James Shore

Die Kunst der agilen Entwicklung

Grundlagen, Methoden und Praktiken

→ Mit einem Geleitwort von Martin Fowler
→ Aus dem Englischen von Wolf-Gideon Bleek und Tim Müller
Übersetzung der 2., vollständig überarbeiteten Auflage

dpunkt.verlag

Stimmen zur zweiten Auflage von
»Die Kunst der agilen Entwicklung«

»*Die Kunst der agilen Entwicklung*« vollbringt in der 2. Auflage das Kunststück, moderne Softwareentwicklung in einem kurzen, lesbaren und unterhaltsamen Buch zusammenzufassen. Für Neueinsteiger der iterativen Lieferung bietet es einen guten Überblick über die gängigen Praktiken. Für diejenigen, die sich in den Wirren der überdimensionierten »skalierten agilen« Prozesse verirrt haben, zeigt es gute Ideen auf, um dieser Hölle zu entkommen. Die erste Auflage hatte vor zwei Jahrzehnten einen großen Einfluss auf meine Karriere, und ich bin sicher, dass auch die zweite Auflage Millionen von Entwicklerinnen und Entwicklern in ähnlicher Weise dabei helfen wird, die Art und Weise, wie sie Software liefern, zu verbessern.

– Gojko Adzic, Autor von *Running Serverless, Impact Mapping, Specification by Example*

Vom Code bis zur Auslieferung des Produkts – dieses Buch bietet alles. Jahrzehntelanges, hart erarbeitetes Wissen in lesbarer und verdaulicher Form – ein Muss für jeden, der mit oder in einem Softwareentwicklungsteam arbeitet.

– Avi Kessner, Staff Engineer, Forter

Dieses Buch wird einen festen Platz in meinem am besten erreichbaren Bücherregal haben.

– Krishna Kumar, Gründer und CEO, Exathink Research

Die erste Auflage dieses Buches hat mich so fasziniert, dass ich es immer noch als Nachschlagewerk in meinem Bücherregal stehen habe. Die zweite Auflage behält diese Rezeptur bei und fügt noch mehr Erkenntnisse aus dem letzten Jahrzehnt hinzu.

– Benjamin Muskalla, Senior Software Engineer, GitHub

Eines der umfassendsten Bücher über agile Softwareentwicklung, die ich je gelesen habe. Sehr pragmatisch, mit aussagekräftigen Beispielen, die leicht auf jedes Softwareentwicklungsprojekt anwendbar sind, unabhängig von der technischen Ausstattung, der Teamgröße oder der Branche. Auf jeden Fall ein Juwel, das Sie an Ihrem Arbeitsplatz griffbereit haben sollten.

– Luiza Nunes, Programm-Managerin, Thoughtworks

Dies ist mein Lieblingsbuch über agile Entwicklung. Es behandelt technische und Managementthemen. Ich verwende es in meinen Kursen und empfehle es auch immer meinen Kunden.

– Nicolás Paez, Software Engineer und Professor, Universidad de Buenos Aires

Jim deckt seinen erfahrungsbasierten Ansatz zur agilen Softwareentwicklung mit leicht verständlichen Kapiteln, die Konzepte mit Praktiken verbinden, umfassend ab.

– Ken Pugh, Principal Consultant, Autor von *Prefactoring: Extreme Abstraction, Extreme Seperation, Extreme Readability*

Es gibt Tausende Bücher zu Agilität und Sie fragen sich, welches Sie lesen sollten? Ich empfehle Ihnen, dieses Buch in Betracht zu ziehen. James ist seit den frühen Tagen der Agilität dabei und kennt sich aus. Das Buch setzt sich von dem Unfug in unserer Branche, dem bedeutungslosen »Agile«, das überall zu finden ist, ab und bietet einen gründlichen, ganzheitlichen Ansatz. Dieser Ansatz wird nicht schnell und einfach sein, aber es wird sich lohnen. Ich habe »*Die Kunst der agilen Entwicklung*« geliebt. Es ist ein Buch mit Haltung!

– Bas Vodde, Mitbegründer von LeSS

James Shore hat »*Die Kunst der agilen Entwicklung*« mit neuen Werkzeugen, Techniken und Lektionen aus dem letzten Jahrzehnt komplett überarbeitet. Dieses Juwel von einem Buch wird Ihnen dabei helfen, Ihre Arbeitsweise zu einer wirklich agilen und effektiven Art und Weise weiterzuentwickeln.

– Bill Wake, XP123, LLC

Autor

James Shore leitet seit 1999 Teams, die agile Entwicklung praktizieren. Er kombiniert ein tiefes Verständnis der agilen Ideen mit jahrzehntelanger praktischer Erfahrung in der Entwicklung und nutzt diese Erfahrung, um Menschen dabei zu unterstützen, zu verstehen, wie alle Aspekte von Agilität zusammenpassen, um herausragende Ergebnisse zu erzielen. James hat den Gordon Pask Award der Agile Alliance für Beiträge zur agilen Praxis erhalten, ist Moderator mehrerer Screencasts zur Softwareentwicklung und Mitbegründer des Agile Fluency Model. Er ist online unter *jamesshore.com* zu finden.

Übersetzer

Wolf-Gideon Bleek ist bei der it-agile GmbH in Hamburg tätig. Nach dem Studium der Informatik promovierte er zum Thema Software-Infrastruktur-Entwicklung. Er ist Ende der 90er-Jahre zuerst mit agilen Methoden in Kontakt gekommen. In Industrieprojekten für verschiedene Kunden konnte er Erfahrungen in Scrum, XP und Kanban als Developer, Agile Coach und Product Owner sowie als Führungskraft sammeln. Sein Interesse gilt dem Zusammenspiel zwischen Technik, Prozess und Organisation. In Publikationen und auf Konferenzen gibt er seine Erfahrungen weiter.

Tim Müller arbeitet bei der it-agile GmbH in Hamburg und ist ausgebildeter Fachinformatiker in der Fachrichtung Anwendungsentwicklung. Als Softwareentwickler lernte er in verschiedenen Projekten sowohl klassische als auch agile Methoden und Werkzeuge aus dem Extreme Programming kennen. Die Arbeit mit agilen Werten und Methoden überzeugten Tim und sorgten dafür, dass er das dort Erlernte wann immer möglich anwendet. Diese Erfahrung und Begeisterung gibt er jetzt gerne an andere Teams weiter.

James Shore

Die Kunst der agilen Entwicklung

Grundlagen, Methoden und Praktiken

Aus dem Englischen von Wolf-Gideon Bleek und Tim Müller

dpunkt.verlag

James Shore • *jshore@jamesshore.com*
Unter Mitwirkung von Diana Larsen, Gitte Klitgaard & Shane Warden

Lektorat: Christa Preisendanz
Lektoratsassistenz: Julia Griebel, Anja Weimer
Übersetzung: Wolf-Gideon Bleek, Tim Müller
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: III-satz, *www.drei-satz.de*
Herstellung: Stefanie Weidner, Frank Heidt
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

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

ISBN:
Print 978-3-86490-860-6
PDF 978-3-96910-866-6
ePub 978-3-96910-867-3
mobi 978-3-96910-868-0

1. Auflage 2023

Wieblinger Weg 17
69123 Heidelberg

Authorized German translation of the English edition of *The Art of Agile Development, 2nd Edition*
ISBN 9781492080695 © 2021 James Shore and Big Blue Marble LLC

Hinweis:
Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *hallo@dpunkt.de.*

5 4 3 2 1 0

Für meine Familie

Inhaltsübersicht

Inhaltsverzeichnis

Geleitwort

Als wir das Manifest für Agile Softwareentwicklung verfassten, handelte es sich bei den Personen, die uns unterstützten, um eine kleine Minderheit, die versuchte, eine Branche zu verändern. Heute, 20 Jahre später, ist »agil« im Mainstream angekommen. Aber ich schreibe »agil« nicht ohne Grund in Anführungszeichen – viele Leute sagen, dass sie agile Softwareentwicklung betreiben, und die meisten glauben das auch wirklich. Doch ihre Handlungen haben wenig Ähnlichkeit mit der gemeinsamen Vision, die wir vor zwei Jahrzehnten teilten.

Die Wahrheit ist, dass eine agile Arbeitsweise ein Netz miteinander verbundener Praktiken erfordert, die sowohl das Management als auch die technische Ausführung der Softwareentwicklungsarbeit umfassen. Viele dieser Praktiken, insbesondere die technischen, sind nicht gut verstanden oder werden nicht allgemein gelehrt. Infolgedessen haben zu viele eine verfälschte Vorstellung davon, wie effektiv die Entwicklung von Softwareprodukten sein kann.

James Shore war einer der frühen Pioniere auf dem Weg zu Extreme Programming, einer zentralen Säule der agilen Bewegung. Seine erste Auflage dieses Buches war einer meiner Favoriten: ein Handbuch für Teams, das ihnen das nötige Wissen vermittelte, um einen agilen Prozess richtig durchzuführen. Später arbeitete er mit Diana Larsen zusammen, um das Agile Fluency Model zu entwickeln – ein Modell, das ihre Erfahrungen festhält, auf welche verschiedenen Arten Menschen ihre Fähigkeiten im Umgang mit agilen Ansätzen entwickeln können. In diesem Modell bietet eine einfache Anwendung von Projektmanagementtechniken, die oft als grundlegender Scrum-Ansatz bezeichnet wird, einen gewissen Wert, indem sie sich auf die Kundenbedürfnisse konzentriert. Es fehlen aber die technischen Fähigkeiten, die Sie benötigen, um die hohe Produktivität und Zuverlässigkeit auszuschöpfen, die viele Teams erreichen.

Diese Sichtweise bestimmt zu Recht die Struktur dieses Buches, das den Schwerpunkt darauf legt, wie Sie sich auf die Wertstiftung konzentrieren und wie Sie diesen Wert zuverlässig liefern. Ausrichtung an der Wertstiftung heißt, dass Sie die Bedeutung einer starken Teamarbeit verstehen, Fähigkeiten zur adaptiven Planung entwickeln und eng mit den Kunden und Benutzerinnen der entstehenden Software zusammenarbeiten. Zuverlässig liefern konzentriert sich auf

wesentliche technische Praktiken für Tests, Refactoring, Entwurf und kollaborative Entwicklung. Dabei wird die oft kontraintuitive Vorstellung berücksichtigt, dass die Erstellung von Software mit hoher interner Qualität die Kosten senkt und die Geschwindigkeit der Codebereitstellung erhöht. Kombiniert mit einer DevOps-Kultur und Continuous Delivery ermöglicht dies eine hohe Frequenz, Funktionen schnell in Produktion zu bringen, was wiederum den Teams die Möglichkeit gibt, mehr darüber zu erfahren, was wertstiftend ist, indem sie beobachten, wie die Software in der Praxis genutzt wird.

Ich hatte das Glück, vor 20 Jahren bei Thoughtworks ein Zuhause zu finden, wo unsere Teams diese Art von Fähigkeiten nutzen, um unsere Kunden bei der Entwicklung neuer Softwareprodukte zu unterstützen und alte Traditionen abzulösen. Wie James haben auch wir festgestellt, dass Extreme Programming eine solide Grundlage für unsere Arbeit darstellt, und wir haben diese Techniken in den letzten zwei Jahrzehnten mit großem Erfolg angewandt. Ich freue mich daher sehr darüber, dass James ein weiteres Jahrzehnt seiner Coaching-Erfahrung in die Überarbeitung seines Buches eingebracht hat. Das Ergebnis ist eine solide Grundlage für das Erlernen dieser Fähigkeiten, die uns so sehr geholfen haben. Wie alles, was sich lohnt, wird es Zeit brauchen, und auf dem Weg dorthin wird es Frustrationen geben. Aber dieser Leitfaden kann Ihnen auf dieser Reise helfen – weg von öden Zeremonien hin zu der Kraft, die wir empfunden haben, als James und ich diese Techniken vor all den Jahren zum ersten Mal angewandt haben.

– Martin Fowler
Chief Scientist, Thoughtworks

Vorwort

F: Wie komme ich zur Carnegie Hall[1]?
A: Üben, üben, üben!

Ich möchte Ihnen helfen, die Kunst der agilen Entwicklung zu meistern.

Die agile Entwicklung ist, wie jeder teambasierte Ansatz zur Softwareentwicklung, eine grundlegend *menschliche* Kunst, die den Unwägbarkeiten des Einzelnen und seiner Interaktionen unterliegt. Um die agile Entwicklung zu meistern, müssen Sie lernen, unzählige Möglichkeiten von Augenblick zu Augenblick zu bewerten und intuitiv die beste Vorgehensweise auszuwählen.

Wie können wir eine so schwierige Kunst überhaupt erlernen? Durch Üben!

Dieses Buch ist in erster Linie ein Leitfaden für die Praxis. Es ist eine detaillierte Beschreibung einer Art und Weise, wie Sie agile Entwicklung praktizieren können. Es basiert auf Extreme Programming, lässt aber auch Ideen und Praktiken aus Scrum, Kanban, DevOps, Lean Software Development, Lean Startup und anderen Methoden einfließen. Letztendlich ist es ein praktischer Leitfaden, der es Ihnen ermöglicht, die agile Entwicklung erfolgreich in Ihrem Team und Ihrer Organisation einzuführen – oder er wird Ihnen helfen, herauszufinden, dass Agilität keine gute Wahl für Ihre Situation ist.

Zweitens soll Ihnen dieses Buch dabei helfen, die *Kunst* der agilen Entwicklung zu meistern. Agilität zu meistern bedeutet, über ein Kochbuch mit Praktiken hinauszugehen. Softwareentwicklung ist zu kontextabhängig, als dass ein einzelner Ansatz perfekt passen könnte, und zu vielschichtig, als dass Ihnen irgendein Buch lehren könnte, wie Sie sie beherrschen. Die Beherrschung kommt von innen: aus Erfahrung und einem intuitiven Verständnis der Auswirkungen, die selbst die kleinste Entscheidung verursacht.

Ich kann Ihnen nicht beibringen, wie sich Ihre Entscheidungen auf Ihr gesamtes Unternehmen auswirken werden. Ich versuche es auch gar nicht. Sie müssen selbst für die Feinheiten und das Verständnis sorgen. Nur so können Sie diese Kunst beherrschen. Befolgen Sie die Praktiken und beobachten Sie, was passiert.

1. Anm. d. Übers.: Die Carnegie Hall ist ein berühmtes Konzerthaus in New York City.

Überlegen Sie, warum sie funktioniert haben ... oder auch, warum sie nicht funktioniert haben. Dann wiederholen Sie es. Was war gleich? Was war anders? Und warum war es das? Dann wiederholen Sie es erneut. Und noch einmal.

Am Anfang werden Sie vielleicht nicht verstehen, wie Sie die einzelnen Praktiken anwenden sollen. Auf dem Papier sehen sie einfach aus, aber die Umsetzung einiger Praktiken wird schwierig sein. Üben Sie weiter, bis sie einfach anzuwenden sind.

Wenn Ihnen das agile Vorgehen leichter fällt, werden Sie feststellen, dass einige meiner Ratschläge für Sie nicht funktionieren. Am Anfang werden Sie nicht erkennen können, ob das Problem in den Anweisungen liegt, die ich Ihnen gebe, oder in der Art und Weise, wie Sie sie befolgen. Üben Sie weiter, bis Sie sich sicher sind. Wenn Sie sicher sind, brechen Sie die Regeln. Ändern Sie meine Anleitungen, um sie besser auf Ihre spezielle Situation abzustimmen. Jede Praktik hat einen Abschnitt »Alternativen und Experimente« mit Ideen zum Ausprobieren.

Eines Tages werden Regeln für Sie nicht mehr von Interesse sein. Schließlich geht es bei Agilität nicht darum, Regeln zu befolgen. »Es geht um Einfachheit und Feedback, Kommunikation und Vertrauen«, werden Sie denken. »Es geht darum, Wert zu liefern – und den Mut zu haben, das Richtige zur richtigen Zeit zu tun.« Sie werden von Augenblick zu Augenblick unzählige Möglichkeiten bewerten und intuitiv die beste Vorgehensweise auswählen.

Wenn Sie das getan haben, geben Sie dieses Buch an eine andere Person weiter, selbst wenn Ihr Exemplar schon ein paar Eselsohren hat. So kann auch die nächste Person die Kunst der agilen Entwicklung meistern.

Für die Pragmatiker

Was, wenn Sie keine sogenannte »Kunst« beherrschen wollen? Was ist, wenn Sie einfach nur gute Software entwickeln wollen?

Keine Sorge – dieses Buch ist auch für Sie. Ich habe meine jahrelange Erfahrung mit agiler Entwicklung in einem einzigen, klar definierten und umfassenden Ansatz zusammengefasst.

Das erlaubt mir, eine einfache, unkomplizierte Sprache zu verwenden. Ich gebe eine Menge praktischer Tipps und beschreibe offen, wann mein Ansatz nicht funktioniert und welche Alternativen Sie dann in Betracht ziehen können. Kapitel 2 wird Ihnen den Einstieg erleichtern.

Die Erörterung nur eines Ansatzes hat einen Nachteil: Kein einzelner Ansatz ist für alle passend. Meine Ratschläge sind möglicherweise nicht für Ihr Team oder Ihre Organisation geeignet. Lesen Sie die Kapitel 4 und 5, um die allgemeinen Voraussetzungen für den Erfolg zu verstehen, und lesen Sie den Abschnitt »Voraussetzungen« für jede Praktik, um Einzelheiten zu erfahren.

Aber gehen Sie nicht einfach davon aus, dass eine bestimmte Praktik für Sie nicht geeignet ist. Einige der Praktiken in diesem Buch sind kontraintuitiv oder

klingen einfach nicht nach Spaß. Die meisten der Praktiken funktionieren am besten im Zusammenspiel mit anderen. Wenn Sie können, probieren Sie die Praktiken ein paar Monate lang so aus, wie sie beschrieben sind. Sammeln Sie praktische Erfahrungen damit, wie sie in Ihrer Umgebung funktionieren, und passen Sie sie *dann* an.

Ich setze diese Ideen seit mehr als 20 Jahren in die Praxis um. In der richtigen Umgebung funktionieren sie wirklich. Agile Entwicklung hat mehr Spaß gemacht und war erfolgreicher als jeder andere Ansatz zur Softwareentwicklung, den ich ausprobiert habe. Kommen Sie mit auf die Reise.

Was ist neu in der zweiten Auflage?

Diese zweite Auflage von *Die Kunst der agilen Entwicklung* ist eine vollständige, grundlegende Überarbeitung der ersten Auflage. Sie behält den bodenständigen, praktischen Ansatz der ersten Auflage bei, ebenso wie die meisten Praktiken der ersten Auflage. Aber fast alle wurden neu geschrieben, um die Vorteile von 14 Jahren Fortschritt in der agilen Praxis zu nutzen – ganz zu schweigen von 14 weiteren Jahren Erfahrung meinerseits.

Ich habe das Buch komplett umstrukturiert, um eine schrittweise Einführung zu ermöglichen und die reale Nutzung von Agilität durch Teams besser widerzuspiegeln. Die Prinzipien und Anpassungen, die in Teil III der ersten Auflage erläutert wurden, sind auf die Praktiken verteilt worden, um sie deutlicher hervorzuheben und zugänglicher zu machen, und ich habe jede Praktik mit Vorschlägen für Experimente erweitert.

Zu den bemerkenswerten Ergänzungen gehören:

- Ein detaillierter Leitfaden zur Einführung von Agilität und zur Anpassung an die Bedürfnisse Ihres Unternehmens, basierend auf dem *Agile Fluency*[2]-Modell, das ich zusammen mit Diana Larsen entwickelt habe.
- Ein neues Kapitel über agile Skalierung, das auf meiner Erfahrung bei der Unterstützung großer und kleiner Unternehmen basiert.
- Ein neues Kapitel zu DevOps mit neuen Inhalten über die Zusammenarbeit mit den Bereichen Betrieb und Sicherheit sowie von DevOps inspirierte Aktualisierungen im gesamten restlichen Buch.
- Anleitungen für den Einsatz von Agilität in Teams, die remote arbeiten; viele neue Praktiken, Geschichten und Ideen und zu viele weitere Verbesserungen und Änderungen, um sie hier alle aufzuführen.

2. »Agile Fluency« ist eine eingetragene Marke von Agile Fluency Project LLC.

Für wen dieses Buch bestimmt ist

Dieses Buch richtet sich an alle, die in einem agilen Team arbeiten oder hoffen, dies in Zukunft zu tun. Dazu gehören natürlich Programmiererinnen, aber auch Manager, Führungskräfte, Domänenexperten, Tester, Produktmanager, Projektmanagerinnen, Architekten, Betriebsleiter, Sicherheitsexperten, Designer und Business-Analysten. Agile Teams sind funktionsübergreifend; dieses Buch spiegelt diese Tatsache wider.

Das Buch ist so angelegt, dass es sowohl als Nachschlagewerk zu verwenden ist als auch von vorne bis hinten gelesen werden kann. Jede Praktik in Teil II bis Teil IV ist so gestaltet, dass sie für sich allein gelesen werden kann. Die Kästen »Verwandte Themen« und die Hyperlinks in der E-Book-Ausgabe helfen Ihnen beim Nachschlagen. Die gedruckte Ausgabe ist außerdem so konzipiert, dass Sie sie in die Hand nehmen und durchblättern können. Blättern Sie durch das Buch und halten Sie inne, um tiefer einzusteigen, wenn eine Aussage Ihre Aufmerksamkeit erregt.

Wenn Sie ein Manager oder eine Führungskraft sind und verstehen möchten, wie Agilität in Ihrem Unternehmen funktionieren kann oder sollte, lesen Sie Teil I. Wenn Sie ein Manager auf Teamebene sind, lesen Sie auch im Abschnitt »Management« auf Seite 382 und möglicherweise die anderen Praktiken in Kapitel 10.

Wenn Sie als Teammitglied oder Manager daran interessiert sind, Agilität in Ihrem Unternehmen einzuführen, oder die Art und Weise, wie Agilität in Ihrem Unternehmen praktiziert wird, zu verbessern, lesen Sie das gesamte Buch von Anfang bis Ende. Teil I wird Ihnen helfen zu verstehen, wie Sie agile Ideen einführen können. Der Rest des Buches unterstützt Sie dabei, zu verstehen, wie Sie Agilität in die Praxis umsetzen können.

Wenn Sie Teil eines agilen Teams sind und nur genug lernen wollen, um Ihre Arbeit zu erledigen, können Sie sich auf die Praktiken in Teil II und Teil III konzentrieren. Beginnen Sie mit Kapitel 1, um sich einen Überblick zu verschaffen, und lesen Sie dann die Praktiken durch, die Ihr Team verwendet. Wenn Ihr Team Praktiken anwendet, die nicht im Inhaltsverzeichnis aufgeführt sind, sehen Sie im Index nach. Sie könnten unter einem anderen Namen aufgeführt sein.

Wenn Sie nicht Teil eines agilen Teams sind, aber mit einem solchen zusammenarbeiten, bitten Sie die Teammitglieder um Vorschläge, was Sie lesen sollten. Produktmanager, Product Owner und Designer sollten mit Kapitel 8 und dem Abschnitt »Zweck« auf Seite 145 beginnen. Wenn Sie aus den Bereichen Sicherheit und Betrieb kommen: Lesen Sie die Abschnitte »Für den Betrieb bauen« auf Seite 589, »Aufdecken blinder Flecken« auf Seite 636 und »Vorfallanalyse« auf Seite 644. Für Tester bietet sich insbesondere Kapitel 16 an.

Wenn Sie einfach nur neugierig auf die agile Entwicklung sind, beginnen Sie mit der Lektüre von Kapitel 1. Werfen Sie danach einen Blick auf Teil II, Teil III und Teil IV. Starten Sie mit den Praktiken, die Ihnen am interessantesten erscheinen. Sie können sie in beliebiger Reihenfolge lesen.

Über die Gastautorinnen und den Gastautor

Ich bin in der glücklichen Lage, dass ich mehrere namhafte Mitstreiter und Mitstreiterinnen für diese Auflage gewinnen konnte. Gitte Klitgaard hat den Abschnitt »Sicherheit« auf Seite 134 geschrieben, in dem sie das Thema der psychologischen Sicherheit fachkundig behandelt. Diana Larsen hat die Abschnitte »Teamdynamik« auf Seite 408 und »Beseitigung von Hindernissen« auf Seite 425 verfasst und bringt damit ihre jahrzehntelange Erfahrung in der Organisations- und Teamentwicklung ein. Shane Warden, mein Co-Autor der ersten Auflage, konnte kein neues Material zu dieser Auflage beisteuern, aber er blieb ein wertvoller Gesprächspartner und unsere Arbeit an der ersten Auflage bildete die Grundlage für diese Auflage.

Gitte Klitgaard

Gitte Klitgaard ist Agile Coach, Trainerin und Mentorin mit dem Schwerpunkt, Organisationen bei der Einführung von psychologischer Sicherheit, Verantwortung und Verantwortlichkeit zu helfen. Gitte ist authentisch; sie bringt die Dinge auf den Punkt und hilft den Menschen, zu sich selbst zu finden und dadurch erfolgreich zu sein.

Zu ihren Beiträgen in der Community gehören die Organisation von Coach-Camps und Vorträge auf Konferenzen, bei denen sie Themen wie psychische Gesundheit und psychologische Sicherheit hervorhebt und zugänglich macht. Sie schafft sichere und respektvolle Umgebungen am Arbeitsplatz und darüber hinaus. Sie hört den leiseren Stimmen und Minderheitengruppen zu und engagiert sich für sie.

In ihrer Freizeit sammelt Gitte LEGO und Yodas und hält Kontakt zum Freundeskreis aus der ganzen Welt, darunter einige Personen, die sie als ihre zweite Familie betrachtet.

Gitte ist Inhaberin von Native Wired und hat den Wandel bei Unternehmen wie LEGO, Spotify und Mentimeter vorangetrieben.

Diana Larsen

Seit über 20 Jahren trägt Diana Larsen zu den Grundlagen und Erweiterungen des agilen Denkens und zur Praxis der Bildung und Befähigung qualifizierter Teams bei. Diana Larsen ist Co-Autorin von *Agile Retrospectives, Liftoff, 2nd ed.* und *Five Rules for Accelerated Learning*, zwei neuen Büchern, die derzeit in Arbeit sind, sowie vom E-Book *The Agile Fluency Model: A Brief Guide to Success with Agile*. Als verantwortlicher Coach, Beraterin, Moderatorin, Sprecherin und Mentorin führt sie ihre Beiträge fort und wird ihrem Titel »Chief Connector« des Agile Fluency-Projekts gerecht. Diana lebt in Portland an der nördlichen Westküste der USA.

Shane Warden

Shane Warden ist ein führender Ingenieur und Autor, insbesondere der Co-Autor von *The Art of Agile Development* (1. Auflage) und *Masterminds of Programming*. Wenn er nicht arbeitet, hilft er dabei, Tieren ein gutes Zuhause zu geben.

In diesem Buch verwendete Konventionen

In diesem Buch werden die folgenden typografischen Konventionen verwendet:

Zielgruppe

In den Kästchen für die Zielgruppe wird die Zielgruppe für jede agile Praktik angegeben.

Kursiv
Kennzeichnet neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateierweiterungen.

`Nichtproportionale Schrift`
Wird für Programmlistings sowie innerhalb von Absätzen verwendet, um auf Programmelemente wie Variablen- oder Funktionsnamen, Datenbanken, Datentypen, Umgebungsvariablen, Anweisungen und Schlüsselwörter zu verweisen.

Hinweis-Kästen heben wichtige Begriffe hervor.

`Nichtproportionale Schrift fett`
Zeigt Codeergänzungen in laufenden Codebeispielen an.

~~`Nichtproportionale Schrift durchgestrichen`~~
Stellt Codelöschungen in laufenden Codebeispielen dar.

Verwandte Themen

Diese Kästen verweisen auf verwandte Praktiken.

Codebeispiele verwenden

Zusätzliches Material steht unter *https://www.jamesshore.com/v2/books/aoad2* zum Download bereit.

Bitte senden Sie eine E-Mail an *hallo@dpunkt.de*, wenn Sie eine technische Frage oder ein Problem bei der Verwendung des Materials haben.

Dieses Buch soll Ihnen helfen, Ihre Arbeit zu erledigen. Wenn in diesem Buch Beispielcode angeboten wird, dürfen Sie diesen in Ihren Programmen und Ihrer Dokumentation verwenden. Sie müssen uns nicht um Erlaubnis bitten, es sei denn, Sie reproduzieren einen wesentlichen Teil des Codes. Wenn Sie zum Beispiel ein Programm schreiben, das mehrere Codeabschnitte aus diesem Buch verwendet, benötigen Sie keine Erlaubnis. Der Verkauf oder die Verbreitung von Beispielen aus Büchern aus dem dpunkt.verlag erfordert jedoch eine Genehmigung. Die

Beantwortung einer Frage durch Zitieren dieses Buches und von Beispielcode erfordert keine Erlaubnis. Das Einbinden eines bedeutenden Teils des Beispielcodes aus diesem Buch in die Dokumentation Ihres Produkts erfordert eine Genehmigung.

Wir freuen uns über eine Namensnennung, verlangen sie aber im Allgemeinen nicht. Eine Quellenangabe umfasst in der Regel den Titel, den Autor, den Verlag und die ISBN. Zum Beispiel: »*Die Kunst der agilen Entwicklung* von James Shore (dpunkt.verlag). Copyright 2023 dpunkt.verlag GmbH, ISBN (Print): 978-3-86490-860-6.«

Wenn Sie der Meinung sind, dass Ihre Verwendung von Codebeispielen nicht in den Bereich der fairen Nutzung oder der oben genannten Erlaubnis fällt, können Sie uns gerne unter *hallo@dpunkt.de* kontaktieren.

Danksagung

Dieses Buch sammelt Inspirationen aus zu vielen Quellen, um sie alle aufzuführen. Ich habe so viele wie möglich in diesem Buch erwähnt, aber zweifellos einige vergessen. (Bitte nehmen Sie meine Entschuldigung dafür an.) Ich möchte insbesondere Kent Beck, Ron Jeffries und Ward Cunningham für die Entwicklung von Extreme Programming danken, mit dem ich meine agile Reise begonnen habe. Alistair Cockburn und sein Software Roundtable waren ein wichtiger Wegweiser zu Beginn dieser Reise, ebenso wie die lebhaften Debatten und Diskussionen im C2-Wiki von Ward Cunningham. Vielen Dank auch an Diana Larsen, mit der ich seit vielen Jahren zusammenarbeite und deren Perspektive so gut mit meiner übereinstimmt und sie ergänzt. Und natürlich Martin Fowler, dessen nachdenkliche Schriften mich seit vielen Jahren inspirieren.

Die Unterstützung von O'Reilly für diese [englische] Ausgabe war geradezu vorbildlich. Ich schulde Gary O'Brien, meinem Entwicklungsredakteur, der mir ständig Feedback und Unterstützung gegeben hat, ein großes Dankeschön. Mein Dank gilt auch Melissa Duffield, die dem Buch zum Erfolg verholfen hat; Ryan Shaw, der mich davon überzeugt hat, dass es Zeit für eine zweite Auflage war; Deborah Baker, die die Early-Release-Ausgaben des Buches vorbereitet hat; Suzanne Huston, die dafür gesorgt hat, dass die Leute das Buch kennen; Nick Adams und dem Team von O'Reilly Tools, die die Produktionspipeline aufgebaut und sich um meine esoterischen und pingeligen Formatierungswünsche gekümmert haben; Christopher Faucher, der die Umwandlung vom »Rohmanuskript« zum »fertigen Buch« überwachte; Tonya Trybula und Stephanie English, die meine grammatikalischen Macken korrigierten; Kate Dullea, die meine handgezeichneten Skizzen in tatsächlich verständliche Abbildungen umwandelte; und Estalita Slivoskey, die dafür sorgte, dass Sie alles im Index finden können.

Apropos Feedback: Ein besonderer Dank geht an meine Reviewer. Das Review wurde offen durchgeführt, und Dutzende von Menschen haben über 700 Feedback-E-Mails beigesteuert. Fast alle waren aufschlussreich und nützlich, und sie haben zu einem besseren Buch geführt. Mein Dank gilt auch denjenigen, die auf meine speziellen Bitten um Feedback geantwortet haben. Insgesamt möchte ich mich bedanken bei Adrian Sutton, Anthony Williams, Avi Kessner, Bas Vodde, Benjamin Muskalla, Bill Wake, Brad Appleton, C. Keith Ray, C. J. Jameson, Christian Dywan, David Poole, Diana Larsen, Diego Fontdevila, Emily Bache, Erik Peterson, »Evan M«, Franz Amador, George Dinwiddie, Gojko Adzic, Jason Yip, Jeff Grigg, Jeff Patton, Jeffrey Palermo, Johan Aludden, Ken Pugh, Krishna Kumar, Liz Keogh, Luiza Nunes, Marcelo Lopez, Markus Gärtner, Martin Fowler, Michal Svoboda, Nicolas Paez, Paul Stephenson, Peter Graves, Reuven Yagel, Ricardo Mayerhofer, Ron Jeffries, Ron Quartel, Sarah Horan Van Treese, Steve Bement, Thomas J. Owens, Todd Little und Ward Cunningham.

Ein besonderer Dank geht an die Reviewer, die alles gegeben haben, um fast jeden Teil des Buches zu lesen und zu kommentieren: Bas Vodde, Bill Wake, Ken Pugh, Martin Fowler und Thomas J. Owens.

Die erste Auflage profitierte ebenfalls von einem offenen Begutachtungsverfahren, und diese Vorteile kamen auch dieser Auflage zugute. Vielen Dank an Adrian Howard, Adrian Sutton, Ann Barcomb, Andy Lester, Anthony Williams, Bas Vodde, Bill Caputo, Bob Corrick, Brad Appleton, Chris Wheeler, Clarke Ching, Daði Ingólfsson, Diana Larsen, Erik Petersen, George Dinwiddie, Ilja Preuß, Jason Yip, Jeff Olfert, Jeffrey Palermo, Jonathan Clarke, Keith Ray, Kevin Rutherford, Kim Gräsman, Lisa Crispin, Mark Waite, Nicholas Evans, Philippe Antras, Randy Coulman, Robert Schmitt, Ron Jeffries, Shane Duan, Tim Haughton und Tony Byrne für ihre ausführlichen Kommentare sowie an Brian Marick, Ken Pugh und Mark Streibeck für ihre Kommentare zum fertigen Entwurf.

Abschließend möchte ich mich noch einmal bei meiner Frau Neeru bedanken. Dieses Mal wusstest du, worauf du dich einlässt, und trotzdem hat deine Unterstützung nie nachgelassen. Ohne dich hätte ich es nicht geschafft.

Vorwort der Übersetzer

Die erste Auflage von *The Art of Agile Development* ist ein Klassiker und leider gab es keine deutsche Übersetzung. Als wir erfuhren, dass James Shore an der zweiten Auflage arbeitet, waren wir begeistert und dachten sofort, dass es dieses Sammelwerk agiler Praktiken leicht zugänglich in deutscher Sprache geben sollte. Wir sind als Übersetzerteam davon fasziniert, wie umfassend das Buch ist und wie dabei trotzdem in allen Kapiteln auf die Grundideen und Prinzipien von Agilität verwiesen wird.

Natürlich stellte sich uns bei der Übersetzung die Frage, wie wir mit Fachbegriffen umgehen sollten. Wir entschieden uns dafür, englische Wörter, die bereits im Duden aufgenommen wurden und die Bedeutung passend wiedergeben, nicht zu übersetzen. Beim ersten Auftreten eines solchen Begriffs im Buch erklären wir diesen auf Deutsch.

Beim Übersetzen vom Englischen ins Deutsche standen wir dann vor der Herausforderung, wie wir in unserer Übersetzung Geschlechter gleichberechtigt ansprechen. Im Austausch mit dem Verlag entschlossen wir uns, die geschlechtsspezifischen Rollen im Text immer mal wieder abzuwechseln, um keine Stereotypen zu bedienen.

Wir hoffen, es ist uns gelungen, unsere Begeisterung für das Original in die deutsche Fassung zu übertragen.

Wolf-Gideon Bleek und Tim Müller
Hamburg, September 2022

Teil I

Agilität verbessern

1 Was ist Agilität?

Agilität ist überall und paradoxerweise nirgendwo.

In den 20 Jahren, in denen die Agilität wie eine Dampfwalze in das Bewusstsein von Softwareentwicklern[1] eingedrungen ist, stieg die Zahl der Unternehmen, die sich selbst »agil« nennen, beachtlich an. Gilt dasselbe auch für die Anzahl der Teams, die wirklich ein agiles Vorgehen für ihre Arbeit verwenden? Leider nicht. Der inflationär verwendete Begriff »Agilität« ist enorm erfolgreich. Die tatsächlichen Ideen hinter Agilität jedoch werden meistens ignoriert.

Das müssen wir ändern!

1.1 Die Entstehung von Agilität

In den 1990er-Jahren schien sich die Softwareentwicklung in einer Krise zu befinden. Tatsächlich wurde genau dieser Begriff verwendet: »die Softwarekrise«. Softwareprojekte überschritten ihre Budgets, waren verspätet und erfüllten nicht die an sie gestellten Anforderungen. Nach dem oft zitierten und ominös benannten »CHAOS Report« wurde nahezu ein Drittel der Projekte komplett abgebrochen [Standish 1994].

Agilität war nicht im Entferntesten eine Antwort auf diese Krise. Agilität war eine Antwort auf die *Antwort*.

Um die Softwareentwicklung unter Kontrolle zu bringen, hatten große Organisationen sehr detaillierte Prozesse definiert, die genau beschrieben, wie Software erstellt werden sollte. Alles wurde streng kontrolliert, damit (zumindest theoretisch) keine Fehler gemacht werden konnten.

Zuerst würden Business-Analysten verschiedene Stakeholder[2] befragen, um die Anforderungen an das System zu dokumentieren. Im Anschluss würden Soft-

1. Anm. d. Übers.: Um alle Geschlechter gleichberechtigt anzusprechen, werden wir bei den geschlechtsspezifischen Rollen im Text immer mal wieder das Geschlecht abwechseln.
2. Anm. d. Übers.: Wir haben uns dazu entschlossen, englische Wörter, die bereits in den Duden aufgenommen wurden und die Bedeutung passend wiedergeben, nicht zu übersetzen.

warearchitektinnen die Systemanforderungen lesen und mithilfe detaillierter Dokumentation sowohl den Entwurf der Komponenten als auch ihre Beziehung zueinander spezifizieren. Dann würden die Programmiererinnen diese Entwurfsdokumente in Quellcode umwandeln. In einigen Unternehmen wurde dies als eine Arbeit betrachtet, die wenig Fähigkeiten erfordert – also lediglich als eine mechanische Übersetzungsübung.

In der Zwischenzeit würden Fachleute aus der Testabteilung anhand derselben Dokumente Ablaufpläne für den Test anfertigen, damit nach der Programmierung Heerscharen von Testern das Programm manuell überprüfen und Abweichungen als Fehler dokumentieren können. Nach jeder Phase dieses Ablaufplans würden die einzelnen Schritte aufmerksam dokumentiert, überprüft und unterschrieben werden.

Ein solches auf Phasen basierendes Vorgehensmodell wurde »Wasserfallmodell« oder auch »Stage-Gate-Modell« genannt.[3] Wenn das für Sie wie ein vorgeschobenes Argument klingt, nun ja, schätzen Sie sich glücklich. Nicht jedes Unternehmen benutzte in der 90er-Jahren solch einen auf Dokumenten und Phasen basierenden Prozess. Auf diese Art zu arbeiten, wurde jedoch als logisch und vernünftig angesehen. *Natürlich* sollten wir Anforderungen definieren, entwerfen, implementieren und testen. *Natürlich* sollten wir jede Phase dokumentieren. So funktionierte *Entwicklung* und wie sollten wir auch sonst erfolgreich sein?

1.2 Aus der Krise geboren

Große Unternehmen haben ihre Prozesse bis ins kleinste Detail definiert. Rollen, Verantwortlichkeiten, Dokumentenvorlagen, Modellierungssprachen, Change Control Boards, die über Anpassungen von Anforderungen berieten, … jeder Aspekt im Ablauf der Entwicklung wurde streng definiert und überprüft. Wenn ein Projekt nicht erfolgreich war – und nach dem CHAOS Report war dies bei weniger als einem Sechstel der Fall –, lag es daran, dass die Prozessbeschreibung mehr Details brauchte, mehr Dokumente, mehr Freigaben. Daraus resultierte ein riesiger Berg an Dokumentation. Martin Fowler nannte es »Der große Donnerschlag« (»The Almighty Thud«) [Fowler 1997].

Dies war keine schöne Art zu arbeiten: bürokratisch und nicht den Menschen gerecht. Es schien, als hätten Fähigkeiten weniger Bedeutung als die Einhaltung

3. Der Begriff »Wasserfall« wird fälschlicherweise oft auf einen Artikel von Winston Royce aus dem Jahr 1970 zurückgeführt. Jedoch reichen phasenbasierte Ansätze bis in die 1950er-Jahre zurück und der Artikel von Royce wurde bis in die späten 1980er-Jahre weitgehend ignoriert, bis er als Beschreibung dafür verwendet wurde, was die Menschen bereits taten [Bossavit 2013, Kap. 7].

von Prozessen. Und in der Programmierung zu arbeiten, fühlte sich an, als seien wir ein austauschbares Zahnrad in einer anonymen Maschine. Und es funktionierte nicht einmal besonders gut.

Also entwickelten verschiedene Personen Methoden für die Softwareentwicklung, die einfacher und schlanker waren und weniger Vorgaben machten. Diese wurden im Gegensatz zu dem schwergewichtigen Vorgehensmodell großer Unternehmen als »leichtgewichtige Methoden« bezeichnet. Diese neuen Methoden hatten Namen wie »Adaptive Software Development«, »Crystal«, »Feature-Driven Development«, »Dynamic Systems Development Method«, »Extreme Programming« und »Scrum«.

In den späten 90er-Jahren erregten diese Methoden starke Aufmerksamkeit. Insbesondere Extreme Programming führte zu einem explosionsartigen Anstieg des Interesses unter Programmierern. Im Jahr 2001 trafen sich 17 Befürworter dieser leichtgewichtigen Methoden in einem Skigebiet in Utah, um zu diskutieren, wie sie ihre Bemühungen vereinheitlichen könnten.

1.3 Das Manifest für Agile Softwareentwicklung

Alistair Cockburn sagte später: »Ich persönlich habe nicht erwartet, dass diese Gruppe [von Menschen] sich jemals auf etwas Wesentliches einigen würde.«

Und tatsächlich erreichten sie nach zwei Tagen nur zwei Dinge: den Namen »Agilität« und eine Angabe von vier Werten (siehe Abb. 1–1). In den darauffolgenden zwölf Monaten erarbeiteten sie per Mail zwölf begleitende Prinzipien (siehe Abb. 1–2) [Beck 2001].

Das war das Agile Manifest. Es hat die Welt verändert. Oder mit Bezug auf die Aussage von Alistair Cockburn oben: Sie haben sich nach zwei Tagen auf wesentliche Dinge einigen können.[4]

4. Alistair Cockburn, zitiert von Jim Highsmith in [Highsmith 2001]. Das vollständige Zitat lautet: »Ich persönlich habe nicht erwartet, dass … diese bestimmte Gruppe von Agilisten sich jemals auf etwas Wesentliches einigen würde … Ich für meinen Teil bin begeistert von der endgültigen Formulierung [des Manifests]. Ich war überrascht, dass die anderen von der abschließenden Formulierung ebenso begeistert zu sein schienen. Wir haben uns also auf etwas Wesentliches geeinigt.«

Manifest für Agile Softwareentwicklung

Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

- **Individuen und Interaktionen** mehr als Prozesse und Werkzeuge
- **Funktionierende Software** mehr als umfassende Dokumentation
- **Zusammenarbeit mit dem Kunden** mehr als Vertragsverhandlung
- **Reagieren auf Veränderung** mehr als das Befolgen eines Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor Ken Schwaber Jeff Sutherland Dave Thomas

Abb. 1–1 *Agile Werte*[5]

Aber es gab nicht **die eine** agile Methode. Es gab sie nie und es wird sie nie geben. Agilität besteht aus drei Dingen: dem Namen, den Werten und den Prinzipien. Mehr gibt es nicht. Es ist nicht etwas, was Sie tun können, sondern eine *Philosophie*. Eine Art, über Softwareentwicklung zu denken. Wir können Agilität nicht »benutzen« oder »machen« … wir können nur agil *sein* oder eben nicht. Wenn das Team die Philosophie von Agilität verkörpert, ist es agil. Wenn es das nicht tut, dann ist es nicht agil.

Wenn das Team die Philosophie von Agilität verkörpert, ist es agil. Wenn es das nicht tut, dann ist es nicht agil.

1.4 Die Essenz von Agilität

Martin Fowler hat Karriere gemacht, indem er für komplizierte Themen der Softwareentwicklung gut durchdachte und ausgewogene Erklärungen gibt. Seine Erklärung für »Die Essenz von agiler Softwareentwicklung« ist eine der besten:

> Agile Entwicklung ist eher *adaptiv* als vorausschauend;
> orientiert sich eher an *Menschen* als an Prozessen[6].
>
> – Martin Fowler

5. © 2001, die oben genannten Autoren. Diese Erklärung darf in jeder Form frei kopiert werden, jedoch nur in ihrer Gesamtheit durch diesen Hinweis.
6. Fowler hat diese Idee im Laufe der Jahre auf vielfältige Weise zum Ausdruck gebracht. Das Original stammt aus [Fowler 2000].

Prinzipien hinter dem Agilen Manifest

Wir folgen diesen Prinzipien:

Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.

Heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.

Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.

Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.

Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen, und vertraue darauf, dass sie die Aufgabe erledigen.

Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.

Funktionierende Software ist das wichtigste Fortschrittsmaß.

Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.

Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.

Einfachheit – die Kunst, die Menge nicht getaner Arbeit zu maximieren – ist essenziell.

Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.

In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann, und passt sein Verhalten entsprechend an.

Abb. 1–2 *Agile Prinzipien*

Eher adaptiv als vorausschauend

Erinnern Sie sich an den »CHAOS Report«, der besagt, dass nur ein Sechstel aller Softwareprojekte erfolgreich ist? Der Report hatte eine sehr genaue Definition von Erfolg:

Erfolgreich abgeschlossen
: Das Projekt wurde rechtzeitig und ohne Kostenüberschreitung sowie mit dem ursprünglich geforderten Funktionsumfang abgeschlossen.

Teilweise erfolgreich
: Das Projekt wurde abgeschlossen, es kam jedoch zu Kosten- und Zeitüberschreitungen oder es wurde nicht der vollständig geplante Funktionsumfang erreicht.

Nicht erfolgreich
Das Projekt wurde abgebrochen oder niemals eingesetzt.

Diese Definitionen beschreiben das Mindset der Vorhersagbarkeit perfekt. Es geht bei allen Begriffen um die *Einhaltung eines Plans*. Hielten wir uns an die Versprechen, waren wir erfolgreich. Konnten wir die Versprechen nicht einhalten, waren wir es eben nicht, so einfach ist das.

Auf den ersten Blick ergibt das einen Sinn. Aber schauen Sie genauer hin. Es fehlt etwas. Ryan Nelson hat es im CIO Magazine geschrieben [Nelson 2006]:

> Projekte, die die klassischen Kriterien für Erfolg – innerhalb der Zeit, des Budgets und der Spezifikationen zu sein – erreichen, können trotzdem Fehlschläge sein, wenn sie die Bedürfnisse der Benutzergruppe nicht treffen oder keinen Mehrwert zum Unternehmen beitragen. … In ähnlicher Weise können sich Projekte, die nach klassischen IT-Kennzahlen Fehlschläge sind, als Erfolg herausstellen. Dies ist der Fall, wenn sie trotz Nichteinhaltung von Zeit, Budget oder Spezifikation die Bedürfnisse ihrer Benutzergruppe treffen oder einen unerwarteten Mehrwert liefern.

Agile Teams bewerten ihren Erfolg anhand des *Werts*, den sie liefern. Nicht daran, ob sie einen Plan einhalten. Faktisch halten echte agile Teams immer danach Ausschau, den zu liefernden Wert zu maximieren, indem sie ihre Pläne *anpassen*.

Agile Teams bewerten ihren Erfolg anhand des *Werts*, den sie liefern. Nicht daran, ob sie einen Plan einhalten.

Schauen Sie noch einmal zurück auf das Agile Manifest (siehe Abb. 1–1 und Abb. 1–2) und nehmen sich einen Moment Zeit, um die agilen Werte und Prinzipien wirklich zu studieren. Wie viele von ihnen beziehen sich darauf, wertvolle Software zu liefern und anhand von Feedback anzupassen?

Orientiert sich eher an Menschen als an Prozessen

Schwergewichtige Prozesse versuchen, Fehler zu vermeiden, indem sie sorgfältig jeden Aspekt der Softwareentwicklung definieren. Durch das Einbringen von Intelligenz (»Smarts«) in den Prozess wurden individuelle Fähigkeiten weniger wichtig. In der Theorie könnten Sie den gleichen Prozess immer wieder mit unterschiedlichen Individuen anwenden und dabei dieselben Ergebnisse erreichen. (Wenn ich so darüber nachdenke, sind sie irgendwie zu denselben Ergebnissen gekommen. Nur nicht zu den Ergebnissen, die sie erhofft hatten.)

Agilität besagt, dass Individuen der wichtigste Faktor für den Erfolg in der Softwareentwicklung sind. Nicht nur ihre Fähigkeiten, sondern ihr gesamtes Wesen. Dazu

Agilität besagt, dass Individuen der wichtigste Faktor für den Erfolg in der Softwareentwicklung sind.

zählt, wie gut sie als Team zusammenarbeiten, wie viele Ablenkungen ihnen begegnen und wie sicher sie sich fühlen, ihre Meinung zu äußern und zu vertreten und zu guter Letzt, ob sie durch ihre Arbeit motiviert sind.

Agile Teams haben einen Prozess – jedes Team hat ihn, selbst wenn er implizit ist –, aber der Prozess steht im Dienst der Individuen, nicht andersherum. Agile Teams sind für ihren eigenen Prozess verantwortlich und wenn sie einen besseren Weg entdecken, ihrer Arbeit nachzugehen, dann folgen sie ihm.

Schauen Sie noch mal auf das Agile Manifest (siehe Abb. 1–1 und Abb. 1–2). Welche Werte und Prinzipien beziehen sich darauf, Individuen an die erste Stelle zu setzen?

1.5 Warum Agilität gewonnen hat

In den ersten zehn Jahren nach Veröffentlichung des Manifests erlebte Agilität enorm viel Kritik. Das Vorgehen sei »nicht diszipliniert« und würde »niemals funktionieren«. Weitere zehn Jahre später sind die kritischen Stimmen verstummt. Agilität war überall, zumindest dem Namen nach vertreten. Schwerfällige Wasserfallmethoden waren praktisch tot und jüngere Programmiererinnen können sich schwer vorstellen, dass irgendjemand jemals so gearbeitet hat.

Es ist nicht so, dass phasenbasierte Verfahren von Natur aus fehlerhaft sind. Sie haben sicherlich ihre Schwächen, doch wenn sie schlank sind und in einer gut verstandenen Domäne Verwendung finden, können sie funktionieren. Das Problem waren die schwergewichtigen Ansätze großer Unternehmen. Ironischerweise haben die Prozesse, die Probleme *verhindern* sollten, viele dieser Probleme *ausgelöst*.

Es ist schwierig, sich vorzustellen, wie Software funktionieren wird, bevor wir sie tatsächlich benutzen. Noch schwieriger ist es, wirklich jede Sache zu bedenken, die die Software können muss. Dies gilt umso mehr für Menschen, die nicht aktiv an der Softwareentwicklung beteiligt sind. Deshalb ist es entscheidend, so schnell wie möglich laufende Software zu den Nutzerinnen zu bringen. Wir brauchen die Rückmeldung darüber, was noch fehlt oder was nicht funktioniert. Danach passen wir auf der Grundlage der gewonnenen Erkenntnisse unsere Pläne an. Wie es im Agilen Manifest nachzulesen ist: Funktionierende Software ist das wichtigste Fortschrittsmaß. Lernen und das Reagieren auf Veränderungen sind der Kern von Agilität.

> Lernen und das Reagieren auf Veränderungen sind der Kern von Agilität.

Bei diesen schwergewichtigen Prozessen wurde ein so großer Wert auf Kontrolle, Dokumentation und Freigaben gelegt, dass enorme Verzögerungen und ein hoher Overhead entstanden. Mit solchen Prozessen brauchte es Jahre, um funktionierende Software zu entwickeln, und bis kurz vor Projektende gab es noch nichts Konkretes vorzuweisen. Anstatt Veränderungen willkommen zu heißen,

wurde aktiv daran gearbeitet, Veränderungen zu verhindern. Tatsächlich gab es sogar einen Bestandteil des Prozesses, das sogenannte Change Control Board, dessen Hauptaufgabe darin bestand, Änderungsanträge abzulehnen. (Oder, genauer gesagt: »Ja, aber das wird Sie etwas kosten.«)

All das führte zu Projekten, die sich jahrelang im Status der Entwicklung befanden, bevor sie etwas vorzuweisen hatten. Als sie es dann konnten, war es zu spät oder zu teuer, um noch Änderungen vorzunehmen. Schließlich lieferten sie Software aus, die nicht das tat, was der Kunde brauchte.

Ein typischer kostspieliger Fehler

Am 3. Februar 2005 erschien Robert S. Mueller III, Direktor der wichtigsten innerstaatlichen Sicherheitsbehörde der Vereinigten Staaten (Federal Bureau of Investigation, FBI) vor einem Untersuchungsausschuss des Senats, um zu erklären, wie das FBI es geschafft hatte, 104,5 Millionen Dollar[7] zu verschwenden.

Doch wie kam es dazu, dass er sich dieser unkomfortablen Situation stellen musste? Im Juni 2001 hatte das FBI das Projekt VCF gestartet, dessen Aufgabe es war, die Verwaltungssoftware für Kriminalfälle der Behörde abzulösen. Vier Jahre später wurde das Projekt abgebrochen. Die Gesamtkosten beliefen sich bis dahin auf 170 Millionen Dollar, von denen 104,5 Millionen unwiederbringlich verloren waren.

Der Zeitplan für das Projekt VCF erzählt eine nur allzu bekannte Geschichte. Das Projekt wurde im Juni 2001 begonnen. Siebzehn Monate später, im November 2002, waren »solide Anforderungen« festgelegt worden. Ein Jahr später, im Dezember 2003, wurde die Software ausgeliefert. Allerdings entdeckte das FBI »sofort eine Reihe von Fehlern, die die Software unbrauchbar machten«. Der Dienstleister, der die Software gebaut hatte, erklärte sich bereit, die Fehler zu beheben, jedoch nur zu Kosten von zusätzlichen 56 Millionen Dollar und einem weiteren Jahr Entwicklungszeit. Letztendlich verzichtete das FBI darauf, die Fehler zu beheben, und jahrelange Arbeit wurde damit zunichtegemacht.

> Agile Teams zeigen ihren Fortschritt anhand von laufender Software und nicht anhand von Dokumenten.

Auch wenn es eine Vielzahl von agilen Methoden gibt und es bei einigen eher darum geht, einen populären Namen zu verwenden als der dahinter stehenden Philosophie zu folgen, haben sie alle etwas gemeinsam: Sie konzentrieren sich darauf, Fortschritt sichtbar zu machen und Stakeholdern die Möglichkeit zu bieten, während des Prozesses Änderungen vorzunehmen. Das scheint eine Kleinigkeit zu

7. Quellen: Die Zeugenaussagen vor dem Kongress von Robert S. Mueller am 3.2.2005 (*https://oreil.ly/GlQSa*) und von Generalinspektor Glenn Fine am 2.5.2005 (*https://oig.justice.gov/node/672*).

sein, tatsächlich ist es aber unglaublich wirkungsvoll. Es ist der Grund, weshalb wir nichts mehr von der Softwarekrise hören. Softwareprojekte sind weiterhin verspätet und Kosten werden überschritten. *Aber agile Teams zeigen ihren Fortschritt anhand von laufender Software und nicht anhand von Dokumenten*, und zwar von Beginn an. Und das ist ein gigantischer Unterschied.

Agilität ist mehr, als nur für Sichtbarkeit zu sorgen. Doch diese eine Sache reichte aus, damit alle agil sein wollten.

1.6 Warum Agilität funktioniert

Der erste durchschlagende Erfolg von Agilität war Extreme Programming (XP), eine Methode, die darauf ausgerichtet ist, Veränderungen zu begrüßen. Das ursprüngliche Motto lautete »Embrace Change«. XP vermischte eine gesunde Dosis davon, über Softwareentwicklung zu philosophieren, mit einer pragmatischen Betonung auf Veränderung. Wie im Vorwort des ersten XP-Buches zu lesen ist:

> Kurz gesagt, verspricht XP, das Projektrisiko zu verringern, die Reaktionsfähigkeit auf geschäftliche Veränderungen zu verbessern, die Produktivität während der gesamten Lebensdauer eines Systems zu erhöhen und den Spaß an der Softwareentwicklung in Teams zu steigern – und das alles gleichzeitig. Wirklich. Hört auf zu lachen. [Beck 2000] (Übersetzung aus [Wolf et al. 2005])
>
> –*Extreme Programming Explained*, 1. Auflage

Einige Leute lachten zwar, doch andere probierten es aus und stellten fest, dass XP – im Gegensatz zur gängigen Vorstellung darüber, wie Softwareentwicklung funktionieren sollte – wirklich seine Versprechen hielt. So kam es, dass sich XP trotz des Gelächters weiter verbreitete, und damit auch die Agilität.

XP war das ursprüngliche Aushängeschild von Agilität und lieferte das Gerüst für die Ideen und Terminologie, die noch heute verwendet werden. Doch die Stärke der agilen Community ist es, dass sie schon immer ein großer, bunter Haufen ist. Agilität ist nicht auf eine bestimmte Methode beschränkt, ständig erweitert sie sich, um neue Menschen und Ideen einzubeziehen. Lean Software Development, Scrum, Kanban, Lean Startup, DevOps und viele andere Methoden haben zu dem beigetragen, was wir heute als »Agilität« bezeichnen.

Werden die Ideen dieser Methoden in Kategorien aufteilt, entstehen fünf Kernkonzepte:

- *Vertrauen Sie auf Individuen.*
  Entwickeln Sie Prozesse, die auf die Bedürfnisse von Menschen Rücksicht nehmen. Legen Sie Entscheidungen in die Hände derer, die für diese Entscheidung am besten qualifiziert sind. Bauen Sie die Grundlage Ihrer Arbeit auf gesunden, kooperativen Beziehungen auf.

- *Liefern Sie Wert.*
  Holen Sie Feedback ein, experimentieren Sie und passen Sie Ihre Pläne an. Der Fokus sollte darauf liegen, Wert zu liefern. Verstehen Sie teilweise erledigte Arbeit als Kosten und nicht als Vorteil. Liefern Sie regelmäßig.
- *Vermeiden Sie Verschwendung.*
  Gehen Sie in kleinen, umkehrbaren Schritten vor. Nehmen Sie die Möglichkeit des Scheiterns in Kauf und gestalten Sie Ihre Pläne so, dass sie möglichst schnell scheitern. Maximieren Sie den Anteil nicht getaner Arbeit und geben Sie Durchsatz eine höhere Priorität als Effizienz.
- *Streben Sie nach technischer Exzellenz.*
  Ermöglichen Sie Agilität durch technische Qualität. Orientieren Sie sich beim Entwurf an dem, was bekannt ist, und nicht an dem, was angenommen wird. Starten Sie leichtgewichtig und erhöhen Sie die Komplexität nur, wenn es der tatsächliche Bedarf erfordert. Bauen Sie Systeme so, dass sie leicht erweiterbar sind – auch oder insbesondere in unvorhergesehene Richtungen.
- *Verbessern Sie Ihren Prozess.*
  Experimentieren Sie mit neuen Ideen. Funktionierende Ideen werden feinjustiert oder angepasst. Gehen Sie niemals davon aus, dass der etablierte, populäre Weg der beste für Sie ist.

> Agilität funktioniert, weil Menschen dafür sorgen, dass sie funktioniert.

Die Definition von Agilität ist im Agilen Manifest beschrieben. Dieses Manifest ist jedoch nur der Startpunkt. Agilität funktioniert, weil Menschen dafür sorgen, dass sie funktioniert. Sie nehmen die Ideen, die hinter Agilität stehen, denken sie weiter und passen sie auf die eigene Situation an. Und sie hören niemals mit dem Verbessern auf.

1.7 Warum Agilität scheitert

Agilität startete als eine Basisbewegung, vor allem getragen von Softwareentwicklern, die sich bessere Ergebnisse und eine höhere Lebensqualität wünschten. Als der Erfolg zunahm, veränderte sich die Dynamik von Agilität von den zugrunde liegenden Ideen zu einem Hype. Anstatt zu sagen: »Lasst uns bessere Ergebnisse erzielen, indem wir unsere Pläne anpassen und Individuen in den Mittelpunkt stellen«, sprachen die Führungspersonen der Unternehmen davon: »Alle reden über Agilität. Bringt mir etwas davon.«

Das Problem ist, dass es die *eine* Agilität, die gebracht werden könnte, nicht gibt. Es gibt nur eine Anhäufung von Ideen und spezifische agile Ansätze wie Extreme Programming und Scrum, die aufzeigen, wie Agilität eingesetzt werden kann. Mit der zugrunde liegenden Philosophie müssen wir uns trotzdem auseinandersetzen.

Vielen Organisationen ist diese zugrunde liegende Philosophie, Pläne anzupassen und Individuen in den Mittelpunkt zu stellen, *sehr* fremd.

Die Cargo-Kulte

Die Geschichte erzählt davon, dass in den 1940er-Jahren amerikanische Soldaten auf einer abgelegenen Insel landeten.[8] Die Inselbewohner waren noch nie zuvor mit der modernen Zivilisation in Kontakt gekommen und waren erstaunt über die Männer und Materialien, die die Streitkräfte auf die Insel brachten. Sie beobachteten, wie die Truppen eine Landebahn und einen Tower errichteten, Kopfhörer aufsetzten und daraufhin metallene Vögel mit wertvoller Fracht vom Himmel schwebten. Als die metallenen Vögel landeten, wurde die wertvolle Fracht teilweise an alle Inselbewohner verteilt, was Wohlstand und Komfort brachte.

Eines Tages verließen die Truppen die Insel und die metallenen Vögel blieben aus. Die Inselbewohner vermissten nun ihre wertvolle Fracht und bauten ihre eigene Landebahn aus geflochtenem Bambus. Sie konstruierten eine hohe Plattform, ließen ihren Häuptling darauf stehen und Kokosnüsse tragen, die so geschnitzt waren, dass sie wie Kopfhörer aussahen. Doch trotz aller Bemühungen kehrten die großen Metallvögel nie zurück.

Die Tragödie des Cargo-Kults besteht darin, an den oberflächlichen, äußerlichen Anzeichen einer Idee festzuhalten, ohne zu wissen, wie die Idee tatsächlich funktioniert. In der Geschichte haben die Inselbewohner alle Elemente von Frachtabwürfen nachgebaut – die Landebahn, den Tower und die Kopfhörer –, aber sie haben nicht verstanden, welche umfangreiche Infrastruktur benötigt wird, um die Ankunft eines Flugzeugs zu ermöglichen.

Die gleiche Tragödie ereignet sich bei Agilität. Der Wunsch der Menschen ist die Fracht von Agilität: bessere Ergebnisse, mehr Sichtbarkeit, weniger geschäftsrelevante Fehler. Doch sie verstehen die dahinter liegende Philosophie nicht und selbst wenn sie sie verstünden, würden sie ihr oft nicht zustimmen. Sie wollen Agilität kaufen, aber eine Idee können Sie nicht kaufen.

Was sie kaufen können, sind die äußerlichen Anzeichen von Agilität: Stand-up-Meetings, Storys, Werkzeuge, Zertifizierungen! Es gibt viele Dinge mit dem Aufkleber Agilität und viele Menschen, die darauf aus sind, sie Ihnen zu verkaufen. Häufig wird das Ganze als »unternehmenstauglich« verkauft, als ein Weg, um Sorgen bezüglich Veränderungen zu beschwichtigen. Unbequeme Ideen wie »adaptives Planen« und »Menschen stehen im Mittelpunkt« werden ignoriert.

8. Ich las diese Geschichte zum ersten Mal in den Texten von Richard Feynman, die auf seiner Antrittsrede 1974 bei Caltech basierten [Feynman 1974]. Die Geschichte beruht auf echten Ritualen, die in Melanesien nach dem Zweiten Weltkrieg praktiziert wurden.

Und so bekommt man einen Cargo-Kult, d.h. Aktivitäten, aber keine Ergebnisse. Es fehlt die Agilität.

»In meiner alten Firma verschwendeten sie eine Menge Arbeitsstunden in Besprechungen.«

»[Agilität] hat einem kompletten Team (über 30 Leute) den Job gekostet, weil es fast ein Jahr lang so gut wie nichts produziert hat.«

»Das Einzige, was [Agilität] bedeutet, ist, dass Entwicklerinnen im Stich gelassen werden, wenn sich das Projekt ändert … und das einen Tag vor Auslieferung.«

– Echte Kommentare über Agilität aus dem Internet

Den *Namen* Agilität finden wir überall, die zugrunde liegenden *Ideen* nicht. Agilität ist zu einem Selbstläufer geworden: Für viele ist die einzige Form von Agilität, die sie kennen, Cargo-Kult-Agilität.

Es ist an der Zeit, diesen Umstand zu ändern. Im weiteren Verlauf des Buches zeige ich Ihnen, wie Sie die Ideen von Agilität tatsächlich in die Praxis umsetzen können. Passen Sie auf die Cargo-Kult-Agilisten auf, die das Buch unterwandert haben. (Sie finden sie auch unter dem Stichwort Cargo-Kult im Index.) Sie zeigen Ihnen, was Sie nicht tun sollten.

Sind Sie bereit? Dann lassen Sie uns loslegen.

2 Wie können wir agil sein?

Wie kommen wir von der Ansammlung agiler Ideen zu tatsächlich funktionierenden agilen Teams?

Durch Praxis. Ganz viel Praxis.

2.1 Agilität praktizieren

Jedes Team hat seine Art zu arbeiten – einen *Prozess* bzw. eine *Methode* –, der es folgt, selbst wenn es nicht formell aufgeschrieben wurde. Diese Methode steht für eine zugrunde liegende Philosophie der Softwareentwicklung, obwohl diese Philosophie selten artikuliert wird und nicht notwendigerweise in sich konsistent sein muss.

Um agil zu sein, müssen wir unseren Prozess so anpassen, dass er die agile Philosophie widerspiegelt. Das ist gleichzeitig einfacher und schwieriger, als es sich anhört.

Um agil zu sein, müssen wir unseren Prozess so anpassen, dass er die agile Philosophie widerspiegelt.

Es ist einfach, weil wir in den meisten Fällen mit einer der agilen Standardmethoden anfangen können, wie beispielsweise mit der in diesem Buch vorgestellten. Und es ist schwierig, weil wir unsere Arbeitsweise anpassen müssen und das bedeutet, viele Gewohnheiten zu ändern.

Die agile Community nennt diese Gewohnheiten *Praktiken*. Ein Großteil dieses Buches ist ihnen gewidmet. Dazu zählen Dinge wie Planungsmeetings, automatische Builds und Stakeholder-Demos. Die meisten Praktiken kennen wir seit Jahrzehnten. Agile Methoden kombinieren sie jedoch auf einzigartige Weise, indem sie die Dinge betonen, die die agile Philosophie unterstützen, den Rest verwerfen und ein paar neue Ideen einbringen. Im Ergebnis entsteht ein schlankes, leistungsstarkes und sich selbst verstärkendes Paket.

Agile Praktiken dienen häufig zwei oder sogar drei Zwecken, indem sie mehrere Herausforderungen gleichzeitig lösen und sich gegenseitig auf clevere und überraschende Weise unterstützen. Wir können erst ein tieferes Verständnis einer agilen Methode erlangen, wenn wir sie bereits einige Zeit im Einsatz gesehen haben.

Es ist deswegen empfehlenswert, mit einer agilen Methode so zu starten, wie sie beispielsweise in einem Buch beschrieben wird, selbst wenn es verführerisch ist, sie gleich am Anfang an eigene Bedürfnisse anzupassen. Die Praktiken, mit denen wir uns am wenigsten auskennen, sind diejenigen, die wir am ehesten weglassen wollen. Aber es sind genau diejenigen, die wir *am meisten* brauchen, wenn wir wirklich agil werden wollen. Diese Praktiken verlangen von uns die größte Anpassung unserer Philosophie.

2.2 Der Weg zur Meisterschaft

Um die Kunst der agilen Entwicklung zu meistern, brauchen wir praktische Erfahrungen bei der Anwendung einer klar definierten agilen Methode. Starten Sie mit Ihrem Vorgehen genau so, wie es in einem Buch beschrieben ist. Setzen Sie das Vorgehen, und zwar das vollständige Vorgehen, in der Praxis um und verbringen Sie mehrere Monate damit, die Umsetzung zu verfeinern und ein besseres Verständnis über die Wirkzusammenhänge zu entwickeln. Erst dann beginnen Sie damit, die Methode anzupassen. Wählen Sie dazu einen Bereich aus, der nicht rund läuft, entwickeln Sie einen begründeten Alternativvorschlag und wiederholen Sie den Vorgang.

Dieses Buch ist genau diesem Zweck gewidmet. Es enthält eine kuratierte Sammlung agiler Praktiken, die sich in der realen Welt bewährt haben. Um diese zu nutzen, um damit die Kunst der agilen Entwicklung zu meistern – oder einfach nur, um agile Praktiken erfolgreicher einzusetzen –, gehen Sie wie folgt vor:

1. Wählen Sie eine Gruppe von agilen Ideen aus, die Sie meistern möchten. Kapitel 3 wird Ihnen bei der Entscheidung helfen.
2. Benutzen Sie so viele der korrespondierenden Praktiken wie möglich. Diese sind in Teil II bis Teil IV des Buches beschrieben. Agile Praktiken sind selbstverstärkend, deshalb ist es am besten, wenn Sie sie alle zusammen anwenden.
3. Wenden Sie die Praktiken rigoros und konsequent an. Falls eine Praktik nicht funktioniert, versuchen Sie, exakt nach der Beschreibung der Praktik vorzugehen. Teams, die sich neu mit agilen Methoden beschäftigen, tendieren dazu, die Praktiken nicht vollständig anzuwenden. Gehen Sie davon aus, dass es zwei bis drei Monate dauern wird, bis sich alle mit diesen Praktiken wohlfühlen, und weitere zwei bis sechs Monate, bis sie in Fleisch und Blut übergegangen sind.
4. Wenn Sie sich sicher sind, dass Sie die Praktiken richtig anwenden – auch hierfür sollten Sie mehrere Monate einrechnen –, beginnen Sie mit Veränderungen zu experimentieren. Jede Praktik in diesem Buch enthält einen Abschnitt, in dem erläutert wird, wie sie funktioniert und wie sie angepasst werden kann. Beobachten Sie jedes Mal, wenn Sie eine Änderung vornehmen, was passiert, und nehmen Sie weitere Verbesserungen vor.

5. Es gibt keinen abschließenden Schritt. Agile Softwareentwicklung ist ein kontinuierlicher Prozess des Lernens und der Verbesserung. Hören Sie niemals damit auf, zu üben, zu experimentieren und sich weiterzuentwickeln.

Abbildung 2–1 veranschaulicht den Prozess. Befolgen Sie zuerst die Regeln, dann können Sie die Regeln brechen und schließlich lassen Sie die Regeln hinter sich.[1]

2.3 Wie es losgeht

Ihre ersten Schritte hängen davon ab, was Sie erreichen möchten. Werden Sie in ein bestehendes agiles Team kommen, Agilität in einem oder mehreren Teams einführen oder Ihre bestehenden agilen Teams verbessern?

In ein agiles Team kommen

Wenn Sie vorhaben, einem bestehenden agilen Team beizutreten, oder einfach nur mehr darüber wissen möchten, wie Agilität in der Praxis funktioniert, dann können Sie vorblättern zu den Teilen II bis IV des Buches. Jeder Teil beginnt mit einer Geschichte »Ein Tag im Leben«, die beschreibt, wie Agilität aussehen kann. Jedes agile Team ist anders und deswegen wird das Team, in das Sie kommen, nicht genau gleich sein, aber die Geschichten geben Ihnen eine gute Vorstellung davon, was Sie erwartet.

Nach dem Lesen der Geschichten können Sie zu den Praktiken wechseln, die Sie interessieren. Jede ist so geschrieben, dass sie für sich allein als Referenzbeschreibung steht. Falls Ihr Team eine Praktik verwendet, die nicht im Inhaltsverzeichnis aufgeführt ist, dann suchen Sie im Index. Es kann sein, dass sie unter einem anderen Namen zu finden ist.

1. Diese Weiterentwicklung wurde von Alistair Cockburns Diskussionen über Shu-Ha-Ri inspiriert.

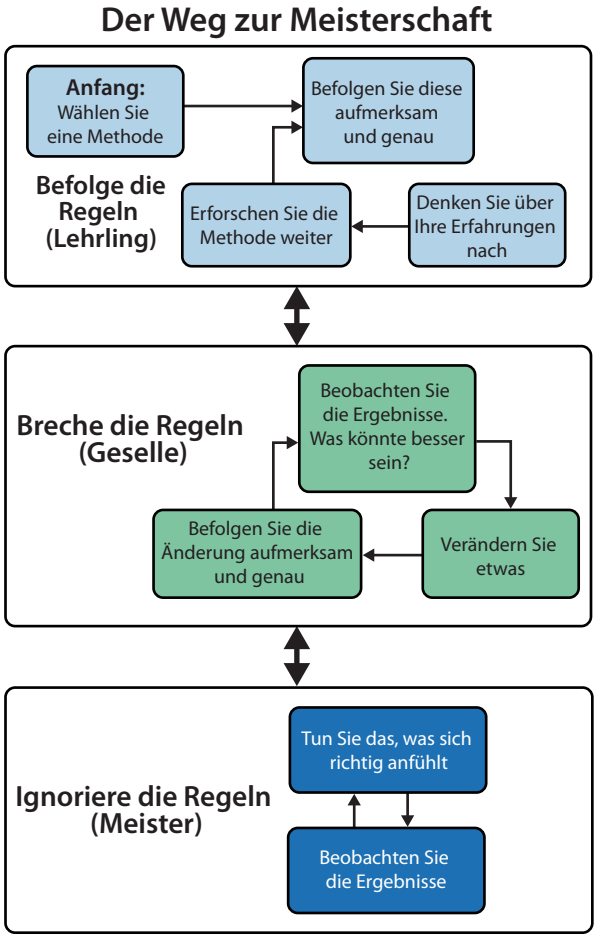

Abb. 2–1 *Der Weg zur Meisterschaft*

Agilität einführen

Wenn Sie Ihre Organisation darin unterstützen, agile Teams aufzubauen, oder diese überzeugen möchten, das zu tun, dann helfen die übrigen Kapitel des Teils I dabei, loszulegen. Benutzen Sie die folgende Checkliste, um den Überblick zu bewahren.

Stellen Sie zunächst sicher, dass Agilität für Ihre Organisation eine gute Wahl ist:

- Wählen Sie eine Herangehensweise für Agilität, die Ihre Organisation unterstützen kann (siehe Kap. 3).
- Bestimmen Sie, was Ihre Organisation braucht, damit Agilität erfolgreich eingeführt wird (siehe Kap. 4).

- Besorgen Sie sich die Erlaubnis, Agilität auszuprobieren (siehe Kap. 5).
- Falls Sie mehrere Teams haben, überlegen Sie sich, wie Sie Ihr Vorgehen skalieren (siehe Kap. 6).

In den Wochen, bevor ein Team Agilität ausprobiert:

- Legen Sie fest, wer der Coach bzw. die Coaches für das Team sein werden, und bestimmen Sie mindestens eine Person, die als Produktmanagerin des Teams fungieren wird (siehe Abschnitt »Komplettes Team« auf Seite 94).
- Sorgen Sie dafür, dass sich die Produktmanagerin des Teams mit dem Hauptsponsor des Teams und den wichtigsten Stakeholdern trifft, um eine Zielsetzung zu entwerfen (siehe Abschnitt »Zweck« auf Seite 145).
- Stellen Sie sicher, dass das Team über einen physischen oder virtuellen Teamraum verfügt (siehe Abschnitt »Teamraum« auf Seite 113).
- Vereinbaren Sie einen Termin, um eine Teamcharta zu erstellen (siehe Abschnitt »Planen Sie Ihre Chartering-Session« auf Seite 153).
- Bringen Sie das Team dazu, sich mit den neuen Praktiken zu beschäftigen. Stellen Sie Exemplare dieses Buches den Teammitgliedern zur Verfügung, sodass sie sich selbst einarbeiten können. Ermutigen Sie sie, einige Praktiken in ihrer aktuellen Arbeit auszuprobieren, und überlegen Sie, Schulungen für Praktiken, die eine Hürde darstellen, anzubieten (die Praktiken sind in Teil II bis Teil IV des Buches beschrieben).

Wenn das Team bereit ist, zu beginnen, atmen Sie tief ein und aus und:

- Lassen Sie die Teammitglieder ihre erste Woche planen (siehe Abschnitt »Ihre erste Woche« auf Seite 278).

Bestehende agile Teams verbessern

Wenn Sie bereits mit agilen Teams arbeiten und möchten, dass diese besser werden, dann hängt Ihr Vorgehen davon ab, welche Art von Verbesserung Sie erzielen möchten.

Wenn Sie an der Feinabstimmung des bestehenden Prozesses Ihres Teams interessiert sind, dann fahren Sie mit den Teilen II bis IV des Buches fort und lesen Sie die Praktiken, die Sie interessieren. Wenn Sie größere Verbesserungen anstreben, entspricht der Prozess dem der Einführung von Agilität im Team, außer dass Sie sich nur auf die Dinge zu konzentrieren brauchen, die Sie ändern möchten. Benutzen Sie die Checkliste »Agilität einführen« als Leitlinie.

Wenn Agilität in Ihrer Organisation nicht funktioniert, können Sie den »Leitfaden zur Fehlerbehebung« auf Seite 50 konsultieren.

Einzelne agile Praktiken anwenden

Agilität funktioniert am besten, wenn alles auf Agilität umgestellt wird. Doch wenn das keine Option ist, können immer noch einzelne agile Praktiken in einen existierenden Prozess eingebracht werden. Dafür gibt es gute Ausgangspunkte:

- *Daily Planning*: Wenn Sie mit häufigen Unterbrechungen zu tun haben, können Iterationen, die nur einen Tag lang sind, helfen (siehe Abschnitt »Aufgabenplanung« auf Seite 265). Nutzen Sie das Planning Game (siehe Abschnitt »Das Planning Game« auf Seite 234) und die gemessene Kapazität Ihres Teams (siehe Abschnitt »Kapazität« auf Seite 283), um ein gemeinsames Planungsmeeting am Beginn eines jeden Tages durchzuführen. Danach können alle Unterbrechungen auf das Planungsmeeting des nächsten Tages verschoben werden. Stellen Sie sicher, dass die Leute ihre eigenen Aufgaben schätzen.
- *Iterationen*: Falls es keine häufigen Unterbrechungen gibt, aber Sie trotzdem Ihre Planung verbessern möchten, lohnt es sich, wöchentliche Iterationen zu nutzen (siehe Abschnitt »Aufgabenplanung« auf Seite 265). In diesem Fall können Sie auch von täglichen Standup-Meetings (siehe Abschnitt »Standup-Meetings« auf Seite 313) und regelmäßigen Stakeholder-Demos (siehe Abschnitt »Stakeholder-Demos« auf Seite 351) profitieren. Mit allmählichem Fortschritt kann die Planung auf Karteikarten umgestellt und ein großes Schaubild genutzt werden, um die anstehende Arbeit zu visualisieren (siehe Abschnitt »Visuelles Planen« auf Seite 217).
- *Retrospektiven*: Regelmäßige Retrospektiven (siehe Abschnitt »Retrospektiven« auf Seite 398) sind ein hervorragendes Mittel für das Team, seinen Prozess anzupassen und zu verbessern. Die anderen Praktiken in Kapitel 11 können ebenfalls hilfreich sein.
- *Schnelles Feedback*: Ein schneller und automatisierter Build wird einen großen Unterschied im Alltag machen und neue Gelegenheiten für weitere Verbesserungen eröffnen (siehe Abschnitt »Keine Reibungsverluste« auf Seite 476 für weiterführende Informationen).
- *Kontinuierliche Integration*: Kontinuierliche Integration (Continuous Integration) – die Praktik, nicht das Werkzeug – reduziert nicht nur Integrationsprobleme, sondern treibt auch Verbesserungen für den Build-Prozess und die Tests voran (siehe Abschnitt »Continuous Integration« auf Seite 489 für Details).
- *Testgetriebene Entwicklung*: Obwohl testgetriebene Entwicklung (siehe Abschnitt »Testgetriebene Entwicklung« auf Seite 501) nicht so einfach umsetzbar ist wie die anderen Praktiken, ist sie doch sehr leistungsfähig. Testgetriebene Entwicklung ist die Grundlage, um die Anzahl der Fehler zu verringern, die Entwicklungsgeschwindigkeit zu erhöhen, die Möglichkeit zu verbessern,

den Quellcode im Rahmen eines Refactorings umzubauen und um technische Schulden zu reduzieren. Da es einige Zeit dauern kann, diese Praktik erfolgreich einzusetzen, ist Geduld gefragt.

Andere Praktiken aus Teil II bis Teil IV des Buches können ebenfalls nützlich sein. Agile Praktiken besitzen in hohem Maße gegenseitige Abhängigkeiten. Deswegen ist es wichtig, den Abschnitt zu den »Verwandten Themen« und den zu den »Voraussetzungen« einer jeden Praktik genau zu lesen.

Seien Sie nicht enttäuscht, wenn sich Schwierigkeiten beim Anwenden einzelner Praktiken ergeben. Es ist meistens schneller und einfacher, mehrere zusammenhängende Praktiken auszuwählen und umzusetzen. Das werden wir uns als Nächstes anschauen.

3 Wählen Sie Ihre Agilität

Es ist sinnlos, Agilität nur um der Agilität willen einzusetzen. Stattdessen sollten wir uns zwei Fragen stellen:

1. Wird Agilität uns helfen, erfolgreicher zu sein?
2. Was braucht es, um diesen Erfolg zu erreichen?

Wenn Sie diese Fragen beantworten können, werden Sie wissen, ob Agilität für Sie das Richtige ist.

Was Organisationen wertschätzen

Zum Erfolg gehört mehr als der Umsatz. Hier ist eine unvollständige Liste:

- *Das finanzielle Ergebnis steigern:* Profit, Umsatzsteigerung, Shareholder Value, Kosteneinsparungen
- *Organisatorische Ziele erreichen:* strategische Ziele, eigene Forschungsergebnisse, gemeinnützige Vorhaben
- *Die Marktposition verbessern:* Markenreichweite, Abgrenzung gegenüber dem Wettbewerb, Kundenloyalität, Attraktivität für neue Kunden
- *Kenntnisse erlangen:* strategische Informationen, analytische Informationen, Rückmeldung von Kunden

3.1 Das Agile Fluency-Modell

Im Jahr 2014 habe ich zusammen mit Diana Larsen analysiert, warum Firmen diese Unterschiede durch ihre agilen Teams erreicht sehen. Wir beide haben von Anfang an mit agilen Teams gearbeitet. Über die Jahre hinweg konnten wir erkennen, dass die Ergebnisse sich deutlich unterscheiden und dass diese sich zu Gruppen zusammenfassen lassen, die wir »Zonen« nennen. Wir haben die Beobachtungen im »Agile Fluency-Modell« zusammengefasst. Eine vereinfachte Darstellung ist in Abbildung 3–1 zu sehen [Shore & Larsen 2018].

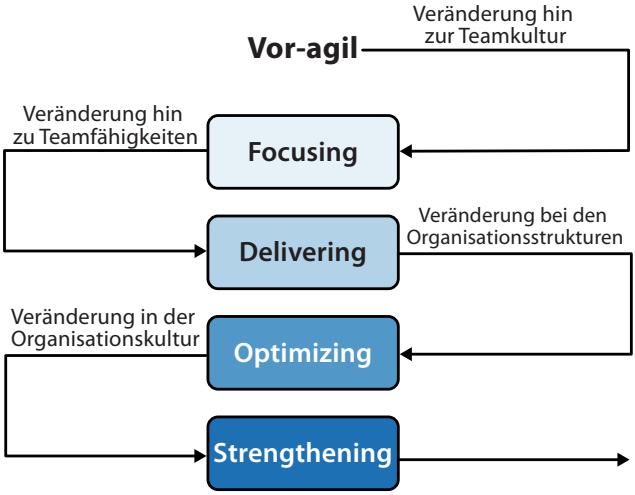

Abb. 3–1 *Ein vereinfachter Blick auf das Agile Fluency-Modell*[1]

Jede Zone ist mit gewissen *Vorteilen* verknüpft. Um diese *Vorteile* zu erzielen, braucht ein Team »*Fluency*«[2] in der Zone. Ein Team besitzt diese *Fluency*, wenn es alle Fähigkeiten, die mit dieser Zone verbunden sind, ohne nachzudenken verlässlich anwenden kann.

ANMERKUNG

Obwohl die Abbildung einen gradlinigen Weg von einer Zone zur nächsten zeigt, ist es in Wirklichkeit unübersichtlicher. Teams können Fluency in jeder Zone und in unterschiedlicher Reihenfolge erreichen. Trotzdem ist das in dieser Grafik angegebene Fortschreiten typisch.

1. © 2012–2018 James Shore und Diana Larsen. »Agile Fluency« ist eine eingetragene Marke des Agile Fluency Project LLC. Sie dürfen diese Grafik in jeder Form verwenden, solange diese Bemerkung unverändert vorhanden ist.
2. Anm. d. Übers.: Der Begriff »Agile Fluency« lehnt sich an die Idee an, eine Sprache flüssig (»fluent«) zu beherrschen. Wenn wir eine Fähigkeit mit Fluency beherrschen, dann tun wir dies quasi unbewusst. Unsere Routine in der Anwendung der Fähigkeit ermöglicht uns, sie ohne Anstrengung und mit Leichtigkeit auszuführen. Anders formuliert können wir auch sagen, dass es um routinierte Professionalität geht. Diese Routine in der Professionalität haben wir erreicht, weil wir viel Zeit in das Erlernen investiert und viel Praxis erworben haben. Deswegen sprechen wir in diesem Buch entweder von »Fluency« oder »routiniertem Vorgehen«.

Die Fähigkeiten, die für Fluency gebraucht werden, sind in den Einleitungen der Teile II bis IV dieses Buches aufgeführt. Allerdings ist Fluency nichts, was ein Team aus sich selbst heraus erreichen kann. Die umgebende *Organisation* muss in die Fluency des Teams investieren. Damit ist gemeint, dass sie nicht nur ein Lippenbekenntnis zur Agilität ablegen darf, sondern tatsächliche, bedeutsame Veränderungen vorantreiben muss, die Zeit, Geld und politisches Kapital kosten.

Die Ergebnisse, die Sie von Ihren agilen Teams erhalten, hängen davon ab, wie gut die Firma die agile Idee akzeptiert. Wenn eine Firma nicht die Ergebnisse erreicht, die sie sich von Agilität verspricht, dann liegt das typischerweise daran, dass sie die notwendigen Investitionen nicht gemacht hat. Häufig hat sie nicht einmal erkannt, was gebraucht wurde.

Treffen Sie eine bewusste Entscheidung, in Agilität zu investieren.

Treffen Sie eine bewusste Entscheidung, in Agilität zu investieren. Prüfen Sie dabei jede Zone sorgfältig. Jede Zone[3] ist mit Kosten verbunden und jede Zone bietet Vorteile. Wählen Sie die Zonen, die das richtige Kosten-Nutzen-Verhältnis für Ihre Situation aufweisen.

Sie werden es wahrscheinlich nicht schaffen, Ihre Firma davon zu überzeugen, in alle Zonen zu investieren. Das ist vollkommen in Ordnung. Im Gegensatz zu Reifegradmodellen, wie z.B. dem Capability Maturity Model Integration (CMMI), zeigt das Agile Fluency-Modell keine Abfolge von niedrigen Fähigkeiten zu hochentwickelten Fähigkeiten. Es zeigt stattdessen mehrere Wahlmöglichkeiten zu Kosten-Nutzen-Verhältnissen. Obwohl also die Abbildung einen typischen Verlauf zeigt, kann jede Zone unabhängig voneinander gewählt werden. Jede Zone hat ihren eigenen Wert.

Fluency und Reife (Maturity)

Fluency ist eine Eigenschaft, die in einem Team entsteht, nicht bei einem Individuum. Es bedeutet nicht, dass jedes Teammitglied jede Fähigkeit, die mit der Zone in Verbindung gebracht wird, besitzt. Stattdessen braucht das Team als Ganzes die Fähigkeit, immer die richtigen Personen zur richtigen Zeit einzusetzen.

→

3. Anm. d. Übers.: Jede Zone ist mit Vorteilen und mit Kosten verbunden. Dafür ist eine Busfahrt in einem Tarifsystem mit Zonen ein guter Vergleich: Wir können mit dem Bus durch mehrere Tarifzonen fahren, bis wir an der Endhaltestelle angelangt sind. Wir steigen jedoch nicht aus Prinzip erst am Endpunkt aus, sondern an der Haltestelle, die unserem Ziel am nächsten ist. Unser Bedürfnis ist es, das Ziel zu erreichen, nicht die maximale Strecke zu fahren. Entsprechend wählen wir den Fahrschein.

Jede Zone besitzt verschiedene Stufen der Reife:

1. *Lernen:* Das Team erlernt die Fähigkeiten.
2. *Bewandert:* Das Team kann eine Fähigkeit zeigen, wenn es sich darauf konzentriert.
3. *Fluent* (dt. *routiniert*): Solange das Team von einem Coach als Teil des Teams unterstützt wird, zeigt es die Fähigkeit automatisch, ohne darüber nachzudenken.
4. *Unabhängig fluent* (dt. *unabhängig routiniert):* Das Team zeigt die Fähigkeit automatisch, ohne von einem Coach oder einer entsprechenden Person unterstützt zu werden.

Focusing-Zone (auf Geschäftswert fokussierend)

Bei der *Focusing*-Zone geht es um die agilen Grundlagen: Konzentration auf Ergebnisse für das Geschäftsfeld, als Team zusammenarbeiten und den Auftrag in Besitz nehmen. Teams, die die Handlungen in dieser Zone routinemäßig erledigen, konzentrieren sich bei der Entwicklung auf ihren Hauptzweck, sie bringen immer das wertvollste Feature zuerst heraus und sind in der Lage, die Richtung zu ändern, wenn es geänderte Geschäftsanforderungen gibt. Sie sind ständig dabei, auf die wertvollste Priorität des Unternehmens zu *fokussieren*.

Für die meisten Unternehmen bedeutet dies, dass sie anders über Teams nachdenken müssen. Vor-agile Organisationen machen im Voraus Pläne, fragen Teams nach Schätzungen und erwarten Berichte darüber, wie das Team in Bezug auf die Schätzung fortschreitet. *Focusing*-Teams überarbeiten ihre Pläne regelmäßig, mindestens monatlich, und zeigen ihren Fortschritt, indem sie Ergebnisse vorführen.

Vor-agile Organisationen teilen ihre Pläne in Aufgaben auf, die sie einzelnen Personen innerhalb eines Teams zuweisen. Dabei werden die Personen danach beurteilt, wie gut die Aufgaben erledigt wurden. *Focusing*-Teams teilen die Pläne eigenständig in Aufgaben auf und entscheiden selbst, wer sich um welche Aufgabe kümmert. Sie erwarten, dass sie an ihrer Fähigkeit, Wert zu generieren, gemessen werden.

Damit ein solches Team Erfolg hat, muss Ihre Organisation diese Veränderungen mit konkreten Investitionen unterstützen. Dazu zählen die Teamstruktur, das Management und die Arbeitsumgebung (ich werde dazu im nächsten Kapitel in die Details gehen). Das ist eine Situation, in der es eine gute und eine schlechte Nachricht gibt: Die schlechte Nachricht ist, dass, wenn es ernst wird, manche Organisationen nicht bereit sind, zu investieren. Die gute Nachricht ist, dass, wenn sie es ablehnen, früh klar wird, dass sie die agile Philosophie *nicht wirklich* mittragen. Sie haben sich gerade viele Jahre Frustration und Kummer beim Austreiben von Cargo-Kult erspart.

Wenn Sie es schaffen, Akzeptanz zu erzielen, werden ca. zwei bis sechs Monate gezielte Anstrengungen benötigt, um ein Team dahin zu bringen, die *Focusing*-Zone wie im Schlaf zu erledigen. Mit der richtigen Unterstützung kann es seinen vorherigen Leistungsgrad innerhalb von ein bis vier Monaten übertreffen[4]. Teil II des Buches zeigt die Praktiken, die dafür gebraucht werden.

Delivering-Zone (immer auslieferungsfähig)

Agile Teams können ihre Pläne jederzeit ändern. Bei den meisten Teams würde das dazu führen, dass die Qualität ihres Quellcodes sich langsam vermindert. Sie verlieren allmählich ihre Fähigkeit, kostengünstige Änderungen zu machen. Irgendwann werden sie sagen, dass sie die Software wegwerfen und neuschreiben müssen – ein teurer und verschwenderischer Vorschlag.

Delivering-Teams verhindern diese Entwicklung durch technische Exzellenz. Sie gestalten ihren Quellcode so, dass er auf regelmäßige Änderungen reagieren kann. Sie halten die Codequalität hoch, sodass sie Zeitverschwendung durch Fehlersuche vermeiden. Sie verbessern ihren Produktionszyklus, sodass Releases schmerzfrei sind und der Betrieb handhabbar bleibt. Sie sind in der Lage, Software mit einer geringen Fehlerzahl zuverlässig *herauszugeben*, wann immer es geschäftlich sinnvoll ist.

Um diese Ergebnisse zu erzielen, erfordert es eine substanzielle Investition in die technischen Fähigkeiten der Teammitglieder. Gleichzeitig müssen strukturelle Veränderungen unternommen werden, um Personen mit Test- und Betriebsfähigkeiten in das Team zu integrieren.

Falls Ihre Firma diese Investitionen tätigt, kann die *Delivering* Fluency innerhalb von drei bis 24 Monaten bei einem Team erreicht werden und Sie werden erste Leistungsverbesserungen innerhalb von zwei bis sechs Monaten beobachten können. Die genaue Zeit, die jedes Team dafür braucht, wird davon abhängen, wie die Qualität des existierenden Quellcodes ist und wie viel technisches Coaching das Team erhält. Teil III des Buches liefert die dazu notwendigen Praktiken.

Optimizing-Zone (auf Geschäftswert optimierend)

Die meisten Firmen geben sich mit der *Focusing* oder *Delivering* Fluency zufrieden. Aber Agilität hat mehr Potenzial. In ihrer vollen Pracht ist Agilität eine Welt, in der Teams als Reaktion auf Marktbedingungen wirbeln und tanzen. Sie experimentieren und lernen, sie erschließen sich neue Märkte und stechen den Wettbewerb aus.

4. Bei den Zeitangaben in diesem Kapitel handelt es sich um Schätzungen, die auf meinen Erfahrungen beruhen. Ihre Erfahrungen können davon abweichen.

Optimizing-Teams erreichen diesen Grad an Agilität. Sie verstehen, was der Markt von ihnen erwartet, was Ihr Betrieb braucht und wie diese beiden zusammengebracht werden können. Oder sie wissen – wie es bei Start-ups üblich ist –, was sie lernen müssen und wie sie dieses Lernen sicherstellen können. Sie *optimieren* ihre Pläne zum Produkt regelmäßig, um den größtmöglichen Geschäftswert zu erzielen.

Dieses Vorgehen erfordert ein Anpassen der Organisationsstruktur. Um optimale Pläne zu erstellen, braucht es dauerhafte Aufmerksamkeit von Menschen, die sich intensiv mit dem Geschäftsfeld auseinandersetzen und Expertise im Produktbereich besitzen. In letzter Konsequenz bedeutet das, dass Produktexperten und Marktexperten dauerhaft in ein Entwicklungsteam gehen. Es bedeutet auch, dass den Teams volle Verantwortung für ihr Produktbudget und den Produktplan übergeben wird.

Diese strukturellen Veränderungen benötigen eine Genehmigung auf höchster Ebene in der Organisation. Es kann schwierig sein, diese zu bekommen. Typischerweise brauchen Teams mindestens ein Jahr, um das Vertrauen durch *Delivering* Fluency aufzubauen, bevor sie die Erlaubnis für diese Investitionen bekommen. Ist das geschehen, braucht es ca. drei bis neun weitere Monate, um *Optimizing* Fluency aufzubauen. Sie sehen erste Verbesserungen innerhalb der ersten drei Monate. Allerdings ist *Optimizing* ein niemals endender Prozess von Experimenten, Lernen und Entdeckung. Teil IV des Buches beschreibt, wie Sie ihn beginnen.

Strengthening-Zone (immer die richtige Organisationsstruktur)

Es gibt noch eine letzte Zone im Agile Fluency-Modell. Diese ist hochgradig spekulativ: Sie ist eine mögliche Zukunft der Agilität. Sie ist gleichzeitig nur für Organisationen geeignet, die auf dem neuesten Stand von Managementtheorie und -anwendung sein möchten. Damit liegt dies nicht im Fokus dieses Buches. Kurz gesagt geht es bei der *Strengthening*-Zone darum, die Gesamtheit der Teamerkenntnisse in organisationale Verbesserungen umzumünzen. Wenn Sie mehr dazu erfahren möchten, lesen Sie in Kapitel 19 weiter.

Zusammenfassung der Agile Fluency-Zonen

Focusing:

- *Hauptvorteil:* Fokus auf die Prioritäten des Geschäfts; Sichtbarkeit der Teamarbeit; Fähigkeit, die Richtung anzupassen
- *Investitionen*: Teamstruktur, Management; Arbeitsumgebung
- *Grober Zeitverlauf:* Leistungseinbruch von 1 bis 4 Monaten; 2 bis 6 Monate bis zum Erreichen dieser Fähigkeit

→

Delivering:

- *Hauptvorteil:* geringe Anzahl von Fehlern und hohe Produktivität; technische Langlebigkeit
- *Investitionen:* Entwicklungsfähigkeiten; Integration von Testen und Betrieb
- *Grober Zeitverlauf:* Leistungseinbruch von 2 bis 6 Monaten; 3 bis 24 Monate bis zum Erreichen dieser Fähigkeit

Optimizing:

- *Hauptvorteil:* Releases mit höherem Geschäftswert und besseren Produktentscheidungen
- *Investitionen*: integriertes Produktmanagement; Teamverantwortung für das Budget und die Planung
- *Grober Zeitverlauf:* Leistungseinbruch von 1 bis 3 Monaten; 3 bis 9 Monate bis zum Erreichen dieser Fähigkeit

3.2 Wählen Sie Ihre Zonen

Welche Fluency-Zone sollte Ihr Team anstreben? Das hängt davon ab, welche Zonen Ihre Organisation unterstützen kann. Auf einem leeren Blatt sind *Focusing*, *Delivering* und *Optimizing* alle zusammen die beste Wahl. Die Kombination von allen dreien liefert die besten Ergebnisse und die unverfälschteste Umsetzung der agilen Ideen.

Aber alle drei Zonen zusammen benötigen auch die größten Investitionen. Wenn sich diese Investitionen nicht rechtfertigen lassen, wird es schwer werden, die erforderliche Unterstützung zu bekommen. Und ohne die notwendigen Investitionen wird es dem Team schwerfallen, Fluency zu erreichen. Es fallen die Kosten für das Lernen an, ohne alle Vorteile zu erhalten. Es kann sogar sein, dass sich zunächst *schlechtere* Ergebnisse einstellen als zum jetzigen Zeitpunkt.

Mit anderen Worten: Wählen Sie nur die Zonen, die Ihr Unternehmen *braucht* und *bereit ist, zu bezahlen*.

Also, welche Zonen sollten Sie auswählen?

- Jedes agile Team braucht die Fähigkeit, die Handlungen in der *Focusing*-Zone ohne nachzudenken auszuführen. Das ist die Grundlage. Wenn Ihre Firma nicht wenigstens in das Beherrschen der *Focusing*-Zone investieren kann, passt Agilität wahrscheinlich nicht zu ihr. Allerdings könnte es sein, dass ein Start mit den Fähigkeiten der *Delivering*-Zone die Bereitschaft dafür erhöht.
- Die Fähigkeiten der *Delivering*-Zone reduzieren Kosten und erhöhen die Entwicklungsgeschwindigkeit. Ohne diese wird der Quellcode irgendwann in

technischen Schulden ersticken. Damit wird die *Delivering*-Zone für die meisten Teams zu einer Selbstverständlichkeit. Aber ungeachtet dessen sind manche Organisationen noch nicht bereit dafür, die großen Investitionen in Lernen und in Codequalität zu stemmen, die die *Delivering*-Zone erfordert. Deswegen ist es in Ordnung, *zuerst* mit der routinemäßigen Ausführung der Handlungen in der *Focusing*-Zone zu starten, Erfolge zu zeigen und dann damit weitere Investitionen zu begründen.

- Durch die Beherrschung der Fähigkeiten der *Optimizing*-Zone ist dies die Zone, in der Agilität am intensivsten strahlt. Und gleichzeitig ist es ein großer Anspruch. Am besten wird Vertrauen dadurch aufgebaut, dass ein Team Routine in den Handlungen in den Zonen *Focusing* und *Delivering* zeigt. Dann kann es schrittweise mehr Verantwortung übernehmen. Wenn die Organisation allerdings bereits eine Kultur etabliert hat, bei der Entscheidungen an cross-funktionale Teams delegiert werden, wie wir das häufig bei Start-ups sehen, dann wird die routinemäßige Ausführung der Handlungen in der *Optimizing*-Zone großartige Ergebnisse liefern.

Zu den Details jeder Zone und den jeweiligen Vorteilen lesen Sie mehr in den Einleitungen von Teil II, Teil III und Teil IV des Buches. Für eine detaillierte Aufstellung der notwendigen Investitionen lesen Sie den Abschnitt »Übersicht der Investitionen« auf Seite 32. Falls Sie nicht sicher sind, welche Zonen Sie wählen sollten, fangen Sie mit *Focusing* und *Delivering* an.

Unabhängig davon, welche Zonen Sie wählen, sollten Sie in das Erlernen aller Praktiken gleichzeitig investieren. Die Praktiken der späteren Zonen verbessern auch die vorhergehenden Zonen. Deswegen erzielen Sie bessere Ergebnisse, wenn Sie sich alles gleichzeitig aneignen, anstatt es nacheinander anzugehen. Aber es ist auch in Ordnung, wenn Sie nicht in jede Zone, die Sie erreichen wollen, gleichzeitig investieren können. Es dauert dann zwar länger, aber man kann sich schrittweise hocharbeiten.

Wenn Sie wissen, welche Zonen Sie anstreben, ist die Zeit gekommen, sich die Investitionen detaillierter anzuschauen. Diese untersuchen wir im nächsten Kapitel.

4 In Agilität investieren

Wie wir im vorhergehenden Kapitel gesehen haben, muss Ihre Organisation in die dahinterliegende Philosophie von Agilität investieren, damit Ihre Teams die **Vorteile** von Agilität nutzen können. Es geht dabei nicht nur darum, Geld auszugeben, das wäre verhältnismäßig leicht. Es geht eher darum, sinnvolle Änderungen an der Struktur, dem System und der Verhaltensweise vorzunehmen.

Wenn das nach einer Menge Arbeit klingt … nun ja, das liegt daran, dass es eine Menge Arbeit ist. Sind diese Investments wirklich so wichtig?

Ja, das sind sie.

In Agilität zu investieren ist wichtig, denn wir investieren darin, die eigenen Einschränkungen zu verändern. Teams werden meistens nicht von ihren Prozessen zurückgehalten; das, was sie zurückhält, sind die Einschränkungen, denen sie unterliegen. Wenn Sie die Investitionen vornehmen und die Praktiken vernachlässigen, dann werden sich Teams wahrscheinlich trotzdem verbessern. Wenden Sie die Praktiken an, aber tätigen keine Investitionen, dann führt das dazu, dass es Teams schwer haben werden.

Teams werden meistens nicht von ihren Prozessen zurückgehalten; das, was sie zurückhält, sind die Einschränkungen, denen sie unterliegen.

Martin Fowler sagte einmal:[1]

> Ich sehe eine verblüffende Ähnlichkeit zwischen DHH [David Heinemeier Hansson, Gründer von Ruby on Rails] und Kent Beck [Erfinder des Extreme Programming]. Wird beiden eine Welt mit Einschränkungen vorgesetzt, erachten sie diese für unwesentlich und erschaffen eine Welt ohne sie … sie spicken sie mit intellektuellem Dynamit und machen weiter. Aus diesem Grund können sie Dinge erschaffen, die einen Ruck in ihrem Berufsfeld bewirken, wie Extreme Programming oder Rails.
>
> – Martin Fowler

1. Auszug aus Fowlers Artikel »EnterpriseRails« (*https://martinfowler.com/bliki/EnterpriseRails.html*).

Machen Sie die Investitionen. Sie sind der Schlüssel zu erfolgreicher Agilität.

Die folgenden Abschnitte beschreiben, welche Investitionen Ihre Teams von Ihrer Organisation benötigen. Sie sind möglicherweise nicht in der Lage, alle zu tätigen. Für diesen Fall zeige ich Alternativen auf. Diese Alternativen haben jedoch den Nachteil, dass sie weniger Wirkung zeigen. Bemühen Sie sich also, so viele Investitionen wie möglich durchzuführen. Ich habe im Buch nur die wichtigsten aufgenommen.

Übersicht der Investitionen

Für alle agilen Teams:

- Holen Sie die Zustimmung des Managements, von Teams und wichtigen Stakeholdern ein, wie in Kapitel 5 beschrieben.
- Erstellen Sie langlebige, cross-funktionale Teams und weisen Sie Personen nur ihrem Team zu (siehe Abschnitt »Wählen oder bilden Sie agile Teams« auf Seite 37).
- Stellen Sie sicher, dass jedes Team einen Coach hat, der den Teammitgliedern helfen kann, ein effektives, eingespieltes Team zu sein (siehe Abschnitt »Wählen Sie agile Coaches aus« auf Seite 39).
- Weisen Sie die Arbeit Teams und nicht Individuen zu. Erwarten Sie von Teams, dass sie ihren eigenen Ansatz zur täglichen Planung und Verteilung von Aufgaben haben (siehe Abschnitt »Delegieren Sie Befugnisse und Verantwortung an Teams« auf Seite 40).
- Anstatt Individuen und Aufgaben zu managen, legen Sie den Managementschwerpunkt auf Teamebene und das System (siehe Abschnitt »Passen Sie den Managementstil auf Teamebene an« auf Seite 43).
- Stellen Sie einen physischen oder virtuellen Teamraum für jedes Team bereit (siehe Abschnitt »Richten Sie Teamräume ein« auf Seite 43).
- Wählen Sie für den ersten Versuch jedes Teams ein wichtiges, aber nicht dringendes Ziel, das sich gut zum Lernen von Agilität eignet (siehe Abschnitt »Etablieren Sie für jedes Team ein lernfreundliches Ziel« auf Seite 44).
- Ersetzen Sie Steuerungsrichtlinien aus dem Wasserfallmodell durch agile Richtlinien (siehe Abschnitt »Ersetzen Sie Steuerungsansätze nach dem Wasserfallmodell« auf Seite 45).
- Entfernen, überarbeiten oder umgehen Sie Vorgaben aus der Personalabteilung, die effektive Teamarbeit behindern (siehe Abschnitt »Passen Sie nachteilige Vorgaben aus der Personalabteilung an« auf Seite 47).

→

Focusing-Teams:

- Stellen Sie sich auf einen Leistungseinbruch ein, der bei jedem Team ein bis vier Monate andauert (siehe Abschnitt »Ermöglichen Sie Zeit für das Lernen« auf Seite 34).
- Beziehen Sie Personen in jedes Team ein, die über die Fähigkeit verfügen, die Sicht von Benutzerinnen und Kunden einzunehmen (siehe Abschnitt »Wählen oder bilden Sie agile Teams« auf Seite 37).
- Stellen Sie sicher, dass sich in jedem Team eine Person befindet, die entscheidet, woran das Team arbeiten sollte. Alternativ sollte das Team regelmäßig Zugriff auf diese Person haben (siehe Abschnitt »Wählen oder bilden Sie agile Teams« auf Seite 37).
- Stellen Sie sicher, dass jedes Team einen Coach hat, der die Praktiken aus der *Focusing*-Zone vermitteln kann (siehe Abschnitt »Wählen Sie agile Coaches aus« auf Seite 39).
- Stellen Sie sicher, dass jedes Team Zugriff auf seine Stakeholder hat oder auf Personen, die diese repräsentieren können (siehe Abschnitt »Delegieren Sie Befugnisse und Verantwortung an Teams« auf Seite 40).

Delivering-Teams:

- Stellen Sie sich auf einen Leistungseinbruch ein, der bei jedem Team zwei bis sechs Monate andauert (siehe Abschnitt »Ermöglichen Sie Zeit für das Lernen« auf Seite 34).
- Integrieren Sie alle benötigten Fähigkeiten für die Entwicklung, beispielsweise Test und Betrieb, in jedes Teams (siehe Abschnitt »Wählen oder bilden Sie agile Teams« auf Seite 37).
- Stellen Sie sicher, dass jedes Team einen Coach hat, der die Praktiken aus der *Delivering*-Zone vermitteln kann (siehe Abschnitt »Wählen Sie agile Coaches aus« auf Seite 39).
- Stellen Sie sicher, dass jedes Team Kontrolle über seine Entwicklungs-, Build-, Test- und Freigabeprozesse hat (siehe Abschnitt »Delegieren Sie Befugnisse und Verantwortung an Teams« auf Seite 40).
- Bestimmen Sie als erste Aufgabe eines Teams ein Ziel, das »auf der grünen Wiese« startet. Es sei denn, der Coach des Teams hält das nicht für notwendig (siehe Abschnitt »Etablieren Sie für jedes Team ein lernfreundliches Ziel« auf Seite 44).
- Sprechen Sie Sicherheitsbedenken an, die eine kollaborative Entwicklung verhindern (siehe Abschnitt »Sprechen Sie Sicherheitsbedenken an« auf Seite 48).

→

Optimizing-Teams:

- Stellen Sie sich auf einen Leistungseinbruch ein, der bei jedem Team ein bis drei Monate andauert (siehe Abschnitt »Ermöglichen Sie Zeit für das Lernen« auf Seite 34).
- Stellen Sie sicher, dass es in jedem Team Personen mit Expertise in Bezug auf das Geschäft, den Markt und das Produkt gibt (siehe Abschnitt »Wählen oder bilden Sie agile Teams« auf Seite 37).
- Stellen Sie sicher, dass jedes Team einen Coach hat, der die Praktiken aus der *Optimizing*-Zone vermitteln kann (siehe Abschnitt »Wählen Sie agile Coaches aus« auf Seite 39).
- Geben Sie jedem Team die Verantwortung über sein Budget, seine Pläne und seine Ergebnisse (siehe Abschnitt »Delegieren Sie Befugnisse und Verantwortung an Teams« auf Seite 40).

4.1 Ermöglichen Sie Zeit für das Lernen

Veränderungen sind störend und neue Ideen benötigen Zeit zum Lernen. Agile Praktiken zu erlernen, sorgt dafür, dass Teams *zunächst* langsamer werden.

Um wie viel langsamer das Team wird, lässt sich nicht objektiv sagen, da es keine objektive Messmethode gibt, die Softwareproduktivität misst [Fowler 2003]. Aber aus Erfahrungen lassen sich zu Beginn Werte von 10 bis 20 % schätzen. In dem Maße, wie die agilen Fertigkeiten zunehmen, wird auch die Leistungsfähigkeit des Teams steigen. Die Leistungsfähigkeit wird bis zu dem Punkt steigen, an dem das Team routinemäßig seine benötigten Fertigkeiten ausübt. Anschließend wird der Anstieg allmählich abflachen, wie es in Abbildung 4–1 dargestellt wird. Das entstehende Bild wird *J-Kurve* genannt und alle signifikanten Änderungen haben sie gemein. In Kapitel 5 werden wir uns näher mit dem Thema Wandel befassen.

Normalerweise wird sich die investierte Zeit innerhalb eines Jahres rentieren. Die Länge des initialen Einbruchs hängt davon ab, welche Fluency-Zonen, die im vorherigen Kapitel beschrieben wurden, das Team anstrebt. Zur Erinnerung:

- *Focusing*-Zone: 1–4 Monate
- *Delivering*-Zone: 2–6 Monate
- *Optimizing*-Zone: 1–3 Monate

Diese Zeitabschnitte überschneiden sich. So wird zum Beispiel ein Team, das Fähigkeiten aus der *Focusing*- und der *Delivering*-Zone zusammen lernt, einen Einbruch der Leistungsfähigkeit von zwei bis sechs Monaten haben. Im Gegensatz dazu hätte ein Team, das zuerst Fähigkeiten aus der *Focusing*-Zone erlernt, um sich später den Fähigkeiten aus der *Delivering*-Zone zu widmen, zwei Einbrü-

che der Leistungsfähigkeit. Einen von einem bis zu vier Monaten für die Fähigkeiten aus der *Focusing*-Zone und einen weiteren Einbruch von zwei bis sechs Monaten für die Fertigkeiten aus der *Delivering*-Zone.

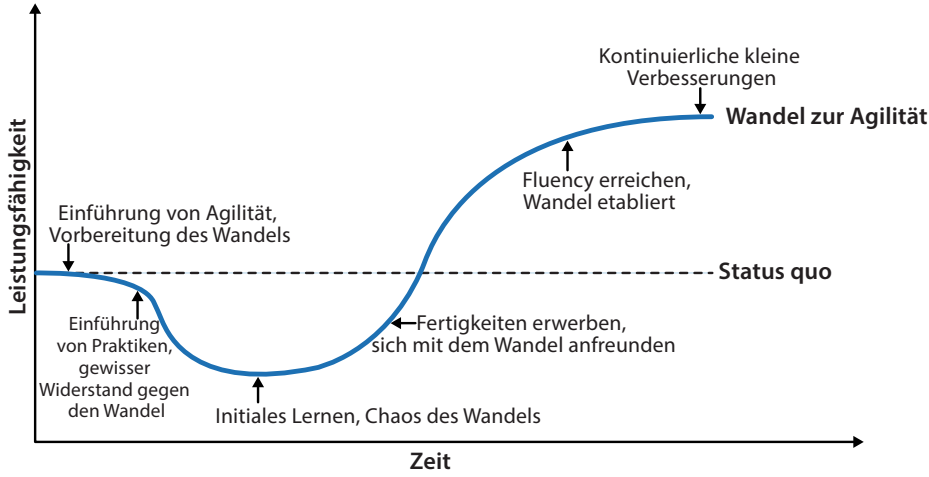

Abb. 4–1 *Agile Leistungsfähigkeit im Laufe der Zeit*

Die Leistungsfähigkeit agiler Teams verändert sich auch noch in anderer Weise. Agile Teams konzentrieren sich darauf, Funktionen komplett fertigzustellen, bevor sie sich der nächsten Funktion widmen. Das gilt insbesondere für Teams mit den Fertigkeiten der *Delivering*-Zone, die von Anfang an ein Produkt mit Qualität bauen, statt am Ende Fehler zu beheben. Dadurch verbessern sich Durchsatz und Leistung. Ironischerweise fühlt es sich für Menschen wie eine Verlangsamung an, die es bisher gewohnt waren, mehrere Funktionen gleichzeitig in Bearbeitung zu sehen.

Am Ende sind Stakeholder, insbesondere im ersten Jahr, von der Geschwindigkeit agiler Entwicklung frustriert. Sie werden in dieser Zeit mit drei Faktoren auf einmal konfrontiert: Einer tatsächlichen Verzögerung durch das Lernen, einer gefühlten Verzögerung durch den Fokus auf die Fertigstellung von Aufgaben und Kosten, die daraus entstehen, Sachen zu reparieren, die in der Zeit vor der agilen Arbeitsweise entstanden sind und fertig sein sollten, es aber doch nicht sind.

Diese Frustration kann dazu führen, dass Teams von dem Erlernen agiler Methoden abgehalten werden, weil sie sich ausschließlich auf das Bereitstellen von Software konzentrieren. Das ist für alle Beteiligten kontraproduktiv. Die Teams fühlen sich hin- und hergerissen und sind dadurch wahrscheinlich frustriert. Für das Unternehmen besteht die Gefahr, die bisher getätigten Investitionen vergeudet zu haben. Bevor Teams mit ihrer Reise in Richtung Agilität beginnen, sollte daher sichergestellt sein, dass Manager und Stakeholder mit der Verzögerung im ersten Jahr einverstanden sind.

Um den Leistungsabfall im ersten Jahr zu reduzieren, kann Ihr Unternehmen Geld gegen Zeit eintauschen und Leute engagieren, die den Teams helfen. Dies wird die Verzögerung im ersten Jahr nicht eliminieren, aber der Leistungsabfall wird kürzer und flacher. Das Angebot an Hilfe ist vielfältig: gelegentliches Mentoring, Schulungen, Unterstützung bei Prozessgestaltung und -umsetzung sowie Vollzeit-Coaching. Die effektivste Hilfe erhalten Sie mit der Anstellung von Personen mit Praxiserfahrung, die Teams in Vollzeit coachen.

Bei der Entscheidung, wen Sie für diese Unterstützung einstellen möchten, sollten Sie die unzähligen agilen Zertifizierungsprogramme ignorieren. Bei vielen von ihnen geht es bloß ums Geldverdienen. Die meisten zeigen einzig eine Fähigkeit auf, nämlich ein paar Tage auf einem Stuhl zu sitzen. Einige Zertifikate werden mit exzellenten Schulungen in Verbindung gebracht, das ist aber abhängig vom jeweiligen Trainer und nicht vom Zertifikat. Schulungen sollten daher unabhängig vom Zertifikat gewählt werden. Das Gleiche gilt für die Auswahl von Beraterinnen oder Coaches. Statt auf Zertifikate zu schauen, bietet Ihnen vielleicht das eigene Netzwerk Empfehlungen. Ansonsten hilft es, öffentlich verfügbares Material zu sichten und Referenzen zu prüfen.

Wenn Sie die in diesem Buch beschriebenen Praktiken anwenden, werden Sie wahrscheinlich auf Probleme und Herausforderungen stoßen, die für Ihre Situation spezifisch sind. Stellen Sie sicher, dass Sie eine erfahrene Person in Ihrem Umfeld haben, an die Sie sich bei Fragen wenden können. Das muss nicht unbedingt etwas kosten. Es kann sich um einen Kollegen in einem Unternehmen handeln, der diesen Prozess schon einmal durchgeführt hat, und auch eine lokale User Group oder ein Onlineforum sind gute Möglichkeiten.

Wenn keine Zeit für das Lernen bleibt …

Sie können den Leistungsabfall weniger bemerkbar gestalten, wenn Sie sich ausschließlich auf die *Focusing*-Zone konzentrieren. Das führt zu einem sanften Start der Agilität mit dem Fokus darauf, Aufgaben zu erledigen, birgt aber die Gefahr höherer Gesamtkosten.

Wenn Ihr Unternehmen keinen Leistungsabfall hinnehmen kann, ist es auch kein guter Zeitpunkt, um in einen Wandel zu investieren. Wenn es diesen Zeitpunkt niemals geben wird, ist das ein großes Warnsignal. Sie müssen die Unternehmensleitung davon überzeugen, für einen solchen Wandel Zeit einzuräumen, bevor Sie Ihr Vorgehen fortsetzen.

Wenn es für Hilfe kein Budget gibt …

Mit diesem Buch, den vielen online und frei verfügbaren Ressourcen und dem Einsatz für das Lernen kann Ihr Team sich alles selbst beibringen, was es wissen muss. Hilfe von außen ist, nun ja, hilfreich, aber nicht zwingend erforderlich.

4.2 Wählen oder bilden Sie agile Teams

Ich kann den Stellenwert von Teams in einer agilen Organisation gar nicht genug betonen. Die meisten Unternehmen betrachten die Menschen als ihre für die Arbeit grundlegende »Personalressource«. Bei Agilität sind Teams diese Ressource.

Ihre Organisation muss in Teams mit den folgenden Merkmalen investieren:

- *Cross-funktional*. Die Teammitglieder verfügen gemeinsam über das gesamte benötigte Fachwissen, um ihren Zweck zu erfüllen.
- *Vollständig zugeordnet*. Fachleute können hin und wieder zur Unterstützung hinzukommen. Aber das Kernteam widmet sich ausschließlich den Aufgaben des Teams.
- *Gemeinschaftlich*. Teammitglieder pflegen freundschaftliche, kollegiale Beziehungen und arbeiten eng zusammen.
- *Langlebig*. Es kann Monate dauern, bis das Team herausgefunden hat, wie es am effektivsten zusammenarbeiten kann. Daher sollten Teams so lange wie möglich zusammenbleiben.

Die Größe und Zusammensetzung von Teams hängen davon ab, welche Fluency-Zone sie anstreben. Der Abschnitt »Komplettes Team« auf Seite 94 beschreibt es detailliert, hier eine kurze Zusammenfassung:

- *Focusing*-Teams konzentrieren sich auf das Erreichen von Geschäftsergebnissen. Sie benötigen die Fähigkeit, sich in die Perspektive von Benutzerinnen und Kunden zu versetzen, um das Verhalten der Software zu bestimmen. Ist der Fokus des Teams auf die Benutzerinnen der Software ausgerichtet, sind Personen mit Fähigkeiten in UI/UX (User Interface, User Experience) erforderlich. Teams brauchen auch einen Weg, um zu bestimmen, woran sie als Nächstes arbeiten wollen. Obwohl es am besten ist, wenn das Team mit Personen besetzt ist, die die Fähigkeit und Autorität hierfür selbst haben, können Teammitglieder auch mit jemandem außerhalb des Teams zusammenarbeiten.
- *Delivering*-Teams übernehmen Ende-zu-Ende-Verantwortung für ihre Software. Sie benötigen alle Fähigkeiten, um ihr Produkt zu bauen und auszuliefern. Verantwortlichkeiten, die zuvor an andere Teams abgegeben wurden, müssen zurück in das Team geholt werden. Das umfasst Build-Management, Datenarchitektur und -verwaltung, Test und Betrieb.
- *Optimizing*-Teams übernehmen Verantwortung für den umfassenden Geschäftserfolg ihres Produkts. Sie verantworten außerdem die Koordination mit Stakeholdern und entscheiden über Prioritäten, die ihr Produkt betreffen. Sie brauchen Teammitglieder, die Expertise in Bezug auf das Geschäft, den Markt und das Produkt haben.

Vielleicht verfügen Sie bereits über Teams, die diesen Anforderungen entsprechen. Wenn Sie neue agile Teams aufsetzen, verwenden Sie die folgenden Schritte. In jedem Fall sollten Sie daran denken, sich die Zustimmung des Teams, wie in Abschnitt »Besorgen Sie sich die Zustimmung des Teams« auf Seite 67 beschrieben, einzuholen.

1. Bestimmen Sie den Zweck jedes Teams (siehe Abschnitt »Zweck« auf Seite 145).
2. Entscheiden Sie, wie viele Menschen jedes Team umfassen soll, basierend auf dem Wert des Zwecks und vorbehaltlich der beschriebenen Größenbegrenzungen aus dem Abschnitt »Komplettes Team« auf Seite 94.
3. Bestimmen Sie, welche Fähigkeiten das Team benötigt.
4. Wählen Sie Personen aus, die über die benötigten Fähigkeiten verfügen, wahrscheinlich gut zusammenarbeiten und bereit sind, Agilität auszuprobieren.

Wenn Sie eine Reorganisation mit vielen Teams vorhaben, gibt es die Möglichkeit einer Selbstzuordnung zu Teams. Die Zusammensetzung von Teams auf diese Art zu bestimmen, ist überraschend effektiv darin, höchst produktive Teams zu bilden, die sich auf die Zusammenarbeit freuen. Das Buch *Creating Great Teams: How Self-Selection Lets People Excel* [Mamoli & Mole 2015] beschreibt, wie dieses Vorgehen funktioniert.

Wenn Sie Leute nicht ihren Teams zuordnen können …

Die Arbeit nach agilen Methoden erfordert eine enge Zusammenarbeit und diese funktioniert nicht gut, wenn Einzelne nicht verfügbar sind. Eine gelegentliche externe Abhängigkeit ist in Ordnung, doch wenn es für Sie nicht möglich ist, Personen dediziert Teams zuzuordnen, werden agile Vorgehensweisen voraussichtlich nicht funktionieren.

Wenn Teammitglieder nicht miteinander auskommen …

Zu Beginn ist es für ein neues Team normal, dass es eine schwierige Phase durchläuft. Dabei findet das Team heraus, wie es zusammenarbeitet. Seien Sie bei Startschwierigkeiten also nicht besorgt. Der Coach und das Management können bei der Mediation von Konflikten unterstützen. Lesen Sie im Abschnitt »Teamdynamik« auf Seite 408 nach für mehr Informationen.

Wenn Sie keine langlebigen Teams bilden können …

Es ist eine Verschwendung, hochperformante Teams aufzulösen. Doch es wird nicht verhindern, dass Ihre Teams agil sind.

Wenn Sie nicht das nötige Fachwissen in Bezug auf Ihr Geschäft, Ihre Kunden oder die Benutzerinnen erhalten …

Optimizing-Teams benötigen mindestens eine Person im Team mit Produktmanagementkenntnissen. Dabei muss es sich nicht um eine traditionelle Rolle mit dem Titel Produktmanager handeln. Manchmal eignen sich Leute aus der Entwicklung, die schon längere Zeit im Unternehmen sind und das eigene Produkt und den Markt kennen, besser als jede andere Person. Wenn Sie so eine Person gewinnen können, kann es losgehen.

Wenn Ihr Team nicht Fluency in der *Optimizing*-Zone anstrebt, brauchen Sie keinen Produktmanager direkt im Team. Es sollte aber eine Person mit diesem Fachwissen geben, die eng mit dem Team zusammenarbeitet, und Sie benötigen immer noch Teammitglieder, die die Perspektiven von Kundinnen und Benutzern einbringen können.

Die Fachseite einzubinden hat enorme Auswirkungen auf den Erfolg des Teams. Es ist einer der Unterschiede, der Agilität von vorherigen Vorgehensweisen abgrenzt. Bemühen Sie sich besonders, die Perspektiven der Fachabteilung, von Kunden und von Benutzerinnen in die Entscheidungen im Team einzubeziehen. Ansonsten wird Ihre gelieferte Software sehr wahrscheinlich enttäuschen.

Wenn Sie nicht die benötigten Fähigkeiten in der Softwareentwicklung bekommen …

Sie werden dann nicht in der Lage sein, Handlungen routinemäßig in der *Delivering*-Zone auszuführen. Aber die Praktiken aus dieser Zone sind es trotzdem wert, erlernt und angewendet zu werden.

4.3 Wählen Sie agile Coaches aus

Jedes Team braucht einen Coach, der den Teammitgliedern dabei hilft, ein effektives agiles Team zu bilden. Der Abschnitt »Coaching-Fähigkeiten« auf Seite 101 beschreibt die Details, die hier kurz zusammengefasst sind:

- Jedes Team braucht eine Person, die den Teammitgliedern dabei hilft, zu lernen, ein agiles eingespieltes Team zu werden.
- *Focusing*-Teams brauchen eine Person, die ihnen die in Teil II des Buches beschriebenen Planungstechniken beibringen kann.
- *Delivering*-Teams brauchen eine Person, die ihnen die technischen Praktiken aus Teil III des Buches beibringen kann.
- *Optimizing*-Teams brauchen eine Person, die ihnen die in Teil IV des Buches beschriebenen Geschäftsentwicklungspraktiken beibringen kann.

Einige Coaches können mehrere dieser Kategorien abdecken und jeder Coach kann mit einem oder zwei Teams arbeiten.

Wenn Sie nicht den Coach einstellen können, den Sie brauchen …

Sie können Ihre eigenen agilen Coaches aufbauen. Wählen Sie erfahrene Fachleute, die Respekt und Vertrauen genießen und bereit sind, diese Herausforderung anzunehmen. Wenn solche Personen nicht klar erkennbar sind, können Sie Teams um Empfehlungen bitten. Dieses Buch umfasst alle Informationen für den Start als agiler Coach. Die Rolle des Mitspieler-Coaches, der sich voll und ganz einem einzelnen Team widmet, eignet sich am besten dazu.

4.4 Delegieren Sie Befugnisse und Verantwortung an Teams

Respekt vor den Fähigkeiten der Menschen ist ein zentraler Bestandteil in der agilen Philosophie. Dies wird nirgends so deutlich wie im agilen Ansatz zu Autorität und Verantwortung.

Respekt vor den Fähigkeiten der Menschen ist ein zentraler Bestandteil in der agilen Philosophie.

> Eine erstklassige Ausführung besteht darin, die Details richtig zu machen, und niemand versteht die Details besser als die Leute, die die Arbeit tatsächlich machen … Wenn sie mit dem notwendigen Fachwissen ausgestattet sind und von einer Führungskraft angeleitet werden, werden sie bessere technische Entscheidungen und bessere Prozessentscheidungen treffen, als jemand für sie treffen kann. [Poppendieck & Poppendieck 2003]
>
> – Mary and Tom Poppendieck

Aus der Perspektive einer unternehmensweiten Investition bedeutet das:

- *Arbeit wird an Teams und nicht an einzelne Personen vergeben.* Teams entscheiden selbst, wie sie die Arbeit in Aufgaben aufteilen und wer sich um die Aufgaben kümmert. Um das möglich zu machen, müssen Sie eventuell Ticketsysteme und andere Prozesse zur Steuerung des Arbeitsflusses anpassen. Das wiederum hat Einfluss auf Ihre bisherige Leistungsbeurteilung, was zum Abschnitt »Passen Sie nachteilige Vorgaben aus der Personalabteilung an« auf Seite 47 führt.
- *Teams entscheiden über ihre eigenen Prozesse.* Insbesondere müssen Teams die Freiheit haben, ihren eigenen leichtgewichtigen und werkzeugfreien Ansatz für die Planung zu verwenden, anstatt an ein Werkzeug aus dem Unternehmen gebunden zu sein. Die Unternehmensführung kann die Prozesse von Teams einschränken, muss aber für jede Einschränkung einen klar formulierten Grund nennen.

- *Focusing-Teams arbeiten mit Stakeholdern zusammen, um fachliche Anforderungen und Prioritäten zu verstehen.* Die Organisation muss sicherstellen, dass die Teams leichten Zugang zu ihren Stakeholdern oder deren Vertretung haben.
- *Delivering-Teams haben ihre Entwicklung, das Bauen, den Test und die Auslieferung ihres Produkts unter Kontrolle.* Auch hier kann die Unternehmensführung Prozesse einschränken, wie zum Beispiel die Verwendung einer unternehmensweiten Build-Strecke. Es muss aber sichergestellt sein, dass Teams autark, ohne Abhängigkeiten zu anderen Teams, ausliefern können.
- *Optimizing-Teams verwalten ihr eigenes Budget und ihre eigenen Produktpläne.* Die Unternehmensführung definiert den Zweck des jeweiligen Teams, legt die übergreifende Strategie fest und gibt die Budgets der jeweiligen Teams frei. Sie übt außerdem eine Aufsichtsfunktion aus, indem sie Indikatoren zum Business überprüft. Innerhalb dieses Rahmens muss die Organisation den einzelnen Teams die Möglichkeit geben, selbst zu entscheiden, wie sie ihr Ziel erreichen und wofür sie ihr Budget ausgeben wollen.

Wenn Arbeit an einzelne Personen gegeben werden muss …

Wenn Ihre Organisation nicht damit einverstanden ist, dass Teams ihre eigenen Entscheidungen über die Zuweisung von Aufgaben treffen, fehlt das notwendige Vertrauen für eine agile Vorgehensweise. Vielleicht können Sie Leute zum Umdenken bringen, indem die teambasierte Arbeit mit einem agilen Team als Pilot ausprobiert wird. Trotzdem sollten wir bei diesem Vorgehen wachsam sein. Agile Vorgehensmodelle sind im Allgemeinen mit einem befehlsorientierten Führungsstil nicht kompatibel.

Vielleicht ist es kein weitverbreitetes Problem, aber es gibt einzelne Personen im Management, die Schwierigkeiten haben loszulassen. Dazu finden Sie Hinweise im Abschnitt »Passen Sie den Managementstil auf Teamebene an« auf Seite 43.

Wenn Werkzeuge die teambasierte Arbeit nicht unterstützen …

Wenn Ihr Unternehmen ein existierendes System für die Zuordnung von Arbeit hat, das sich schwer anpassen lässt, besteht eine kurzfristige Lösung darin, eine ausgedachte Person als Platzhalter für das Team anzulegen. Dieser Person werden dann die Aufgaben des Teams zugeordnet. Alternativ kann das Team auch individuell Zugeordnetes so behandeln, als sei es dem Team zugeordnet.

Auf lange Sicht ist es besser, die Werkzeuge anzupassen.

Wenn Teams ein unternehmensweites Werkzeug zur Nachverfolgung nutzen müssen …

Einer der größten Hebel für die Wirkung agiler Teams ist die Fähigkeit, ihre Prozesse zu verbessern und effizienter zu gestalten. Unternehmensweite Werkzeuge zur Nachverfolgung, sogenannte Werkzeuge zum Management des agilen Lebenszyklus eingeschlossen, limitieren die Einflussmöglichkeit der Teams. Wie so viele Produkte, die auf den agilen Zug aufgesprungen sind, verfehlen diese Werkzeuge den Sinn eines agilen Vorgehens so sehr, dass sie den Grad von agilem Vorgehen in den Teams sogar verringern.

Agile Teams in die Pflicht zu nehmen, ein unternehmensweites Werkzeug zur Nachverfolgung für ihre tägliche Arbeit zu verwenden, wird ihre Leistung senken. Sollte es keine andere Möglichkeit geben, besteht eine gängige Lösung darin, zwei Werkzeuge zur Nachverfolgung zu unterhalten: eines mit leichtgewichtigem agilem Ansatz und die unternehmensweite Lösung (siehe Abschnitt »Unternehmensweite Werkzeuge zur Nachverfolgung« auf Seite 378).

Agile Teams in die Pflicht zu nehmen, ein unternehmensweites Werkzeug zur Nachverfolgung für die tägliche Arbeit zu verwenden, wird ihre Leistung senken.

Wenn Teams keinen Zugang zu Stakeholdern haben …

Anders als bei Prozessen nach dem Wasserfallmodell, die im Vorfeld eine Phase der Anforderungs- und Geschäftsanalyse vorsehen, arbeiten agile Teams während der gesamten Entwicklung mit den Stakeholdern zusammen, um Pläne zu verfeinern und Feedback einzuholen. Ohne den Zugang zu diesen Personen wird das Team nicht das Richtige bauen.

Kann ein Team nicht mit einer oder mehreren Gruppen von Stakeholdern zusammenarbeiten, sollte sichergestellt sein, dass das Team mit einer Person sprechen kann, die diese vertritt. Diese Person sollte mit Sorgfalt ausgewählt werden. Von ihr hängt die Qualität des Produkts des Teams ab, je nachdem, wie verfügbar und wie befähigt sie darin ist, die Bedürfnisse der repräsentierten Stakeholder genau wiederzugeben.

Wenn Delivering-Teams nicht die Kontrolle über ihre Auslieferungsprozesse haben …

Den vollen Nutzen der *Delivering*-Zone werden Sie erst sehen, wenn Teams die Kontrolle über ihre Freigabe- und Auslieferungsprozesse haben. Dennoch haben die Techniken aus der *Delivering*-Zone einen so großen Wert, dass es sich lohnt, sie weiter zu verfolgen. An dem grundsätzlichen Problem der fehlenden Kontrolle der Auslieferprozesse können Sie mit der Zeit arbeiten.

Wenn Optimizing-Teams nicht die Kontrolle über ihre Produktpläne und Ausgaben haben …

Teams der *Optimizing*-Zone brauchen die Fähigkeit, Experimente durchzuführen und ihre ursprünglichen Pläne anzupassen. Hierfür benötigen sie die Kontrolle über ihre Pläne und Ausgaben. Ohne diese Kontrolle ist es für ein Team nicht möglich, zu routinemäßigen Handlungen in der *Optimizing*-Zone zu kommen.

4.5 Passen Sie den Managementstil auf Teamebene an

Führungspersonen auf der Teamebene haben häufig das Gefühl, keinen Platz mehr im agilen Vorgehensmodell zu haben, wenn Teams über ihre eigenen Prozesse entscheiden, sich selbst Aufgaben zuweisen und sich mit Stakeholdern abstimmen. Doch das stimmt nicht im Entferntesten. Die Arbeit von Führungspersonen verändert sich, doch sie selbst sind nicht weniger wichtig, als sie es vor der Einführung von Agilität waren (siehe Abschnitt »Management« auf Seite 382 für Details).

Sprechen Sie mit Führungspersonen über ihre neue Rolle und bieten Sie auch bei Bedarf Schulungen an. Stellen Sie sicher, dass sich die Erwartungen der übergeordneten Führungspersonen passend verändert haben.

Wenn Führungspersonen Probleme damit haben, loszulassen …

Mikromanagement ist lästig, aber kurzfristig kein Grund zur Sorge. Es hemmt jedoch den Lernprozess der Teams, da ihnen das Mikromanagement Entscheidungen abnimmt. Durch Mikromanagement erhöht sich der benötigte Zeit- und Kostenaufwand für das Erreichen von Fluency.[2]

Führungspersonen wenden häufig Mikromanagement an, wenn sie sich unsicher sind, was sie sonst tun sollen. Oder wenn sie befürchten, dass es in einer agilen Arbeitsumgebung keinen Platz für sie gibt. Daher muss den Führungspersonen vermittelt werden, dass sie immer noch eine Rolle haben, indem ihnen gezeigt wird, wie diese Rolle aussieht. Schulungen oder ein guter agiler Coach können dabei helfen.

4.6 Richten Sie Teamräume ein

Agile Teams arbeiten intensiv zusammen und kommunizieren stetig. Damit diese Kommunikation gut funktioniert, benötigen Teams einen Raum, der ihren Bedürfnissen entspricht. Es kann sich dabei um einen physikalischen oder virtuellen Raum handeln. Der Abschnitt »Teamraum« auf Seite 113 beschreibt die Details.

2. Vielen Dank an George Dinwiddie für diesen Hinweis.

Für Teams, die gemeinsam vor Ort arbeiten, kann die Einrichtung physischer Teamräume eine der teuersten Investitionen sein. Es ist aber auch eine der wertvollsten, wie Abschnitt »Teamraum« auf Seite 113 beschreibt. Physische Teamräume wirken als Multiplikatoren für die Leistung.

> Teams, die neu in die agile Arbeitsweise einsteigen, unterschätzen die Freude an der Zusammenarbeit.

Wenn Ihre Teams gerade starten, ist vielleicht noch nicht bekannt, welche Art von Raum das Team benötigt oder ob Agilität auf lange Sicht überhaupt das passende Mittel ist. Ihre Teams wissen das vermutlich auch nicht. Teams, die neu in die agile Arbeitsweise einsteigen, unterschätzen die Freude an der Zusammenarbeit und überschätzen ihr Bedürfnis nach Privatsphäre.

Es ist also in Ordnung, auf physische Teamräume zu setzen. Legen Sie das Budget dafür zur Seite. Sie werden es irgendwann für gute Teamräume brauchen, wenn Sie bei der Agilität bleiben. Kurzfristig können Sie für jedes Team einen großen Konferenzraum oder einen Teil eines Großraumbüros in Beschlag nehmen.

Egal wofür Sie sich entscheiden, beginnen Sie damit rechtzeitig. Die Einrichtung physischer Teamräume braucht viel Zeit.

Wenn ein Team remote arbeitet …

Sie können einen virtuellen Teamraum anlegen. Abschnitt »Virtuelle Teamräume« auf Seite 127 beschreibt, wie man das macht.

Wenn Sie keinen physischen Teamraum für ein vor Ort arbeitendes Team einrichten können …

Teams, die vor Ort arbeiten, können ebenfalls virtuelle Räume verwenden. Davon rate ich aber dringend ab. Das Team wird die schlechten Dinge beider Welten erleben: die eingeschränkte Flexibilität und das Pendeln bei der Arbeit vor Ort, gepaart mit den Problemen bei der Kommunikation in der Remote-Arbeit.

4.7 Etablieren Sie für jedes Team ein lernfreundliches Ziel

Jedes Team hat einen Zweck: seinen Platz in der übergreifenden Strategie des Unternehmens (siehe Abschnitt »Zweck« auf Seite 145). Wenn ein Team neu mit agilen Methoden startet und diese erlernt, ist es wichtig, ein Ziel zu wählen, das dem Team das Lernen erlaubt. In der Praxis ergeben sich daraus drei Dinge:

- *Ein Ziel, das einen Wert schafft, aber nicht zeitkritisch ist.* Steht das Team unter großem Zeitdruck, haben die Teammitglieder Schwierigkeiten zu lernen. Sie werden eher auf das zurückgreifen, was bereits in der Vergangenheit funktioniert hat, als sich die Zeit zu nehmen, neue Ideen auszuprobieren.

- *Ein Ziel, das in sich geschlossen ist.* Je mehr das Team von anderen Teams abhängt, desto mehr Koordinationsprobleme wird es wahrscheinlich geben. Einige Koordinationsprobleme sind zu erwarten, doch zu viele lenken das Team vom Lernen ab.
- *Eine (brandneue) Codebasis auf der grünen Wiese.* Teams, die sich die Praktiken der *Delivering*-Zone aneignen, haben eine Menge zu lernen und das geht »auf der grünen Wiese« leichter. Trotzdem werden sie irgendwann lernen müssen, wie sie mit bestehendem Quellcode arbeiten. Wird das Team von einem Coach betreut, der selbst Erfahrungen in den Praktiken der *Delivering*-Zone hat, und ist der Coach damit einverstanden, kann diese Anforderung ignoriert werden. Das Gleiche gilt für Teams, die keine Praktiken der *Delivering*-Zone erlernen.

Was ist, wenn es einen wichtigen Abgabetermin gibt …

Jedes Team braucht viel Zeit zum Lernen. Lässt der Abgabetermin genügend Freiraum, ist alles in Ordnung. Ist das nicht der Fall, ist es in der Regel besser, mit der Einführung agiler Methoden erst nach dem Termin zu beginnen. Alternativ können andere Teams dafür ausgesucht werden.

Was ist, wenn es keine wertschöpfende Arbeit auf der grünen Wiese gibt …

Es ist für Teams wichtiger, wertschöpfende Arbeit zu verrichten, als auf einer »grünen Wiese« zu starten. Ohne einen erfahrenen Coach werden Teams, die zum ersten Mal Praktiken der *Delivering*-Zone erlernen, wahrscheinlich Probleme mit bestehendem Quellcode bekommen. Sie müssen daher mit einem länger andauernden Abfall der Leistung, also mehr Zeit, rechnen, die das Team benötigt, um routinemäßige Handlungen der *Delivering*-Zone auszuführen. Außerdem wird es mehr Frustration bei den Programmiererinnen des Teams geben.

4.8 Ersetzen Sie Steuerungsansätze nach dem Wasserfallmodell

Damit Agilität gut funktioniert, müssen Steuerungsrichtlinien aus der Wasserfall-Zeit geändert werden.

Mit Steuerungsrichtlinien wird auf einer hohen Ebene festgelegt, wie Arbeit abgenommen, nachverfolgt und gemanagt wird. Die Führungsrichtlinien der meisten Organisationen gehen von einem Entwicklungsansatz nach dem Wasserfallmodell aus. Das erfordert manchmal eine Dokumentation im Vorwege oder Ziele für verschiedene Phasen der Entwicklung. Oft ist für die Planung ein vorausschauender Ansatz erforderlich.

Für die besten Ergebnisse müssen Vorgaben der Führung verändert werden, um zum agilen Vorgehen zu passen. Das bedeutet, die Ziele einzelner Phasen zu entfernen und adaptives Planen zu verwenden. Lesen Sie in Abschnitt »Agile Steuerung« auf Seite 374 mehr Details dazu.

Wenn die Führung nach dem Wasserfallmodell gefordert wird …

Es ist Verschwendung und schadet dem agilen Vorgehen Ihrer Teams. Wenn es sein muss, kann trotzdem an den Führungsrichtlinien nach dem Wasserfallmodell festgehalten werden. Zumindest für ein paar Teams, die als Pilot fungieren, ist das in Ordnung. Doch bevor Agilität weiter ausgerollt wird, sollte es einen Wechsel hin zu agiler Führung geben.

Häufig fordert die Unternehmensführung, im Voraus einen festen Plan mit Budget zu erstellen. Der einfachste Weg, dem nachzukommen, besteht darin, zunächst bei dem bisher verwendeten Vorgehen zu bleiben, unabhängig davon, welches es war. Nach der Projektgenehmigung können Sie dann mit dem agilen Teil des Prozesses beginnen. Teams, die bereits vertraut mit den Praktiken aus der *Focusing*- und der *Delivering*-Zone sind, können alternativ vier bis acht Wochen für die »Planung« reservieren, mit der normalen Arbeit beginnen und eine qualitativ hochwertige Roadmap erstellen (siehe Abschnitt »Roadmaps« auf Seite 374).

Erfolgreich mit dem Wasserfallvorgehen

Ein agiles Vorgehen ist nicht für jedes Unternehmen der richtige Weg. Und das ist in Ordnung! Sie können auch mit einem Vorgehen nach dem Wasserfallmodell erfolgreich sein. Für Unternehmen, die vorausschauende Pläne benötigen oder eine Befehls- und Kontrollkultur haben, könnte ein Vorgehen nach dem Wasserfallmodell die bessere Wahl sein.

Der sicherste Weg, ein Wasserfallprojekt durchzuführen, ist das iterative Wasserfallverfahren. Statt eines großen Wasserfallprojekts, das viele Risiken beinhaltet, können Sie eine Reihe kleiner Wasserfallprojekte durchführen. Jedes einzelne sollte nicht länger als drei bis sechs Monate dauern und als Ergebnis laufende Software liefern, die tatsächlich auf den Markt gebracht wird. Jedes der Projekte beinhaltet alle Standardphasen eines Wasserfallvorgehens: Analyse der Anforderungen, Architektur und Design, Implementierung, Test oder welche Varianten dieser Phasen das Unternehmen auch immer bevorzugt.

Ein Vorgehen nach dem Wasserfallmodell funktioniert am besten in gut verstandenen Domänen ohne viele Unwägbarkeiten. Es sollten Personen eingestellt werden, die bereits viel Erfahrung in der Entwicklung der Software haben, die ausgeliefert werden soll.

Weitere im Voraus zu erstellende Dokumentationen, so wie ein Dokument zur Anforderungsanalyse oder ein Entwurfsdokument, können auch mit den vorher verwendeten Ansätzen durchgeführt werden, bevor die Arbeit am agilen Prozess

beginnt. Aus der alten Welt verbleibende Arbeit, bei der vielleicht Bestimmungen eingehalten werden müssen, wie auch alle anderen Anfragen können nach der Umstellung auf agile Methoden mittels Storys (siehe Abschnitt »Storys« auf Seite 184) eingeplant werden.

Steuerungsrichtlinien nach dem Wasserfallmodell sind nicht mit der *Optimizing*-Zone kompatibel, die darauf baut, die vorhandenen Planungen anzupassen. Sind Sie gezwungen, die Kontrollstruktur aus dem Wasserfallmodell beizubehalten, sollte ihr Anspruch an Agilität auf die Zonen *Focusing* und *Delivering* beschränkt werden.

4.9 Passen Sie nachteilige Vorgaben aus der Personalabteilung an

Agiles Vorgehen ist ein Teamsport. Obwohl sich Unternehmen zu Teamarbeit bekennen, haben viele von ihnen eine Vorgabe, die ungewollt von Teamarbeit abschreckt. Jede Vorgabe, die Menschen in Konkurrenz zueinander setzt, wird Agilität erschweren. Ein besonders destruktives Beispiel ist das sogenannte Stack-Ranking, bei dem Teammitglieder relativ zueinander bewertet werden. Dabei werden die obersten Plätze des Stapels befördert und die untersten Plätze, unabhängig von ihrer tatsächlichen Leistung, entlassen.

Führungspersonen, die nur greifbare Ergebnisse wertschätzen, sind ein ähnliches Problem. In einem agilen Team gibt es viele Wege, um zum Erfolg beizutragen. Nehmen wir zum Beispiel die Person, die wenig neuen Quellcode schreibt, dafür aber viel Zeit damit verbringt, Fehlern nachzugehen. Oder das Teammitglied, das im Hintergrund daran arbeitet, die Kommunikation zu verbessern.

Manche Unternehmen entwickeln eine Kultur des Beschuldigens. Dabei wird auf Fehler reagiert, indem die schuldige Person bestraft wird. In der agilen Denkweise werden Fehler als eine Möglichkeit zum Lernen begriffen. Als Beispiel könnte ein Unternehmen ohne agile Denkweise eine Person entlassen, die versehentlich eine wichtige Datenbank im produktiven System gelöscht hat. In einer agilen Organisation würde sich stattdessen die Fragestellung ergeben, welche Prüfungen wir einführen müssen, um ein versehentliches Löschen einer Datenbank zu verhindern, und wie wir die Wiederherstellung nach solchen Fehlern erleichtern können.

In der Personalpolitik spiegeln sich die kulturellen Aspekte in Bezug auf Beförderungen und Belohnungen häufig wider. Wenn die Karrieren von Mitarbeitenden davon abhängen, sich selbst in einem guten Licht darzustellen, und es unerheblich ist, inwieweit sie zur Leistungsfähigkeit des Teams beitragen, dann werden Ihre Teams wahrscheinlich Probleme mit dem agilen Ansatz von Zusammenarbeit bekommen.

Sie werden nicht in der Lage sein, Ihre Unternehmenskultur über Nacht zu verändern. Aber es ist möglich, Vorgaben aus der Personalabteilung zu ändern.