4.5.6	Die optimale Länge des Backlogs	56
4.6	Meetings	57
4.6.1	PI-Planung – Start des Programminkrements	57
4.6.2	Die zweiwöchentliche Systemdemo	62
4.6.3	Die ART-Demo	62
4.6.4	Das Inspect & Adapt-Meeting	63
4.7	Architectural Runway – die Landebahn	64
4.8	Der Innovations- und Planungs-Sprint	66
4.9	Nicht funktionale Anforderungen	69
4.10	Priorisierung	71
4.10.1	Weighted Shortest Job First (WSJF)	72
4.11	DevOps	74
4.12	Die Value-Stream-Ebene von SAFe 4.0	76

5	Rollen auf der Programmebene	79
5.1	Produktmanagement	79
5.2	Releasemanagement	81
5.3	Der Release Train Engineer	81
5.4	Der Systemarchitekt	83
5.5	Übergreifende und geteilte Ressourcen	84
5.6	Benutzerschnittstellen-Verantwortlicher	85
5.7	Die Business Owner	86
6	Portfoliomanagement	89
6.1	Die Portfolioebene	89
6.2	Die Portfoliovision	91
6.3	Strategische Themen	92
6.4	Die Bearbeitung von Epics	94
6.4.1	Was ist ein Epic?	94
6.4.2	Der Business Case eines Epics	95
6.4.3	Business- und Architektur-Epic-Kanban	97
6.4.4	Ein prototypisches Kanban-System	97
6.4.5	Schrittweise Implementierung	99
6.4.6	Priorisierung im Portfolio-Backlog	99
6.4.7	Schätzen und Planen für das Portfolio	100
6.5	Budgets	101
6.5.1	Das Problem mit klassischen Budgets	101
6.5.2	Beyond Project Cost Accounting	102
6.6	Wertströme	103
6.6.1	Den Wertstrom identifizieren	103
6.6.2	Mehrere Release Trains organisieren	104
6.6.3	Koordination über mehrere Trains	105
6.6.4	Andere Arten der Entkopplung	106
7	Portfolio-Rollen	107
7.1	Programm-Portfolio-Management	107
7.2	Epic Owner	109
7.2.1	Verantwortlichkeiten des Epic Owners	109
7.3	Der Enterprise-Architekt	110
7.3.1	Verantwortlichkeiten des Enterprise-Architekten	111

8	Metriken und Praktiken	113
8.1	Das Problem des Messens	113
8.2	Agile Evolution	114
8.3	Teambasierte Praktiken	116
8.4	Metriken für Teams	119
8.5	Metriken für den Agile Release Train	120
8.6	Portfolio-Metriken	125
8.6.1	Allgemeine Bewertungen	126
8.6.2	Das Portfolio-Kanban	127
8.6.3	Epics messen	128
8.6.4	Selbst-Assessment des Portfoliomanagement-Teams	130
8.6.5	Klassische Bewertungsmechanismen	131
9	Leadership	133
9.1	Alignment und Compliance	133
9.2	Führungsstile	134
9.2.1	Der Experte	134
9.2.2	Der Dirigent	135
9.2.3	Der Teamentwickler	135
9.2.4	Der Superheld	136
9.3	Führung in einer agilen und Lean-Umgebung	136
9.3.1	Führung bei Scrum	136
9.3.2	Führung bei Lean Development	137
9.3.3	Führung bei SAFe	137
9.4	Was motiviert Menschen	138
9.4.1	Voraussetzung für Motivation schaffen	138
9.5	Arbeiten mit Teams	141
9.5.1	Quick Win – Retrospektiven	141
9.5.2	Ein Motivations-Toolkit für Teams	142
9.6	Die Organisation entwickeln	144
9.6.1	Quick Win – Communities of Practice	144
9.6.2	Safe to Fail anstatt Fail-Safe	145
9.6.3	Organisatorische Gründe für Ineffektivität	145

10	SAFe erfolgreich einführen	147
10.1	Der SAFe-Standardweg	147
10.2	Vor dem Start	148
10.2.1	Bestandsaufnahme	148
10.2.2	Ziele und Voraussetzungen explizit machen	149
10.3	Change-Prozesse verstehen	150
10.4	Mehr als ein normaler Change-Prozess	154
10.4.1	Startpunkt eines Transformationsprozesses	154
10.5	Strategien zur Einführung	155
10.5.1	Referenzszenario	156
10.6	Nachhaltigkeit planen	157
10.7	Fehler und Fallen	157
11	Agile Architektur skalieren	159
11.1	Die Rolle der Softwarearchitektur	159
11.2	Architektur in SAFe	159
11.3	Sieben SAFe-Prinzipien zur agilen Architektur	160
11.4	Enterprise-Architektur	162
11.5	Solider Unterbau durch agile Softwarepraktiken	163
12	SAFe im Kontext	167
12.1	Wurzeln	167
12.1.1	Das Missverständnis mit dem Wasserfall	167
12.1.2	Agile ändert die Spielregeln	168
12.1.3	Agile stammt von Lean ab	168
12.1.4	Das Agile Manifest	170
12.1.5	Das Manifest der Interdependenz	172
12.2	Lean Thinking und agile Software	173
12.2.1	Das Haus von Lean	173
12.2.2	Lean Thinking	175
12.3	Agile Skalierungsframeworks im Vergleich	177
12.3.1	LeSS von Craig Larman und Bas Vodde	178
12.3.2	Scrum@Scale von Jeff Sutherland	180
12.3.3	Enterprise Scrum von Mike Beedle	182
12.3.4	Nexus von Ken Schwaber	184
12.3.5	DAD von Scott Ambler	186
12.3.6	Verschiedene Zielsetzungen der Frameworks	188
12.3.7	Starterkits – über Sinn und Unsinn agiler Frameworks ...	189

12.4	Fortgeschrittene agile Implementierungen	190
12.5	Die Grenzen von SAFe	191
12.5.1	Missverständnisse von SAFe	191
12.5.2	Wer braucht SAFe?	192
12.5.3	Feedbackschleifen schließen	194
12.6	Kultur verspeist Prozess zum Frühstück	194
	Literaturverzeichnis	195
	Index	197