

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	Solution-Ebene	12
2.2.4	Portfolioebene	13
2.3	Strukturierung von Anforderungen	13
2.4	Rollen	14
2.5	Schlüsselwerte	14
2.6	Agile Architektur und Software Engineering	15
2.7	Integrierte Qualitätskultur	16
2.8	SAFe-Prinzipien für einen effektiven Entwicklungsfluss	18
2.9	Leadership	22
3	Agile Teams in SAFe	25
3.1	Zwei Wurzeln agiler Entwicklung	25
3.2	Vier Vorteile agiler Teams	26
3.3	Beschleunigtes Lernen ist Team-Lernen	27
3.4	Vier Vorteile agiler Softwaretechniken	30
3.5	Herausforderungen in skalierten Umgebungen	31
3.6	Teams bei SAFe	35

4	Die Programmebene	39
4.1	Einführung	39
4.2	Der Agile Release Train	41
4.2.1	Arbeitsprinzipien	41
4.2.2	ARTs sind multifunktional	42
4.2.3	Rollen	42
4.2.4	Release Trains aufsetzen	43
4.2.5	Arbeiten im Programminkrement	43
4.3	Das Programminkrement	45
4.3.1	Das PI als Super-Sprint	45
4.4	Entwickeln im Takt – liefern nach Bedarf	46
4.4.1	Entwickeln im Takt	47
4.4.2	Liefern nach Bedarf	49
4.5	Artefakte des Release Train	52
4.5.1	Die Programmvision	52
4.5.2	Die Roadmap	53
4.5.3	Das ART-Backlog	53
4.5.4	Programm-Epics	54
4.5.5	Features	55
4.5.6	Enablers oder Architektur-Features	56
4.6	Planen für den Agile Release Train	57
4.6.1	Das ART-Backlog vorbereiten	57
4.6.2	Features aufsplitten	58
4.6.3	Features zwischen Teams aufteilen	58
4.6.4	Schätzen von Features	58
4.6.5	Kapazitäten allokalieren	59
4.6.6	Die optimale Länge des Backlogs	59
4.7	Meetings	60
4.7.1	PI-Planung – Start des Programminkrements	60
4.7.2	Die zweiwöchentliche Systemdemo	65
4.7.3	Die ART-Demo	66
4.7.4	Das Inspect & Adapt-Meeting	66
4.8	Architectural Runway – die Landebahn	68
4.9	Der Innovations- und Planungs-Sprint	69
4.10	Nicht funktionale Anforderungen	72
4.11	Priorisierung	74
4.11.1	Weighted Shortest Job First (WSJF)	75
4.12	DevOps	77

5	Rollen auf der Programmebene	81
5.1	Produktmanagement	81
5.2	Releasemanagement	83
5.3	Der Release Train Engineer	83
5.4	Der Systemarchitekt	85
5.5	Übergreifende und geteilte Ressourcen	86
5.6	Benutzerschnittstellen-Verantwortlicher	87
5.7	Die Business Owner	88
6	Der Solution Train	91
6.1	Die Solution-Ebene	91
6.2	Features und Capabilities	92
6.3	Solution Intent	92
6.4	Analogien zum ART	93
6.5	Prinzipien eines Solution Train	94
6.6	Neue Rollen im Solution Train	95
6.7	Koordination von ARTs und Zulieferern	96
6.8	Release Governance	97
7	Portfoliomanagement	99
7.1	Portfolios in einem Unternehmen	99
7.2	Lean Portfolio Management – Lean Budgeting	101
7.3	Strategische Themen	102
7.4	Die Bearbeitung von Epics	103
7.4.1	Was ist ein Epic?	103
7.4.2	Der Lean Business Case eines Epics	105
7.4.3	Portfolio-Kanban	107
7.4.4	Ein prototypisches Kanban-System	108
7.4.5	Schrittweise Implementierung	110
7.4.6	Priorisierung im Portfolio-Backlog	110
7.4.7	Schätzen und Planen für das Portfolio	111
7.4.8	Weiterentwicklung von Epics im Portfolio-Kanban	112
7.5	Budgets	113
7.5.1	Das Problem mit klassischen Budgets	113
7.5.2	Budgeting durch Lean Portfolio Management	114

7.6	Wertströme	115
7.6.1	Den Wertstrom identifizieren	115
7.6.2	Mehrere Release Trains organisieren	117
7.6.3	Koordination über mehrere Trains	117
7.6.4	Andere Arten der Entkopplung	118
8	Portfolio-Rollen	119
8.1	Lean Portfolio Management	119
8.2	Epic Owner	121
8.2.1	Verantwortlichkeiten des Epic Owners	121
8.3	Der Enterprise-Architekt	122
8.3.1	Verantwortlichkeiten des Enterprise-Architekten	123
9	Die Lieferpipeline	125
9.1	Überblick	125
9.2	Kontinuierliche Exploration	126
9.3	Kontinuierliche Integration	128
9.3.1	Integration von Zulieferern	131
9.4	Kontinuierliche Auslieferung	131
9.4.1	Sechs Praktiken für erfolgreiche Auslieferung	132
9.5	DevOps	134
10	Metriken und Praktiken	141
10.1	Das Problem des Messens	141
10.2	Agile Evolution	142
10.3	Teambasierte Praktiken	144
10.4	Metriken für Teams	147
10.5	Metriken für den Agile Release Train	148
10.6	Portfolio-Metriken	153
10.6.1	Allgemeine Bewertungen	154
10.6.2	Das Portfolio-Kanban	155
10.6.3	Epics messen	156
10.6.4	Selbst-Assessment des Portfoliomanagement-Teams	158
10.6.5	Klassische Bewertungsmechanismen	159

11	Leadership	161
11.1	Alignment und Compliance	161
11.2	Führungsstile	162
11.2.1	Der Experte	162
11.2.2	Der Macher	163
11.2.3	Der Katalysator oder Teamentwickler	163
11.2.4	Der Superheld	164
11.3	Führung in einer agilen und Lean-Umgebung	164
11.3.1	Führung bei Scrum	164
11.3.2	Führung bei Lean Development	165
11.3.3	Führung bei SAFe	165
11.4	Was motiviert Menschen	166
11.4.1	Voraussetzung für Motivation schaffen	166
11.5	Arbeiten mit Teams	169
11.5.1	Quick Win – Retrospektiven	169
11.5.2	Ein Motivations-Toolkit für Teams	170
11.6	Die Organisation entwickeln	172
11.6.1	Quick Win – Communities of Practice	172
11.6.2	Safe to Fail anstatt Fail-Safe	173
11.6.3	Organisatorische Gründe für Ineffektivität	173
12	SAFe erfolgreich einführen	175
12.1	Der SAFe-Standardweg	175
12.2	Vor dem Start	178
12.2.1	Bestandsaufnahme	178
12.2.2	Ziele und Voraussetzungen explizit machen	179
12.3	Change-Prozesse verstehen	180
12.4	Mehr als ein normaler Change-Prozess	183
12.4.1	Startpunkt eines Transformationsprozesses	184
12.5	Strategien zur Einführung	185
12.5.1	Referenzszenario	186
12.6	Nachhaltigkeit planen	186
12.7	Fehler und Fallen	187

13	Agile Architektur skalieren	189
13.1	Die Rolle der Softwarearchitektur	189
13.2	Architektur in SAFe	190
13.3	Sieben SAFe-Prinzipien zur agilen Architektur	190
13.4	Enterprise-Architektur	193
13.5	Solider Unterbau durch agile Softwarepraktiken	193
14	SAFe im Kontext	197
14.1	Wurzeln	197
14.1.1	Das Missverständnis mit dem Wasserfall	197
14.1.2	Agile ändert die Spielregeln	198
14.1.3	Agile stammt von Lean ab	198
14.1.4	Das Agile Manifest	200
14.1.5	Das Manifest der Interdependenz	202
14.2	Lean Thinking und agile Software	203
14.2.1	Das Haus von Lean	203
14.2.2	Lean Thinking	205
14.3	Agile Skalierungsframeworks im Vergleich	207
14.3.1	LeSS von Craig Larman und Bas Vodde	208
14.3.2	Scrum@Scale von Jeff Sutherland	210
14.3.3	Enterprise Scrum von Mike Beedle	212
14.3.4	Nexus von Ken Schwaber	214
14.3.5	DAD von Scott Ambler	216
14.3.6	Verschiedene Zielsetzungen der Frameworks	218
14.3.7	Starterkits – über Sinn und Unsinn agiler Frameworks	219
14.4	Fortgeschrittene agile Implementierungen	220
14.5	Die Grenzen von SAFe	221
14.5.1	Missverständnisse von SAFe	221
14.5.2	Wer braucht SAFe?	222
14.5.3	Feedbackschleifen schließen	224
14.6	Kultur verspeist Prozess zum Frühstück	224
	Literaturverzeichnis	225
	Index	227