



Sascha Block

Large-Scale Agile Frameworks

Agile Frameworks, agile Infrastruktur
und pragmatische Lösungen zur digitalen
Transformation

EBOOK INSIDE

 Springer Vieweg

Large-Scale Agile Frameworks

Sascha Block

Large-Scale Agile Frameworks

Agile Frameworks, agile Infrastruktur
und pragmatische Lösungen zur digitalen
Transformation



Springer Vieweg

Sascha Block
Hamburg, Deutschland

ISBN 978-3-662-62047-2 ISBN 978-3-662-62048-9 (eBook)
<https://doi.org/10.1007/978-3-662-62048-9>

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Springer Vieweg

© Springer-Verlag GmbH Deutschland, ein Teil von Springer Nature 2023

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von allgemein beschreibenden Bezeichnungen, Marken, Unternehmensnamen etc. in diesem Werk bedeutet nicht, dass diese frei durch jedermann benutzt werden dürfen. Die Berechtigung zur Benutzung unterliegt, auch ohne gesonderten Hinweis hierzu, den Regeln des Markenrechts. Die Rechte des jeweiligen Zeicheninhabers sind zu beachten.

Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag, noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen. Der Verlag bleibt im Hinblick auf geografische Zuordnungen und Gebietsbezeichnungen in veröffentlichten Karten und Institutionsadressen neutral.

Planung: David Imgrund

Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer-Verlag GmbH, DE und ist ein Teil von Springer Nature.

Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany

Vorwort

Mein Entschluss, dieses Buch zu schreiben, reifte im Zusammenhang mit meiner Masterarbeit an der Universität Hamburg, die sich mit der Eignung der Large-Scale Agile Frameworks als Organisationsmodell für Software-Hersteller auseinandersetzt. Schon der Entschluss, noch in „fortgeschrittenem Alter“ einen Masterstudiengang zu absolvieren, war maßgeblich davon geprägt, dass ich in meiner Berufspraxis in vielen Belangen agile Organisationsformen vermisst habe. Dabei waren es nie die einzelnen Teams oder Kollegen, die solcher Methodik entgegenstanden – sogar ganz im Gegenteil!

Regelmäßig war festzustellen, dass Best Practices und ausgereifte agile Organisationsmodelle fehlen oder schlicht unbekannt waren. Ohnehin war ich fest entschlossen, mich genau mit diesem Themenkomplex der Large-Scale Agile Frameworks auseinanderzusetzen. Bis heute nehme ich wahr, wie schwer es Unternehmen fällt, agile Arbeitsweisen ganzheitlich in ihrer Organisation zu verankern.

Ich habe dieses Buch entwickelt, um dabei zu unterstützen, Agilität als festen Bestandteil unserer modernen Arbeitskultur in Unternehmen und Organisationen jeder Art zu etablieren. Losgelöst davon, in welchem Reifegrad Sie sich oder Ihre Kolleg*innen und Mitarbeiter*innen aktuell befinden, soll dieses Buch als Leitfaden dabei unterstützen, agiles Mindset und agile Arbeitsweisen in Ihrem Berufsalltag zu verankern. Unabhängig davon, ob Sie als Führungskraft oder Teammitglied mehr über agile Methodik erfahren wollen, soll Ihnen dieses Werk als Kompass dienen und navigiert Sie entlang an den zurzeit brandaktuellen Themen wie Cloud-Technologien, DevOps und IT-Security. Ein weiteres Ziel ist es, das Bewusstsein für IT-Sicherheit zu schärfen und praktikable Strategien und Lösungen zur Umsetzung aufzuzeigen. Es ist mir ein Anliegen, dabei zu helfen, Ihre Kenntnisse über die Grundlagen von Agilität hinaus zu erweitern. Das vorliegende Fachbuch wurde auf der Grundlage jahrelanger Erfahrung in unzähligen IT-Projekten und vor dem Hintergrund verschiedenster Unternehmensumgebungen erstellt. Es gibt viele Bücher, die in Agilität und die dazu gehörigen Grundlagen einführen. Mein Plan war es, ein Buch zu schreiben, das darüber hinausgeht. Dieses Buch soll Sie und Ihre Organisation effektiv dabei unterstützen, agil durchzustarten und eine neuartige Organisationsform zu etablieren, damit Ihnen gemeinsam mit Ihren Kolleg*innen und Teams – und vor allem mit Spaß – eine erfolgreiche digitale Transformation gelingt!

Der Zeitraum, in dem dieses Buch entstanden und gewachsen ist, wurde von der Corona-Pandemie geprägt; nicht nur die immense Bedeutung, sondern vor allem auch die Dringlichkeit agiler Methodik wurde noch einmal verschärft. Als IT-Architekt im Kontext der Telematikinfrastruktur sind die Ergebnisse meiner täglichen Arbeit stark von vielen Einzelpersonen und Teams abhängig, wobei externe Organisationen, rechtliche Rahmenbedingungen und neue Themenfelder – in und abseits von Software-Architekturen – eine maßgebliche Rolle spielen.

In diesem Zusammenhang durfte ich in dieser prägenden Zeit die Einführung zahlreicher Collaboration Tools mitgestalten und erleben, wie sich diese positiv auswirken; zudem konnte ich lernen wie wichtig das Feedback des Einzelnen und der Gemeinschaft ist und wie sehr ein agiles Mindset geeignet ist, Herausforderungen im Team zu meistern.

Weil ich durch meine Arbeit als IT-Architekt weiß, wie wichtig visuelle Darstellungen für die Entwicklung eines gemeinsamen Verständnisses – gerade auch über einzelne Teams hinaus – sind und wie wirksam grafische Artefakte für eine erfolgreiche Gestaltung akzeptierter Lösungen sind, habe ich dieses Buch um zahlreiche Darstellungen erweitert. Bekanntlich sagt ein Bild oft mehr als 1000 Worte ...

Dieses Buch ist das Ergebnis unzähliger Iterationen, beeinflusst durch immer neue Ideen, sei es während meiner täglichen Arbeit als IT-Architekt und im Dialog mit Kolleg*innen oder mit Freund*innen und anderen IT-Spezialist*innen auf zahlreichen Onlineveranstaltungen und Meetups.

Aufbau des Buchs

Das Buch strukturiert das Problem im Kontext agiler Software-Entwicklung und richtet dabei den Fokus auf Large-Scale Agile Frameworks, die den besonderen Anforderungen von Unternehmen gerecht werden, die mit einer Vielzahl agiler Teams in diversen, oft parallel verlaufenden, Digitalisierungsprojekten konfrontiert sind.

Teil I – Digitale Transformation mit Large-Scale Agile Frameworks Praxisnahe Methodik basierend auf realer Projekterfahrung

Kap. 2 schildert die Problemstellungen mit denen Projektteilnehmer und Stakeholder bei der digitalen Transformation konfrontiert sind. Die **agile Priorisierung** ist dabei regelmäßig eine Herausforderung für alle Beteiligten.

Über klar definierte **Ziele der digitaler Transformation** und dem Wechsel in agile Arbeitsweisen wird die Bedeutung agiler Prozesse und der Large-Scale Agile Frameworks dargestellt.

Agile Konzepte und grundlegende Begriffe werden erläutert und im Kontext von Multi-Projekt-Management und Portfolio Management beleuchtet.

Software-Lizenzprodukte und **Individualsoftware** sind von einer **Software-Plattform**, der **Software-Produktfamilie** und einer **Software-Produktlinie** abzugrenzen, weil hier anforderungsseitig unterschiedliche Rahmenbedingungen zu berücksichtigen sind. Der **Produktlebenszyklus von Software** sowie **Software Releases**, das

Release-Management, die **Software-Architektur** und **Wissensmanagement** sowie DevOps haben wesentlichen Einfluss auf die agile Software-Entwicklung.

Agile Konzepte und das **agile Projektmodell** werden im Kontext von **Scaled-Agile** und **Large-Scale Agile**, Agile Teams, deren Rollen, Aufgaben und Prozesse dargestellt.

Kap. 3 führt in das Konzept der Large-Scale Agile Frameworks ein und stellt die relevantesten und in der Praxis bewährtesten Modelle vor, zeigt deren Unterschiede und Grenzen auf.

Kap. 4 bietet eine praxisorientierte Unterstützung bei der Einführung eines Large-Scale Agile Frameworks in einer Organisation. Mit der **Methode des Action Design Research** steht Ihnen ein moderner Ansatz zur praxisorientierten Problemlösung in Organisationen zur Verfügung.

Mit einer **faktenbasierten Zusammenfassung der relevantesten Einflussfaktoren des Cloud-Trends und der Virtualisierung** werden nicht nur hochaktuelle Technologiethemen berücksichtigt, sondern auch deren Auswirkungen in Bezug auf die Erfordernisse agiler Organisationsformen berücksichtigt.

Software-Architektur und **IT-Security** spielen eine zentrale Rolle für alle Aktivitäten der Digitalisierung. Vorgestellt werden nicht nur die Paradigmen der Ansätze **IT-Security-by-Design** und der **Zero-Trust-Strategie**, sondern auch effektives Secret-Management und der erweiterte Schutzbedarf virtueller Container-Umgebungen. Somit ergibt sich eine ganzheitliche Betrachtung.

Hierbei dürfen die Aspekte **von Public-Key-Infrastrukturen, Microservice-Architekturen, REST-APIs** und **RESTful-Design**, dem **Architektur-Pattern der Service Meshs** und darauf ausgerichtete Qualitätsmerkmale nicht außer Acht gelassen werden.

Auf Grundlage der **OWASP-Guidelines** lässt sich IT-Sicherheit nachweislich durch iteratives, begleitendes Pentesting verbessern. Penetration-Tests hingegen dokumentieren untersuchte Eigenschaften zu einem bestimmten Zeitpunkt unter definierten Rahmenbedingungen mitsamt vereinbarter Testkriterien und attestieren somit IT-Sicherheit in Bezug auf einen definierten Testumfang. Kombiniert ergänzen sich beide Maßnahmen wechselseitig als wirksame Mittel um Software zu härten, also die IT-Sicherheit zu erhöhen.

Mit der Methodik des **Threat Modeling** wird ein wirksamer Mechanismus erläutert, um Bedrohungen vorzeitig zu identifizieren und diese Sicherheitsrisiken dann systematisch zu eliminieren.

Sie erfahren nachvollziehbar, warum frühestmögliches, agiles Deployment und automatisierte Tests geeignete Mittel sind um robuste und **hinsichtlich der IT-Sicherheit gehärtete Software-Lösungen** zu entwickeln.

In diesem Kontext wird die relative junge Disziplin **DevSecOps** vorgestellt und mit den Arbeitsmitteln von **Code Reviews** und des **Pair Programmings** werden praxisorientierte Methoden zur Software-Entwicklung dargestellt.

Die oft außer Acht gelassene Anforderung zum **Logging und Monitoring** hat in diesem Abschnitt den Fokus auf IT-Sicherheit und zu beachtende Datenschutzaspekte.

Zur Bewältigung dieser Herausforderungen werden die erforderlichen **agilen Teams, Rollen, Aufgaben und Prozesse** skizziert, die ein agiles Organisationsmodell erfordert. Mit der Vorstellung der jeweiligen Spezialisierungen wird eine verständliche und leicht nachvollziehbare Einordnung der einzelnen Fachdisziplinen gegeben. Gleichzeitig kann dieser Abschnitt Organisationen wirkungsvoll dabei unterstützen, eigene Teams und Rollenprofile zu schärfen oder zu ergänzen.

Im Zusammenhang von **Design Thinking** und **Prototyping** wird das erforderliche Grundlagenwissen zur prototypischen Vorgehensweise und Rapid Prototyping und das Prototypenphasenmodell vorgestellt.

Kap. 5: Agiles Priorisierungsmodell für Software-Hersteller reflektiert die Unternehmen in der Rolle eines Software-Herstellers anhand von drei bekannten Unternehmen und verdeutlicht somit die Tragweite und Bedeutung von Software-Entwicklung für Organisationen. In **Abschn. 5.2** werden die **Agilen Teams und Rollen die für Organisationen in der Rolle eines Software-Herstellers** typisch sind und in **Abschn. 5.3** die entsprechenden **Prozesse und Aktivitäten** präsentiert.

Auf Grundlage der Anforderungen und Ergebnisse der vorhergehenden Kapitel wird dann das *entwickelte agile Modell zur übergreifenden Priorisierung innerhalb unterschiedlicher Unternehmenskonstellationen* vorgestellt und begründet, warum dieses Modell geeignet ist, um die Anforderungspriorisierung und agile Software-Entwicklung für eine Vielzahl unterschiedlicher Software-Projekte in Einklang zu bringen.

Um Hürden wirksam zu begegnen, stellt „**Kap. 6 – Herausforderungen bei der Etablierung eines Large-Scale Agile Frameworks im Unternehmen**“ entsprechende Maßnahmen vor, damit Sie auch in Ihrer Organisation die regulären Herausforderungen meistern, die einer effektiven Anwendung agiler Methodik entgegenstehen.

Mit der Etablierung von **Friendly User Tests** erfahren Sie, warum Entwicklungsstufen innerhalb einer Organisation nicht übersprungen werden können, sondern mit **agilen Reifegraden** genommen werden müssen. Außerdem werden Sie darüber lesen, wie **neuartige Entwicklungs- und Testumgebungen mit abstrahierter Hardware-Ebene** Ihnen neben zuverlässigen Tests auch solide Testergebnisse liefern. Auch die **Bedeutung transparenter Entscheidungsprozesse und eines agilen Anforderungsmanagements** werden dargestellt.

Abschn. 6.2 verdeutlicht, **wie und warum gutes Digital Leadership funktioniert**. Von der Vision zur Strategie wird das Konzept einer **Produkt-/Service-Roadmap** vorgestellt. Überdies werden die Rollen des **Chief Information Officer (CIO)** und die neue Führungsrolle des **Chief Digital Officer (CDO)** detailliert erklärt.

Ebenso sind die wesentlichen Punkte des **Change-Managements** erläutert, die gerade während der Einführung neuer Organisationsstrukturen relevant sind. Hierzu zählen insbesondere die für die Software-Entwicklung wesentlichen Verfahren des **Change-Request- und Release-Managements**, die Etablierung agiler Fähigkeiten über eine Agile Academy und die Etablierung agiler Werte.

Die Ausführungen zum **Notfallmanagement** schließen den ersten Teil mit dem Weitblick auf agile und angemessene Reaktionen in Notfallsituationen ab.

Teil II – Agile Infrastruktur

Kap. 7 – Agile Tools: Werkzeugkasten für Product Owner & Agile Teams

Der zweite Teil des Buches widmet sich den agilen Tools und dient als Werkzeugkasten für Product Owner & Agile Teams.

Neben dem **Agilen Manifest** werden **agile Ziele**, die agilen Konzepte der **Personas**, **User Stories**, **Epics**, **Tasks** und **Sub-Tasks** und des **Backlog** erläutert.

Collaboration Tools sind die agilen Werkzeuge mit denen Stakeholder und die Mitglieder der unzähligen agilen Teams miteinander kooperieren und Informationen austauschen. Sie erfahren die Bedeutung und Funktionsweise der **Git-Repositories** sowie **OpenAPI** als ein **Werkzeug zum API-Design**.

Damit **agile Architekturen das Fundament softwarebasierter Digitalisierung** bilden, lernen Sie welche wichtigen Einflussfaktoren für eine agile Infrastruktur gelten.

Mit Abschn. 7.5 erhalten Sie einen **Leitfaden zur Erstellung einer pragmatischen Software-Architektur-Dokumentation** und erfahren, wie Sie **aussagekräftige visuelle Artefakte für Ihre Software-Architektur** erstellen und welche **Standards, Kriterien und Normen** sinnvoll sind.

Beantwortet wird auch: Wer sind die **Adressaten** und was sind die **Themenfelder der Software-Architektur Dokumentation**? Wie formulieren Sie aussagekräftige Fragestellungen für die wesentlichen Software-Architektur-Entscheidungen? Auch die Relevanz der **technischen Schulden** ist dargestellt. Das **Arc42-Template** liefert Ihnen abschließend ein **direkt nutzbares Handwerkszeug für die Software-Architektur-Dokumentation**. Und die Norm der **ISO/IEC 25010** definiert die **Qualität von Software**.

In Abschn. 7.6 werden **DevOps-Methoden** und **DevOps-Tools** mit der Überleitung zu **BizDevOps** vorgestellt.

Der Abschn. 7.7 stellt **Content** und die **Ziele einer Content-Strategie** vor und führt aus, welche Anforderungen an wertvolle Inhalte zu stellen sind und wie sich die **Wirkksamkeit von Content messen** lässt. Der **Darstellung und Visualisierung von Content** widmen wir uns ebenfalls, und stellen die besonderen **Content-Formate der Infografiken und Dialogbilder** vor.

Im Abschn. 7.8 werden Möglichkeiten zur **Umsetzung eines Monitorings und Controllings** auf der Basis von **Key-Performance-Indikatoren (KPI)** erklärt und mit welchen Strategien und Werkzeugen sich ein solches Reporting umsetzen lässt.

Abschn. 7.9 stellt **Methoden und Werkzeuge zur agilen Priorisierung** vor, die für Product Owner und die agilen Teams im Umfeld der Software-Entwicklung eine echte Erleichterung bieten, um Anforderungen zu priorisieren und bestehende Abhängigkeiten zwischen Anforderungen erkennbar werden lassen. Sowohl **Timeboxing** als auch das **Feature-Driven Development** zählen hierbei zu den beliebtesten und bewährten Verfahrensweisen. Mit der Vorstellung von **Liquid Democracy** lernen Sie ein agiles Abstimmungsverfahren kennen und erfahren wie sie Transparenz innerhalb einer Organisation gemeinsam leben können.

Den Abschluss bildet das Kap. 8, das sich der Datenqualität und damit dem Lebenselixier der Digitalisierung widmet. Von den Grundlagen und Voraussetzungen für eine

exzellente Datenqualität, geht es zu Techniken der Datenbereinigung bis hin zu wirksamen Kontrollmechanismen zur Optimierung der Datenqualität.

Danksagung

Mein persönlicher Dank gilt insbesondere der mentalen Unterstützung meiner Frau Martina und der Geduld unserer Tochter Coco. Mein besonderer Dank richtet sich an meine, zur Entstehungsphase dieses Buchs, nach langer Krankheit verstorbene Mutter, die mich stets von ganzem Herzen bei meinen Zielen unterstützt und bestärkt hat. Aileen und Jean Paul soll dieses Buch auf Ihrem Weg begleiten und zeigen, dass jedes große Ziel – auch in kleinen Schritten – gelingt.

Ihre Fragen & Ihr persönliches Feedback ist gefragt!

Feedback von unseren Lesern ist immer herzlich willkommen!

Weil mir persönlich ein agiles Mindset, Transparenz und der offene Dialog sehr am Herzen liegen, bitte ich Sie als Leser unbedingt jegliches Feedback an mich als Autor dieses Werks zu richten.

Wenn Sie Fragen zu irgendeinem Aspekt dieses Buches haben, senden Sie mir bitte eine eMail an info@large-scale-agile-frameworks.com

Errata: Obwohl ich jede mir mögliche Sorgfalt darauf verwendet habe, die Richtigkeit meiner Inhalte zu gewährleisten, können Fehler passieren. Wenn Sie einen Fehler in diesem Buch gefunden haben, sind wir Ihnen dankbar, wenn Sie uns diesen melden würden.

Selbstverständlich freuen wir uns ebenso über Ihr allgemeines Feedback: Senden Sie uns eine eMail und erwähnen Sie den Titel des Buches im Betreff Ihrer Nachricht.

Aktuelle Informationen finden Sie ebenfalls auf der Website www.large-scale-agile-frameworks.com

Teilen Sie Ihre Gedanken in Form einer Buch-Rezension !

Wenn Sie Large-Scale Agile Frameworks – Agile Frameworks, agile Infrastruktur und pragmatische Lösungen zur digitalen Transformation gelesen haben, würden wir gerne Ihre Meinung dazu hören!

Ihre Rezension ist wichtig für uns und die Tech-Community und hilft uns, sicherzustellen, dass wir exzellente Qualitätsinhalte liefern.

Hamburg, Deutschland

Sascha Block

Inhaltsverzeichnis

Teil I Digitale Transformation mit Large-Scale Agile Frameworks

- 1 Einleitung** 3
 - Literatur..... 8

- 2 Digitale Transformation & Agile Priorisierung** 9
 - 2.1 Agile Modelle zur Organisation digitaler Transformation..... 9
 - 2.1.1 Digitale Transformation – Herausforderung mit vielen Chancen .. 10
 - 2.1.2 Welche Faktoren beeinflussen digitale Transformation?..... 12
 - 2.1.3 Ziele digitaler Transformation 14
 - 2.1.4 Digitale Transformation und Large-Scale Agile Frameworks 16
 - 2.1.5 Die Bedeutung agiler Prozesse und Large-Scale Agile Frameworks ... 18
 - 2.2 Wie bewährt sind agile Vorgehensmodelle?..... 20
 - 2.3 Agile Konzepte und grundlegende Begriffe..... 21
 - 2.3.1 Multi-Projekt-Management 22
 - 2.3.2 Portfolio-Management..... 24
 - 2.3.3 Software-Lizenzprodukte und Individualsoftware..... 25
 - 2.3.4 Software-Plattform, Software-Produktfamilie und Software-Produktlinie 28
 - 2.3.5 Produkt-Lebenszyklus von Software 30
 - 2.3.6 Software Releases und Release Management 32
 - 2.3.7 Software-Architektur und Wissensmanagement 34
 - 2.3.8 Requirement Management/Priorisierung..... 35
 - 2.3.9 DevOps und DevOps-Modell..... 37
 - 2.3.10 Agilität und Agiles Projekt-Modell 38
 - 2.3.11 Scaled Agile/Large-Scale Agile Development..... 44
 - Literatur..... 45

- 3 Large-Scale Agile Frameworks** 49
 - 3.1 Bewertungskriterien für Large-Scale Agile Frameworks..... 49
 - 3.2 Ausgewählte Scaled Agile Frameworks..... 51

3.3	Domänenorientiertes Modell/Domain-Driven Design.	52
3.4	Spotify Engineering Model	54
3.5	Scaled Agile Framework (SAFe).	60
3.6	Vergleich der drei ausgewählten Large-Scale Agile Frameworks	63
	Literatur.	65
4	Wie Sie ein Large-Scale Agile Framework anpassen und in Ihrer Organisation einführen.	67
4.1	Action Design Research.	68
4.1.1	Individuelle Anpassungen für ein Large-Scale Agile Framework definieren	72
4.1.2	Ist-Analyse im Transformationsprozess mittels Evaluation	73
4.1.3	Befragungsdesign.	76
4.1.4	Auswertung und Ergebnisse der formativen Evaluation	78
4.1.5	Summative Evaluation.	78
4.1.6	Experteninterview	79
4.2	Einflussfaktoren des Cloud-Trends und der Virtualisierung im Fokus eines agilen Frameworks berücksichtigen	79
4.2.1	Cloud Computing.	80
4.2.2	Cloud-Eigenschaften	81
4.2.3	Cloud-Service-Modelle	81
4.2.4	Cloud-Modelle	82
4.2.5	Virtualisierung & Containerisierung	85
4.2.6	Relevante Gremien zur Etablierung von Internet- und Cloud-Standards	89
4.3	Software-Architektur & IT-Security als integraler Bestandteil eines agilen Frameworks.	92
4.3.1	IT-Security-by-Design: Software-Architektur & IT-Security	93
4.3.2	Zero-Trust-Strategie.	94
4.3.3	Schutzprinzipien und deren technische Umsetzung auf Basis von Zero Trust	96
4.3.4	Secret-Management	100
4.3.5	Erweiterter Schutzbedarf für virtuelle Container-Umgebungen	100
4.3.6	Key-Management & kryptografische Schutzmaßnahmen	101
4.3.7	Public-Key-Infrastrukturen	104
4.3.8	Microservice-Architekturen.	105
4.3.9	APIs, Ressourcen und dynamische IP-Adressen in Cloud-Netzwerken	108
4.3.10	APIs und REST	109
4.3.11	Qualitätsmerkmale von Microservices und Web-APIs	109
4.3.12	RESTful API	113

4.3.13	Fazit und Bezug von APIs im Hinblick auf Large-Scale Agile Frameworks	115
4.3.14	Service Mesh & Agile Microservice-Architekturen	116
4.3.15	IT-Security auf Grundlage der OWASP Guidelines verbessern.	118
4.3.16	Penetrationstests/Pentesting.	118
4.3.17	Empfehlungen um IT-Security als festen Bestandteil in einem agilen Framework zu integrieren	122
4.3.18	Threat Modeling.	130
4.3.19	Frühestmöglich zum agilen Deployment & automatisierten Tests	134
4.3.20	DevSecOps.	137
4.3.21	Code Reviews	138
4.3.22	Pair Programming	139
4.3.23	Logging & Monitoring.	141
4.4	Agile Teams, Rollen, Aufgaben und Prozesse	143
4.4.1	Agile Software-Entwicklungs-Teams	144
4.4.2	IT-Security-Teams	146
4.4.3	Umsetzung rechtlicher Rahmenbedingen und Datenschutz	147
4.4.4	Software-Produktmanagement/Service-Management	148
4.4.5	UX-Teams: Frontend Design, Usability und User Experience	148
4.4.6	Qualitätssicherung und Testverfahren	149
4.4.7	Technische Redaktion	150
4.4.8	Infrastruktur-Teams	151
4.4.9	DevOps-Teams.	152
4.4.10	Sales – Der Vertrieb von Produkten und Dienstleistungen Ihrer Organisation.	154
4.4.11	Forschung, Innovation und PreSales	155
4.4.12	Systemintegration	155
4.4.13	Support-Teams	157
4.5	Mit Design Thinking & Prototyping durchstarten.	158
4.5.1	Prototyping und Rapid Prototyping	160
4.5.2	Prototyping-Phasenmodell.	162
4.6	Übergeordnete agile Prozessphasen auf eine prototypische Vorgehensweise ausrichten	163
4.6.1	Phase 1: Anforderungserhebung und Ideenfindung	164
4.6.2	Phase 2: Rapid Prototyping und Beratungsprozess	167
4.6.3	Phase 3: Testmanagement und summativ Evaluation	173
4.6.4	Anmerkungen und Empfehlungen zum prototypischen Vorgehensmodell	175
	Literatur.	176
5	Agiles Priorisierungsmodell für Software-Hersteller	179
5.1	Inwieweit agiert Ihre Organisation in der Rolle eines Software-Herstellers?	180

5.1.1	Beispiel 1: Ist LEGO ein Software-Hersteller?	182
5.1.2	Beispiel 2: Flaschenpost.de – innovativer, App-getriebener Getränkesservice	185
5.1.3	Beispiel 3: Moia – digitaler Shuttle Service	187
5.2	Agile Teams und Rollen bei Software-Herstellern	189
5.3	Prozesse und Aktivitäten	193
5.4	Fazit	201
5.5	Schlussfolgerungen	201
5.6	Ausblick	205
5.6.1	Implikationen für die Praxis	205
	Literatur	207
6	Herausforderungen bei der Etablierung eines Large-Scale Agile Frameworks im Unternehmen	209
6.1	Start-up-Wind in etablierte Organisationen bringen	209
6.1.1	Agile Software-Service-Entwicklung	211
6.1.2	Friendly User Tests (FUT)	211
6.1.3	Unterschiedliche Agilitätsgrade der Teams	212
6.1.4	Neuartige Entwicklungs- und Testumgebungen mit abstrahierter Hardwareebene etablieren	212
6.1.5	Transparente Entscheidungsprozesse und agiles Anforderungsmanagement	213
6.2	Digital Leadership	215
6.2.1	Vision, Strategie und Product/Service Roadmap	217
6.2.2	Chief Information Officer	218
6.2.3	Chief Digital Officer	220
6.3	Change-Management – Digital Leadership im Management	223
6.3.1	Perspektiven des Change-Management	223
6.3.2	Change-Request-/Release-Management	224
6.3.3	Agile Academy	225
6.3.4	Agile Werte etablieren	225
6.4	Notfallmanagement: Kann Ihre Organisation agil in Notfallsituationen reagieren?	226
6.4.1	Wie können Sie Notfällen proaktiv begegnen?	228
6.4.2	Wie gehen wir ein Notfallkonzept organisatorisch an?	229
	Literatur	231

Teil II Agile Infrastruktur

7	Agile Tools: Werkzeugkasten für Produkt Owner & Agile Teams	235
7.1	Agiles Mindset & die 12 Prinzipien des agilen Manifests	237
7.2	Agile Ziele	242

7.2.1	Personas	243
7.2.2	User Story	244
7.2.3	Epic	248
7.2.4	Task & Sub-Task	250
7.2.5	Backlog	251
7.3	Collaboration Tools	252
7.3.1	Confluence	254
7.3.2	Jira	256
7.3.3	Git-Repositories, GitHub und GitLab	256
7.3.4	OpenAPI – Werkzeuge zum API-Design	261
7.3.5	Messenger und Chat-Systeme	265
7.4	Agile Architekturen – Fundament softwarebasierter Digitalisierung	267
7.4.1	Einflussfaktoren für agile IT-Architekturen	268
7.5	Pragmatische Software-Architektur-Dokumentation	269
7.5.1	Wie Sie visuelle Software-Architektur-Artefakte erstellen	270
7.5.2	Standards, Kriterien und Normen für Software- Architektur-Artefakte	271
7.5.3	Adressaten und Themenfelder der Software- Architektur-Dokumentation	273
7.5.4	Fragestellungen für Software-Architektur- Entscheidungen formulieren	274
7.5.5	Technische Schulden	277
7.5.6	Arc42-Template zur Software-Architektur-Dokumentation	278
7.5.7	ISO/IEC 25010 – Qualität von Software	280
7.6	DevOps-Methoden und DevOps-Tools	282
7.6.1	Das DevOps-Periodensystem	282
7.6.2	DevOps ist mehr als = Software Engineering + IT-Management	282
7.6.3	BizDevOps als Konsequenz für agile Unternehmen	285
7.7	Content	288
7.7.1	Ziele einer Content-Strategie	289
7.7.2	Anforderungen an den Content	290
7.7.3	Content Controlling	290
7.7.4	Content-Prozess und Koordination	291
7.7.5	Content-Leitfaden	291
7.7.6	Design-System	291
7.7.7	Infografiken	294
7.7.8	Dialogbilder	296
7.8	Monitoring & Controlling	298
7.8.1	Key-Performance-Indikatoren (KPI)	299
7.8.2	Monitoring	300
7.8.3	Strategien und Lösungswege zu digitaler Transformation	300
7.8.4	Die optimale Architektur für die digitale Organisation	302

7.9	Methoden & Werkzeuge zur agilen Priorisierung	302
7.9.1	Agile Anforderungspriorisierung mit dem Feature Graph.	303
7.9.2	Agile Priorisierung mit der Einpunktabfrage.	304
7.9.3	Agile Priorisierung mit der Mehrpunktabfrage	304
7.9.4	Agile Priorisierung mit Liquid Democracy	306
7.9.5	Timeboxing	306
7.9.6	Feature-Driven Development.	307
	Literatur.	309
8	Datenqualität – Lebenselixier der Digitalisierung.	311
8.1	Kundenindividuelle Produkte und Dienstleistungen auf Basis von Smart Data.	313
8.2	Datenqualität: Grundlagen und Voraussetzungen	314
8.2.1	Datenabhängigkeiten	315
8.2.2	Datenreparatur	318
8.2.3	Dateneduplikation	320
8.2.4	Informationsvollständigkeit	322
8.2.5	Datenaktualität	323
8.2.6	Datenpräzision	323
8.3	Techniken zur Datenbereinigung	323
8.3.1	Qualitätsregeln für Daten entdecken	324
8.3.2	Fehlererkennung für Daten	324
8.3.3	Datenreparatur	324
8.4	Kontrollmechanismen zur Optimierung der Datenqualität sind unverzichtbar	326
8.4.1	Mit Machine Learning zu Smart Data	326
8.4.2	Mit Smart Data Innovationen aus Daten gewinnen	327
8.4.3	Semantische Datenanalyse.	328
8.4.4	Framework für Data-Driven Design	329
8.4.5	Fazit	330
	Literatur.	331

Abkürzungsverzeichnis

ADR	Action Design Research – Bitte ergänzen: API - Application Programming Interface
ART	Agile Release Train
AWS	Amazon Web Services
B2B	Business to Business
B2C	Business to Customer
BizDevOps	Business, Development & Operations
BL	Bereichsleitung
BSI	Bundesamt für Sicherheit in der Informationstechnik – bitte ergänzen: CI – Configuration Item
CR	Change Request
CWA	Closed World Assumption – bitte ergänzen: DDD – Domain Driven Design
DEV	Development (Entwicklung)
DevOps	Development & Operations
DoS	Denial of Service
DSVGO	Datenschutzgrundverordnung
EAM	Enterprise Architecture Management
FAQ	Frequently Asked Questions
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastruktur als Dienstleistung
IAM	Identity Access Management
IDLE	Infrastruktur Dienstleistungen
IoT	Internet of Things
IT	Informationstechnologie
KPI	Key-Performance-Indikatoren
LDAP	Lightweight Directory Access Protocol
MPM	Multi-Projekt-Management
MVC	Model View Controller
MVP	Minimum Viable Product
NIST	National Institute of Standards and Technology

OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
OPS	Operations (IT-Betrieb)
OWA	Open World Assumption
PaaS	Plattform-as-a-Service
PI	Programminkremente
PJM	Projektmanager
PM	Produktmanagement
PO	Produkt Owner
RTE	Release Train Engineer
SA	Software-Architekt*in
SaaS	Software-as-a-Service
SAFe	Scaled Agile Framework
TLS	Transport Layer Security – bitte ergänzen: TOM – Technisch organisatorische Maßnahmen
UML	Unified Modeling Language
UX	User Experience – bitte ergänzen: VCS – Versioning Control System
Ver	Vertrieb
VMM	Virtual Machine Monitor
XP	Extreme Programming
XML	Extensible Markup Language

Abbildungsverzeichnis

Abb. 1.1	Änderungskosten im Projektverlauf. (Quelle: Sascha Block – Eigene Darstellung)	6
Abb. 1.2	Iteratives Projektmodell. (Quelle: Sascha Block – Eigene Darstellung).	7
Abb. 2.1	Domänen und Rollen in einem Unternehmen. (Quelle: Eigene Darstellung – Sascha Block)	18
Abb. 2.2	Multi-Projekt-Management. (Quelle: Eigene Darstellung in Anlehnung an Steinle et al.).	22
Abb. 2.3	Synchronisation von Strategie & Zielen. (Quelle: Eigene Darstellung in Anlehnung an Lang et al.)	23
Abb. 2.4	Entwicklung von Absatz, Rentabilität und Liquidität über den Produkt-Lebenszyklus. (Quelle: Sascha Block)	31
Abb. 2.5	Arc42-Workflow in Anlehnung an Hruschka/Starke	36
Abb. 2.6	DevOps-Modell. (Quelle: Sascha Block)	38
Abb. 2.7	Werte agiler Prozesse. (Quelle: Sascha Block)	43
Abb. 3.1	Model-View-Controller-Konzept. (Quelle: Sascha Block)	54
Abb. 3.2	Organisationsmodell Spotify-Engineering-Modell Themenfelder. (Quelle: Sascha Block in Anlehnung an Kniberg/Ivarsson: Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds – 10/2012. – Quelle: Sascha Block).	55
Abb. 3.3	Status-Umfragen des Spotify-Engineering-Modell zu definierten Themen. (Quelle: Sascha Block in Anlehnung an Kniberg/Ivarsson: Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds – 10/2012)	57
Abb. 3.4	Organisation von Themenfelder nach dem Spotify Engineering Modell. (Quelle: Sascha Block in Anlehnung an Kniberg/Ivarsson: Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds – 10/2012).	58
Abb. 3.5	Agile Release Train – Fotografie und Bildbearbeitung: Sascha Block . . .	61

Abb. 4.1	Action Design Research.	69
Abb. 4.2	Cloud-Arten.	84
Abb. 4.3	Hypervisor Type-1 und Type-2 in Anlehnung an Bernstein.	86
Abb. 4.4	Abbildung 18 Docker Container versus Virtualisierung – Sascha Block.	88
Abb. 4.5	Zero-Trust-Strategie – Darstellung in Anlehnung an Mehraj/Banday.	95
Abb. 4.6	Lebenszyklus kryptografischer Schutzmaßnahmen – Sascha Block – Eigene Darstellung in Anlehnung an NIST Klassifikation.	102
Abb. 4.7	Qualitätsmerkmale zur Interoperabilität und deren Einfluss auf Wiederverwendbarkeit von Software-Artefakten – Sascha Block.	110
Abb. 4.8	Hierarchische-Ebenen-REST-API.	114
Abb. 4.9	Software-Bibliothek als Achillesferse einer digitalen Infrastruktur. (Quelle: Sascha Block).	123
Abb. 4.10	Software-Architektur Dokumentation mit dem Collaboration Tool Atlassian Confluence. (Quelle: Sascha Block).	124
Abb. 4.11	Code-Dokumentation mit dem Collaboration Tool Atlassian Confluence. (Quelle: Sascha Block).	125
Abb. 4.12	Die sechs Hauptaufgaben von Software-Architekt*innen in Anlehnung an Hruschka/Starke.	146
Abb. 4.13	Iteratives Prototyping-Modell.	160
Abb. 4.14	Prototyping-Phasenmodell nach Pomberger/Pree, eigene Darstellung Sascha Block.	163
Abb. 4.15	App POP – Prototyping on Paper.	168
Abb. 4.16	Digitaler Pen+Paper Prototype. (Quelle: Sascha Block).	169
Abb. 4.17	Entwicklungsumgebung einer iOS-App in Xcode. (Quelle: Sascha Block).	170
Abb. 4.18	Schaubild MVP – Anforderungen an ein Minimum Viable Product. (Quelle: Sascha Block – Eigene Darstellung in Anlehnung an Olsen) ...	171
Abb. 5.1	Screenshot des LEGO-Onlineshops als Teil der LEGO-Website.	182
Abb. 5.2	LEGO-Apps im iOS-Store.	183
Abb. 5.3	Bestellvorgang über die Flaschenpost-App.	186
Abb. 5.4	iOS App des digitalen Shuttle Service MOIA.	187
Abb. 5.5	Agiles Software-Portfolio-Management. (Quelle: Sascha Block).	189
Abb. 5.6	Agile-Software-Produktlinien - Sascha Block.	195
Abb. 6.1	Schematischer Ablauf eines Notfallprozesses.	228
Abb. 7.1	Agile Onion.	237
Abb. 7.2	Persona-Profil.	244
Abb. 7.3	User Stories in Atlassian Jira.	247
Abb. 7.4	Epic.	249

Abb. 7.5	Grundprinzip eines Product Backlog.	252
Abb. 7.6	Product Backlog in Form eines Kanban Boards in Atlassian Jira	257
Abb. 7.7	GitHub Repository hier der OpenSource IAM-Lösung Keycloak. (Quelle des Screenshot: Sascha Block)	258
Abb. 7.8	Komponentenbasierte Gliederung eines OpenAPI-Objekts	262
Abb. 7.9	OpenAPI Dokumentation	265
Abb. 7.10	Discord als Messenger-basiertes Collaboration Tool.	266
Abb. 7.11	Draw.io – Anwendung zur Erstellung von grafischen Artefakten für die Software-Architektur. (Quelle: Sascha Block).	271
Abb. 7.12	Themenfelder und Adressaten von Software-Architektur in Anlehnung an Bass et al. (Quelle: Sascha Block)	275
Abb. 7.13	Kontextabgrenzung für Software-Architektur. (Quelle: Sascha Block)	276
Abb. 7.14	Bausteinsicht Software-Architektur. (Quelle: Sascha Block)	277
Abb. 7.15	Struktur des arc42-Template. (Quelle: Sascha Block in Anlehnung Hruschka / Starke).	279
Abb. 7.16	Dimensionen der Qualität von Software auf der Basis der ISO/IEC 25010 - Sascha Block	281
Abb. 7.17	Periodic Table of DevOps	283
Abb. 7.18	BizDevOps-Modell in Anlehnung an Fitzgerald/Stol. (Quelle: Sascha Block)	286
Abb. 7.19	Design-System der Otto Group mit umfangreicher Pattern Library	293
Abb. 7.20	Dimensionen einer Infografik	295
Abb. 7.21	Dialogbild zum konstruktiven Austausch während einer Mitarbeiterbesprechung. (Quelle: Dialogbild GmbH).	297
Abb. 7.22	Dialogbilder zur aktiven Kundenkommunikation. (Quelle: Dialogbild GmbH).	298
Abb. 7.23	Timeboxing zur Release-Planung mit festen Zeitintervallen.	307
Abb. 7.24	Feature Boxing – Methodik des Feature-Driven Development.	308
Abb. 8.1	Themenfelder rund um „Smart Data“. (Quelle: Sascha Block).	312
Abb. 8.2	Continuous Build Measure Learn Cycle. (Quelle: Sascha Block).	327
Abb. 8.3	Künstliche Intelligenz und Machine Learning am Beispiel der Bildererkennung auf einem Apple iPhone. (Quelle: Sascha Block).	329
Abb. 8.4	Framework für Data-Driven Design. (Quelle: Sascha Block).	330

Tabellenverzeichnis

Tab. 2.1	Klassifizierte Konzepte mit Relevanz für agile Zusammenarbeit.	21
Tab. 3.1	Im Spotify-Modell gültige Aspekte zur Agilität.	57
Tab. 4.1	Teilnehmer einer formativen Evaluation. (Quelle eigene Darstellung – Sascha Block).	75
Tab. 4.2	Arbeitsgruppen der Internet Engineering Task Force (IETF).	92
Tab. 4.3	Qualitätsmerkmale zur Interoperabilität und deren Einfluss auf Wiederverwendbarkeit	111
Tab. 4.4	REST-Definitionen Ressource und Repräsentation	113
Tab. 4.5	Anforderungen zum RESTfull API Design	114
Tab. 4.6	Ebenenmodell von REST APIs.	115
Tab. 4.7	Klassifikation von IT-Security-Schutzzielen	131
Tab. 4.8	Klassifikation von IT-Security-Risikogruppen	133
Tab. 4.9	Phasen und zugehörige Methoden des Design-Thinking-Prozesses. (Quelle: Gerstbach – „Design Tinking im Unternehmen“)	159
Tab. 5.1	Übersichtstabelle Teams und Rollen. (Quelle: Sascha Block, Eigene Darstellung)	190
Tab. 6.1	Kommunikationsbeispiele und deren Bedeutung	217
Tab. 6.2	Definition technisch organisatorischer Maßnahmen (TOMs).	230
Tab. 6.3	Prioritätskennzeichnung für TOMs nach dem Ampel-Prinzip	230
Tab. 7.1	Best Practices zur Backlog-Nutzung	253
Tab. 7.2	Feld-Bezeichnungen und deren Bedeutung im OpenAPI-Kontext.	263
Tab. 7.3	Kriterien und Checkliste für Architekturartefakte	272
Tab. 7.4	Darstellungsformen und Diagrammtypen der Unified Modeling Language (UML)	273
Tab. 7.5	Kreuztabellen als pragmatisches Hilfsmittel für Software-Architekturen	273
Tab. 7.6	Beispieltabelle für einen Feature-Graphen. (Quelle: Sascha Block – Eigene Darstellung).	303

Tab. 7.7	Anwendungsbeispiele für Einpunktabfragen. (Quelle: Eigene Darstellung in Anlehnung an Stach)	304
Tab. 7.8	Leitfragen zur moderierten Priorisierung. (Quelle: Eigene Darstellung in Anlehnung an Stach)	305
Tab. 7.9	Regeln zur Durchführung der Mehrpunktabfrage. (Quelle: Eigene Darstellung in Anlehnung an Stach)	305
Tab. 7.10	Timeboxing – Regeln, Vorteile, Nachteile und Risiken. (Quelle: Eigene Darstellung – Sascha Block).	307
Tab. 7.11	Feature Boxing – Regeln, Vorteile, Nachteile und Risiken. (Quelle: Eigene Darstellung, Sascha Block).	308
Tab. 8.1	D_0 – Datensatz personenbezogener Daten	315
Tab. 8.2	Pattern Tableaus PT_1, PT_2	316
Tab. 8.3	Komplexitätsstufen der Implikationsanalyse [4, Kap. 2.1, Seite 3]	318
Tab. 8.4	Komplexitätsstufen der Reparatur-Checks [4, Kap. 2.1, Seite 3]	319

Teil I

**Digitale Transformation mit
Large-Scale Agile Frameworks**



Dieses Buch stellt praxisnahe Lösungen zur übergreifenden Priorisierung von Anforderungen und Dokumentation am Beispiel einer Organisation in der Rolle oder mit wesentlichen Merkmalen eines Software-Herstellers dar. Dabei wird das Zusammenspiel aktueller Technologiethemen wie des Cloud-Trends oder die organisatorischen Anforderungen im Hinblick auf Microservices reflektiert. Unter der Erfordernis kundenzentrierter und serviceorientierter Produkte und Dienstleistungen sind Organisationen zunehmend damit konfrontiert, ihre IT-Strategie eng an den Bedürfnissen ihrer Kunden auszurichten. Zudem erfordern verschärfte Anforderungen in Richtung IT-Security und im Kontext der digitalen Transformation mittels Cloud-Technologien und Containerisierung eine radikale Neuausrichtung bisheriger IT-Strategien.

Unternehmen kämpfen im Umfeld unberechenbarer Märkte unter zunehmendem Konkurrenzdruck, sich schnell ändernden Kundenanforderungen, regulatorischen Änderungen und einem rasanten technologischen Fortschritt, um den bestmöglichen Unternehmenserfolg zu erreichen. Unter solchen Bedingungen gelten langfristig ausgelegte Strategiepläne eingeschränkt und bedürfen der fortlaufenden Kontrolle und agiler Anpassungsmechanismen, um Hersteller von B2C- oder B2B-Produkten oder Dienstleister jedweder Branche in einem derart schwierigen Marktumfeld bestmöglich zu unterstützen.

Nicht nur ein schonender, sondern überdies weitgehend flexibler Umgang mit den zur Verfügung stehenden, begrenzten Ressourcen entspannt die Wettbewerbssituation maßgeblich. Wenn Software sich über für Kunden relevante digitale Services definiert und agil an den realen Bedürfnissen der Kunden orientiert, dann wandelt sich IT zum Enabler im Unternehmen und trägt maximal zum Unternehmenserfolg bei.

Die Software-Entwicklung gilt nicht nur selbst als komplexes Problemfeld, sondern auch der Wert von Software ist mitunter schwer zu verifizieren. Dabei muss Software die strategischen und wirtschaftlichen Interessen eines Unternehmens so weitreichend, einfach und kostengünstig wie möglich abdecken. Der Komplexitätsgrad von Software steigt

mit den heterogenen Interessen unterschiedlicher Kundengruppen. Somit werden individualisierte Software-Lösungen, die optimal auf spezialisierte Kundeninteressen ausgerichtet sind, und deren agiles Handling unumgänglich.

Die Wirtschaftlichkeit optimierter Verfahrensweisen steht im scheinbaren Widerspruch zur Individualisierung von Software. Je individueller ein an den Markt gerichtetes Angebot ausgestaltet ist, desto komplexer und aufwendiger werden die Anforderungen und umso relevanter wird die Notwendigkeit einer übergreifenden Priorisierung. Auf diesem Weg lassen sich erwünschte Skaleneffekte durch strukturierte digitale Services, Wirtschaftlichkeitsaspekte hinsichtlich der entsprechenden Ressourcenermittlung zur Umsetzung von Software, das Bedürfnis einer handhabbaren Wartbarkeit, höchste Ansprüche an die IT-Security und eine möglichst kurze Time-to-Market verwirklichen. Wie wichtig agiles und wissenschaftsbasiertes Handeln ist, zeigt sich in den Phasen der jüngsten Corona-Krise und im Kontext nationaler und europäischer Sicherheitsbemühungen um kritische Infrastrukturen umso mehr.

Welche besonderen Anforderungen ein differenziertes Angebot von Software Services stellt und inwiefern agile Prinzipien geeignet sind, dieses Problem wirksam zu lösen, wird in diesem Buch analysiert und anhand praxisbewährter agiler Methodik nachvollziehbar erläutert. Neben der Betrachtung des Produktlebenszyklus und einer grundlegenden Klassifikation von Software werden dabei insbesondere die hierzu erforderlichen Prozesse, Aktivitäten und Rollen beleuchtet. Auf Grundlage dieser Anforderungen werden geeignete agile Frameworks ausgewählt, vorgestellt und hinsichtlich möglicher Schwächen und Stärken untersucht.

Dabei stellt das vorliegende Buch den engen Zusammenhang und das Optimierungspotenzial entlang von Software-Produktlinien dar. Eine Software-Produktlinie ist eine Gruppe von Software-Produkten, die Varianten eines Basisprodukts darstellen und über eine gemeinsame Software-Architektur verfügen.

Damit steht die Betrachtung im Kontext des sogenannten Large-Scale Agile Development. Large-Scale Agile Development wird relevant, sobald es darum geht, eine agile Vorgehensweise auf die Strukturen größerer Organisationen auszuweiten. Auf diese Weise können Projekte mit großen Teams und einer Vielzahl unterschiedlichster Projekte entsprechend den Prinzipien der agilen Entwicklung – über die Grenzen der einzelnen agilen Teams – auf die ganze Organisation ausgeweitet werden.

Motivation und Ziele

Das Buch soll Organisationen bei einer effektiven digitalen Transformation unterstützen. Hierzu werden geeignete Vorgehensmodelle, Methoden und agile Tools beschrieben und erläutert, so dass Unternehmen ihre Organisationsstruktur nachhaltig in Richtung einer übergreifenden, agilen Priorisierung verändern können. Mit solchen agilen Organisationsstrukturen, auch als Large-Scale Agile Frameworks bekannt, wird eine jederzeit anpassungsfähige Umsetzung von beispielsweise cloudbasierten Microservices und die Priorisierung der in agilen Teams organisierten Unternehmenseinheiten bestmöglich unterstützt.

Im Rahmen von – an Beispielen und Anwendungsfällen orientierten Empfehlungen – erfolgt eine praxis- und wissenschaftsorientierte Analyse hinsichtlich der übergreifenden Anforderungspriorisierung zwischen produktspezifischen Entwicklerteams und cross-

funktionalen Teams mit übergreifenden Unterstützungsfunktionen sowie produktübergreifenden Expertenteams mit spezialisierten Aufgaben.

Neuartig ist die Reflexion brandaktueller technischer Themen und die Berücksichtigung des Erfordernisses bestmöglicher IT-Sicherheit in Bezug auf die beschriebenen Technologien und hinsichtlich der Auswirkung auf agile Organisationsmodelle und Priorisierungsmodelle.

Im Ergebnis wird ein geeignetes agiles Organisations- und Priorisierungsmodell vorgestellt, das Organisationen, die sich am Beginn oder bereits mitten im Prozess der agilen Transformation befinden, effektiv unterstützt, um die serviceübergreifende Organisation und Priorisierung zu verbessern.

Mit Ausrichtung auf die effiziente Gestaltung von agilen Software Services stehen die Optimierung, Dimensionen der technischen Rahmenbedingung, der fachlichen Dokumentation und der Interprozesskommunikation im Fokus eines solchen Modells.

Für wen ist dieses Buch?

Ich habe dieses Buch entwickelt, um dabei zu unterstützen, Agilität als festen Bestandteil unserer modernen Arbeitskultur in Unternehmen und Organisationen jeder Art zu etablieren. Losgelöst davon in welchem Reifegrad Sie, Ihre Kolleg*innen und Mitarbeiter*innen sich aktuell befinden, soll dieses Buch als Leitfaden dabei unterstützen, agiles Mindset und agile Arbeitsweisen in Ihrem Berufsalltag zu verankern. Losgelöst davon, ob Sie als Führungskraft oder Teammitglied mehr über agile Methodik erfahren wollen, soll Ihnen dieses Werk als Kompass dienen und navigiert dabei entlang der zurzeit brandaktuellen Themen wie Cloud-Technologien, DevOps und IT-Security. Ein weiteres Ziel ist es, das Bewusstsein für IT-Sicherheit zu schärfen und praktikable Strategien und Lösungen zur Umsetzung aufzeigen. Es ist mir ein Anliegen, dabei zu helfen, ihre Kenntnisse über die Grundlagen der Agilität hinaus zu erweitern. Es wurde auf der Grundlage jahrelanger Erfahrung in unzähligen Projekten und vor dem Hintergrund verschiedenster Unternehmensumgebungen erstellt. Es gibt viele Bücher, die in Agilität und die dazugehörigen Grundlagen einführen. Mein Plan war es, ein Buch zu schreiben, das darüber hinausgeht. Dieses Buch soll Sie und Ihre Organisation effektiv dabei unterstützen, agil durchzustarten und eine neuartige Organisationsform zu etablieren, damit Ihnen gemeinsam mit Ihren Kolleg*innen und Teams – und vor allem mit Spaß – eine erfolgreiche digitale Transformation gelingt!

Vorgehensmodelle, Frameworks und Standards

Welche Vorgehensmodelle, Frameworks und Standards sind ideal geeignet, um agile Software-Projekte erfolgreich zum Abschluss oder in die nächste Prozessphase zu überführen?

Ein Vorgehensmodell beschreibt, wie ein Team innerhalb eines Projekts eine Aufgabe löst und wie das Team hierzu effektiv kooperiert. Wichtig ist dabei, dass das Vorgehen wiederholbar erfolgreich ist, so dass vergleichbare Projekte in ähnlicher Form – in Bezug auf Kosten, Qualität und Zeit – optimiert angegangen werden können.

Dabei legt ein Vorgehensmodell drei wesentliche Elemente fest [1]:

- **Rollen**
- **Artefakte/Produkte**
- **Aktivitäten und Workflows**

Sind Anforderungen am Anfang eines Projekts vollständig bekannt, gelten Wasserfallmodelle als bewährtes Vorgehensmodell, sofern auch keine Erfordernis im weiteren Projektverlauf besteht, diese Anforderungen zu verändern [1, Seite 759].

Eine zweite wichtige, im Projektkontext nicht gegebene Voraussetzung zur Anwendung des Wasserfallmodells ist, dass ein erfahrenes Projektteam mit einer etablierten Technologie die Entwicklung übernimmt. Zusätzlich sind Änderungskosten im Projektverlauf ein relevantes Entscheidungskriterium.

Das Schaubild 1.1 „Änderungskosten im Projektverlauf“ illustriert schematisch, welche Höhe Änderungskosten im Projektverlauf haben, je später diese Anpassungserfordernisse erkannt werden [2, Seite 71/72].

Um die genannten Nachteile teilweise zu kompensieren, können Projekte iterativ umgesetzt werden, mit dem Ziel, Anpassungserfordernisse früher zu erkennen und frühzeitig vom Nutzen der Teilergebnisse zu profitieren. Auch Projekte nach dem Wasserfallmodell können – statt in einem einzigen großen Wasserfall – in Stufen durchgeführt werden. Hierbei entspricht jede Stufe wiederum einem kleinen Wasserfall. Von Stufe zu Stufe können sich Anforderungen ändern und Gelerntes kann umgesetzt werden. Auch das Feedback

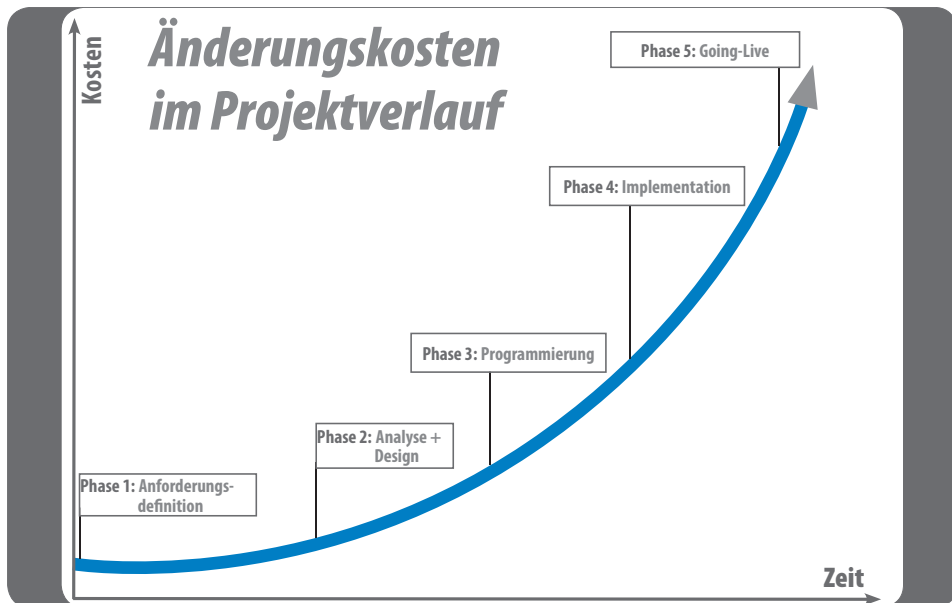


Abb. 1.1 Änderungskosten im Projektverlauf. (Quelle: Sascha Block – Eigene Darstellung)

vom Kunden erfolgt bereits nach der ersten Stufe und nicht erst am Projektende. Jede Stufe kann als eigenes Projekt zum Festpreis behandelt werden.

Mit immer kürzeren Änderungsintervallen von Anforderungen bietet sich inzwischen in den meisten Fällen ein iteratives, prototypisches Vorgehensmodell an.

Bei iterativen Vorgehensmodellen wird ein Projekt in mehreren, kleinen Iterationen organisiert, siehe Abb. 1.2 „Iteratives Projektmodell“. Nach dem iterativen Modell werden somit mit jeder Iteration vollständig alle Phasen durchlaufen.

Von Iteration zu Iteration ist Lernen im Team möglich. Auch Feedback des Kunden findet schnell Berücksichtigung, da bereits nach der ersten Iteration eine funktionierende Software zur Verfügung steht. Damit ist das Projektrisiko für alle Projektpartner geringer, da das Projekt nach jeder Iteration abgebrochen werden kann [1, Seite 760]. In jüngeren iterativen Vorgehensmodellen ist diese Vorgehensweise explizit als Build-Measure-Learn-Zyklus bzw. -Prinzip enthalten.

Zusätzlich wird bei iterativem Vorgehen zwischen evolutionärer und inkrementeller Entwicklung unterschieden. Wird **inkrementell** entwickelt, wächst die Software von Iteration zu Iteration. Auch der Funktionsumfang wird fortlaufend ergänzt, Anforderungen hingegen bleiben weitgehend unverändert.

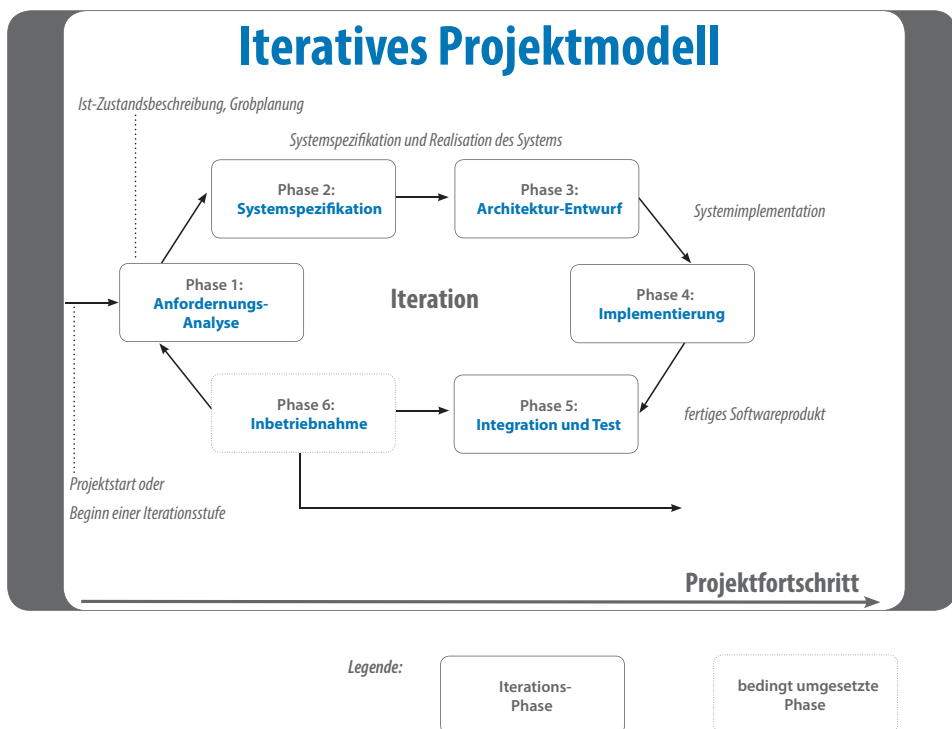


Abb. 1.2 Iteratives Projektmodell. (Quelle: Sascha Block – Eigene Darstellung)

Bei **evolutionärer Iteration** von Software-Lösungen hingegen, können sich auch Spezifikationen und Anforderungen in der Software von einer Iteration zur nächsten Iteration ändern. Somit wird mit der evolutionären Iteration – daher auch der Name dieses Iterationsmodells – auch eine komplette Restrukturierung von Software-Systemen möglich, wenn die Architektur einer Software geänderten Anforderungen nicht genügen kann.

Für Ausgangssituationen mit unklaren Projektvorgaben bieten sich agile Vorgehensmodelle wie Scrum an. Scrum nimmt als Prozessmanagement-Framework einen sehr engen Bezug auf die in einem Projekt geschaffenen Produktartefakte und gibt dafür ein Verfahren zur Verwaltung und Planung von Anforderungen vor. Dabei wird das konkrete Vorgehen im laufenden Entwicklungsprozess vom Projektteam festgelegt [3, Seiten 3 und 4].

Im Unterschied zu plangetriebenen Vorgehensmodellen steht das Team selbst in der Verantwortung für das Ergebnis und lernt somit im konkreten Vorgehen eigenständig, wie es am produktivsten arbeiten kann. Die als Sprints bezeichneten Iterationen dauern in der Regel längstensfalls 30 Tage an. Für jeden Sprint werden Anforderungen ausgewählt, die direkt umgesetzt werden. Somit ist auch die flexible und im Projektteam eigenständige Anpassung von Anforderungen von Sprint zu Sprint möglich. Anforderungen werden nicht zwingend schriftlich, sondern vielfach in persönlichen Gesprächen mit den Stakeholdern definiert. Agile Verfahren nutzen Produktinkremente oder Prototypen, die in kurzen, regelmäßigen Intervallen zur Beurteilung an die Projektkunden ausgeliefert werden. Wertvolles Feedback steht so dem Projektteam stets unmittelbar und direkt zur Verfügung.

Dieses Buch folgt der Motivation, darüber hinausgehende Mechanismen zur Synchronisierung und Priorisierung von Anforderungen zwischen agilen Teams, die dazu förderlichen Methoden, Prozesse und Tools sowie agile Konzepte der Large-Scale Agile Frameworks vorzustellen.

Agile Prozesse und Wissensmanagement

Wie lassen sich agile Prozesse zur Software-Entwicklung einfach, verständlich und transparent gestalten und pragmatisch dokumentieren?

Anforderungen an den Product Owner

Welche Anforderungen sollte ein Product Owner erfüllen und welche Methodik bewährt sich bei wiederkehrenden Aufgaben und zur agilen Dokumentation?

Literatur

1. Ernst, H., Schmidt, J., & Beneken, G. (2016). *Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis. – Eine umfassende praxisorientierte Einführung* (6. Aufl., S. 757). Springer/Vieweg. hier Kapitel 17.4 „Vorgehensmodelle“.
2. Schoeneberg, K.-P. (Hrsg.). (2014). *Komplexitätsmanagement in Unternehmen: Herausforderungen im Umgang mit Dynamik, Unsicherheit und Komplexität meistern* (1. Aufl.). Springer/Gabler.
3. Schwaber, K., & Sutherland, J. (2017). „The Scrum Guide“ – Edition November 2017. <http://www.scrumguides.org>. Zugegriffen am 03.11.2022.