Robert C. Martin

Mit Beiträgen von:
Jerry Fitzpatrick
Tim Ottinger
Jeff Langr
Eric Crichlow
Damon Poole
Sandro Mancuso

Deutsche Ausgabe

Clean Agile

Die Essenz der agilen Softwareentwicklung

Zurück zu den Ursprüngen:
Die agilen Werte und Prinzipien
effektiv in der Praxis umsetzen

Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)

Inhaltsverzeichnis

Stimmen zum Buch

»Auf dem Weg zur vollständigen Umstellung auf agile Programmierung hat sich Uncle Bob gründlich umgesehen und sowohl Licht als auch Schatten erblickt. Dieses wunderbare Buch erzählt die Entwicklung, schildert aber auch persönliche Erfahrungen. Wenn Sie wissen möchten, was agile Softwareentwicklung ist und wie sie erarbeitet wurde, dann ist dieses Buch für Sie das Richtige.«

– *Grady Booch*

»Bobs Frustration spiegelt sich in jedem Satz des Buchs wider – und das zu Recht. Der Zustand, in dem sich die Welt der agilen Softwareentwicklung *befindet*, ist nichts im Vergleich zu dem, was sie *sein könnte*. Das Buch beschreibt, auf was man sich aus Bobs Sicht konzentrieren sollte, um das zu erreichen, was sein könnte. Und er hat es schon erreicht, deshalb lohnt es sich, ihm zuzuhören.«

– *Kent Beck*

»Es tut gut, Uncle Bobs Ansichten zur agilen Softwareentwicklung zu lesen. Dieses Buch ist es wert, gelesen zu werden, sowohl von Einsteigern als auch von erfahrenen Programmierern. Ich stimme nahezu allem darin zu. Manche Stellen machen mir allerdings meine eigenen Unzulänglichkeiten bewusst. Ich habe daraufhin gleich überprüft, wie hoch unsere Testabdeckung ist (85,09 Prozent).«

– *Jon Kern*

»Dieses Buch betrachtet agile Softwareentwicklung aus einer historischen Perspektive und ermöglicht so ein umfassenderes und präziseres Verständnis der Thematik. Uncle Bob ist einer der klügsten Menschen, die ich kenne, und seine Begeisterung für Programmierung kennt keine Grenzen. Wenn es jemandem gelingt, agile Softwareentwicklung zu entmystifizieren, dann ihm.«

– *Aus dem Vorwort von Jerry Fitzpatrick*

Für alle Programmierer, die jemals mit
Windmühlen oder Wasserfällen
zu kämpfen hatten.

Robert C. Martin

Clean Agile

Die Essenz der agilen Softwareentwicklung

Zurück zu den Ursprüngen: Die agilen Werte und Prinzipien effektiv in der Praxis umsetzen

Übersetzung aus dem Amerikanischen
von Knut Lorenzen

Impressum

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 978-3-7475-0113-9
1. Auflage 2020

www.mitp.de
E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Lektorat: Janina Bahlmann
Sprachkorrektorat: Petra Heubach-Erdmann
Fachkorrektorat: Prof. Dr. Michael Fellmann
Covergestaltung: Christian Kalkert
Bildnachweis: © peresanz / stock.adobe.com
electronic **pub**lication: III-satz, Husby, www.drei-satz.de

Mikroverfilmungen und Einspeicherung und Verarbeitung in elektronischen Systemen.

Der Verlag schützt seine ebooks vor Missbrauch des Urheberrechts durch ein digitales Rechtemanagement. Bei Kauf im Webshop des Verlages werden die ebooks mit einem nicht sichtbaren digitalen Wasserzeichen individuell pro Nutzer signiert.

Bei Kauf in anderen ebook-Webshops erfolgt die Signatur durch die Shopbetreiber. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

Vorwort

Was genau ist agile Softwareentwicklung? Wie ist sie entstanden? Wie hat sie sich entwickelt?

In diesem Buch liefert Uncle Bob wohldurchdachte Antworten auf diese Fragen. Zudem benennt er die vielfältigen Weisen, auf die agile Softwareentwicklung fehlinterpretiert oder korrumpiert wurde. Seine Sichtweise ist von besonderer Bedeutung, weil er auf diesem Fachgebiet eine Autorität ist und an der Entstehung agiler Softwareentwicklung beteiligt war.

Bob und ich sind seit vielen Jahren befreundet. Wir trafen uns zum ersten Mal, als ich 1979 in der Telekommunikationsabteilung von Teradyne eingestellt wurde. Bei meiner Tätigkeit als Elektrotechniker installierte und betreute ich die verschiedensten Produkte. Später wurde ich dann Hardware-Designer.

Etwa ein Jahr darauf begann das Unternehmen, nach neuen Produktideen zu suchen. 1981 schlugen Bob und ich einen elektronischen Rezeptionisten für die Telefonanlage vor. Dabei handelte es sich im Wesentlichen um ein Voicemail-System, das Anrufe weiterleiten konnte. Das Konzept gefiel dem Unternehmen und schon bald begannen wir mit der Entwicklung von »E.R. – The Electronic Receptionist«. Unser Prototyp war auf dem neuesten Stand der Technik. Er verwendete das Betriebssystem MP/M, das auf einem 8086-Prozessor von Intel lief. Die Sprachnachrichten wurden auf einer ST-506-Festplatte von Seagate gespeichert, die über 5 Megabyte Speicherplatz verfügte. Ich entwarf die Hardware für das Sprachdialogsystem und Bob entwickelte die Anwendung. Als ich fertig war, beteiligte ich mich auch an der Programmierung – seitdem bin ich Softwareentwickler.

Um 1985 oder 1986 stellte Teradyne die Entwicklung von E.R. plötzlich ein und zog – ohne dass wir davon wussten – die Patentanmeldung zurück. Das war eine Entscheidung, die das Unternehmen schon bald bereuen sollte und die Bob und mir heute noch zu schaffen macht.

Schließlich verließen wir beide das Unternehmen, um nach neuen Möglichkeiten zu suchen. Bob gründete ein Beratungsunternehmen im Großraum Chicago. Ich wurde freiberuflicher Softwareentwickler und Dozent. Wir blieben weiter in Kontakt, obwohl ich in einen anderen Bundesstaat umgezogen war.

Im Jahr 2000 unterrichtete ich objektorientierte Analyse und Design bei Learning Tree International. Der Lehrgang umfasste UML (Unified Modeling Language) und USDP (Unified Software Development Process). Mit diesen Technologien kannte ich mich gut aus, nicht aber mit Scrum, Extreme Programming oder ähnlichen Praktiken.

Im Februar 2001 wurde das agile Manifest veröffentlicht. Wie bei vielen anderen Entwicklern auch war meine erste Reaktion: »Das Agile was?« Das einzige Manifest, das ich kannte, war das von Karl Marx, einem leidenschaftlichen Kommunisten. War dieses Manifest ein Ruf zu den Waffen? Von verdammten Software-Radikalisten?

Das Manifest löste eine Art Rebellion aus. Es sollte die Entwicklung von schlankem, sauberem Code durch einen kollaborativen, anpassungsfähigen und Feedback-getriebenen Ansatz inspirieren. Es bot eine Alternative zu »schwergewichtigen« Prozessen wie dem Wasserfallmodell oder dem USDP.

Es ist jetzt 18 Jahre her, dass das agile Manifest veröffentlicht wurde. Für die meisten heutigen Entwickler ist

es also uralte Geschichte. Aus diesem Grund wäre es möglich, dass Ihr Verständnis von agiler Softwareentwicklung nicht mit dem übereinstimmt, was ihre Urheber beabsichtigten.

Dieses Buch möchte das richtigstellen. Es betrachtet agile Softwareentwicklung aus einer historischen Perspektive und ermöglicht so ein umfassenderes und präziseres Verständnis der Thematik. Uncle Bob ist einer der klügsten Menschen, die ich kenne, und seine Begeisterung für Programmierung kennt keine Grenzen. Wenn es jemandem gelingt, agile Softwareentwicklung zu entmystifizieren, dann ihm.

– Jerry Fitzpatrick

Software Renovation Corporation

März 2019

Einleitung

Dieses Buch ist keine Forschungsarbeit. Ich habe die vorhandene Literatur nicht gewissenhaft gesichtet. Was Sie lesen werden, sind meine persönlichen Rückbesinnungen, Beobachtungen und Ansichten basierend auf meiner

zwanzigjährigen Beschäftigung mit agiler Softwareentwicklung – nicht mehr und nicht weniger.

Der Schreibstil ist erzählerisch und umgangssprachlich. Meine Wortwahl ist manchmal etwas ungeschliffen. Auch wenn ich eigentlich nicht zum Fluchen neige, findet sich hier und da vielleicht ein (mildes) Schimpfwort, weil mir kein besserer Weg eingefallen ist, meine Absichten zu vermitteln.

Zudem ist dieses Buch keine reine Lobhudelei. Wenn es mir notwendig erschien, habe ich Verweise auf Referenzen hinzugefügt, denen Sie folgen können. Ich habe meine Erkenntnisse mit denen anderer verglichen, die der Community der agilen Softwareentwicklung ebenso lange angehören wie ich. Einige habe ich sogar darum gebeten, für eigene Kapitel und Abschnitte ergänzende und entgegengesetzte Sichtweisen bereitzustellen. Dennoch sollten Sie das Buch nicht für eine akademische Arbeit halten. Es ist wohl besser, es als persönliche Aufzeichnung zu betrachten – als das Grummeln eines Griesgrams, der die nächste Generation agiler Softwareentwickler von seinem Land vertreiben will.

Das Buch richtet sich sowohl an Programmierer als auch an Nicht-Programmierer. Es ist nicht technisch; es gibt keinen Code. Es soll einen Überblick über die ursprüngliche Absicht agiler Softwareentwicklung verschaffen, ohne sich in technischen Details der Programmierung, des Testens und des Managens zu verlieren.

Das Buch ist ziemlich kurz, weil das Thema nicht sehr umfassend ist. Agile Softwareentwicklung ist ein einfaches Konzept für einfache Aufgaben, die kleine Teams von Programmierern lösen. Agile Softwareentwicklung ist *kein* großes Konzept zum Lösen umfassender Aufgaben durch große Teams. Es ist schon etwas paradox, dass diese

einfache Lösung für ein kleines Problem einen eigenen Namen trägt. Das fragliche kleine Problem wurde schließlich schon in den 1950er- und 1960er-Jahren gelöst, fast unmittelbar, nachdem Software erfunden wurde. Damals lernten kleine Softwareteams, einfache Aufgaben ziemlich gut zu lösen. Das änderte sich in den 1970er-Jahren, als kleine Softwareteams, die einfache Aufgaben lösten, in eine Ideologie verwickelt wurden, die den Standpunkt vertrat, dass große Teams umfassende Aufgaben lösen sollten.

Sollten wir umfassende Aufgaben mit großen Teams lösen? Ach du lieber Himmel, nein! Umfassende Aufgaben lassen sich nicht von großen Teams lösen. Umfassende Aufgaben werden durch die Zusammenarbeit vieler kleiner Teams gelöst, die einfache Aufgaben lösen. Das war den Programmierern in den 1950er- und 1960er-Jahren instinktiv klar. Und das wurde in den 1970er-Jahren vergessen.

Weshalb wurde es vergessen? Ich hege den Verdacht, dass es zu einem Bruch kam. Die Anzahl der Programmierer nahm in den 1970er-Jahren explosionsartig zu. Vorher gab es weltweit nur ein paar Tausend Programmierer, nachher waren es mehrere Hunderttausend. Mittlerweile nähert sich diese Zahl hundert Millionen.

Die ersten Programmierer in den 1950er- und 1960er-Jahren waren keine jungen Leute. Sie lernten mit 30, 40 oder 50 zu programmieren. Aber in den 1970er-Jahren, als die Zahl der Programmierer drastisch zunahm, gingen die alten Hasen allmählich in den Ruhestand. Die erforderliche Ausbildung des Nachwuchses fand also nie statt. Ein Heer junger Leute in den Zwanzigern betrat den Arbeitsmarkt genau in dem Moment, in dem die erfahrenen Mitarbeiter in den Ruhezustand gingen. Ihre Erfahrung wurde also faktisch nicht weitergegeben.

Manche Leute behaupten, dass dadurch eine Art »dunkles Zeitalter« der Programmierung eingeläutet wurde. 30 Jahre lang kämpften wir mit der Vorstellung, dass umfassende Aufgaben von großen Teams erledigt werden müssen, ohne zu wissen, dass der Trick darin besteht, dass viele kleine Teams viele kleine Aufgaben erledigen.

Mitte der 1990er-Jahre wurde uns allmählich klar, dass wir den Kampf verloren hatten. Das Konzept kleiner Teams keimte auf und gedieh. Es verbreitete sich in der Community der Softwareentwickler und nahm Fahrt auf. Im Jahr 2000 stellten wir fest, dass wir einen branchenweiten Neustart brauchten. Wir mussten daran erinnert werden, was unsere Vorgänger instinktiv wussten. Wir mussten abermals zur Kenntnis nehmen, dass umfassende Aufgaben von vielen zusammenarbeitenden kleinen Teams erledigt werden, die kleine Aufgaben erledigen.

Um dieser Vorstellung zum Durchbruch zu verhelfen, gaben wir dem Konzept einen Namen. Wir nannten es »agile Softwareentwicklung«.

Ich habe dieses Vorwort Anfang 2019 verfasst. Der »Neustart« im Jahre 2000 ist nun fast zwei Jahrzehnte her und ich habe den Eindruck, dass ein weiterer Neustart erforderlich ist. Warum? Weil die schlichte und einfache Botschaft, die agile Softwareentwicklung vermitteln soll, in all den Jahren durcheinandergeraten ist. Sie wurde mit Konzepten wie Lean, Kanban, LeSS, SAFe, Modern, Skilled und vielen anderen vermengt. Diese anderen Konzepte sind auch nicht schlecht, haben aber nichts mit der ursprünglichen Botschaft agiler Softwareentwicklung zu tun.

Es ist also wieder an der Zeit, dass wir uns daran erinnern, was unsere Vorgänger in den 1950er- und 1960er-Jahren wussten und was wir schon im Jahr 2000 abermals lernen

mussten. Wir müssen uns daran erinnern, was agile Softwareentwicklung tatsächlich ist.

In diesem Buch werden Sie nichts finden, was besonders neu, überraschend oder verblüffend ist, und nichts Revolutionäres, das den üblichen Rahmen sprengt, sondern eine Neuformulierung der agilen Softwareentwicklung, wie man sie aus dem Jahr 2000 kennt. Sie wird allerdings aus einer anderen Perspektive betrachtet, und wir haben in den letzten zwanzig Jahren ein paar Dinge gelernt, die ich ebenfalls erörtern werde. Im Großen und Ganzen entspricht die Botschaft dieses Buchs aber denjenigen aus den Jahren 2001 und 1950.

Diese Botschaft ist schon alt, aber sie ist wahr. Sie liefert uns eine einfache Lösung für das kleine Problem, dass kleine Teams kleine Aufgaben erledigen.

Danksagungen

Mein erster Dank gilt zwei unerschrockenen Programmierern, die freudig die in diesem Buch beschriebenen Verfahren entdeckt (oder wiederentdeckt) haben: Ward Cunningham und Kent Beck.

Gleich darauf folgt Martin Fowler, ohne dessen ruhige Hand in der Anfangsphase die agile Revolution wohl nicht möglich gewesen wäre.

Ken Schwaber verdient Erwähnung, weil er sich mit unerschöpflicher Energie für die Förderung und Verbreitung agiler Softwareentwicklung eingesetzt hat.

Auch Mary Poppendieck soll erwähnt werden, weil sie selbstlos und unermüdlich für agile Softwareentwicklung eingetreten ist und die Agile Alliance geführt hat.

Aus meiner Sicht war Ron Jeffries dank seiner Reden, Artikel, Blogbeiträge und seines warmherzigen Charakters stets das gute Gewissen der frühen Bewegung der agilen Softwareentwicklung.

Mike Beedle kämpfte beherzt für die agile Softwareentwicklung, doch er wurde auf den Straßen von Chicago von einem Obdachlosen ermordet.

Die anderen Autoren des agilen Manifests sollen an dieser Stelle ebenfalls genannt werden:

Arie van Bennekum, Alistair Cockburn, James Grenning, Jim Highsmith, Andrew Hunt, Jon Kern, Brian Marick, Steve Mellor, Jeff Sutherland und Dave Thomas.

Mein Freund und damaliger Geschäftspartner Jim Newkirk hat die agile Softwareentwicklung unermüdlich unterstützt, während er mit privaten Problemen kämpfte, die sich die meisten von uns (ich jedenfalls) kaum vorstellen können.

Die Mitarbeiter von Object Mentor Inc. sollen hier ebenfalls Erwähnung finden. Sie haben das Risiko auf sich genommen, agile Softwareentwicklung willkommen zu heißen und zu fördern. Viele davon sind auf dem folgenden Foto zu sehen, das beim Start des ersten XP-Immersion-Kurses aufgenommen wurde.

Ebenfalls bedanken möchte ich mich bei den Menschen, die sich zusammengefunden haben, um die Agile Alliance zu gründen. Einige von ihnen sind auf dem folgenden Foto zu sehen, das beim ersten Treffen der inzwischen etablierten Organisation entstand.

Und schließlich danke ich allen Mitarbeitern von Pearson, insbesondere meiner Herausgeberin Julie Phifer.

Abb. 1: Hinten: Ron Jeffries, Autor, Brian Button, Lowell Lindstrom, Kent Beck, Micah Martin, Angelique Martin, Susan Rosso, James Grenning
Vorne: David Farber, Eric Meade, Mike Hill, Chris Biegay, Alan Francis, Jennifer Kohnke, Talisha Jefferson, Pascal Roy
Nicht auf dem Foto: Tim Ottinger, Jeff Langr, Bob Koss, Jim Newkirk, Michael Feathers, Dean Wampler und David Chelimsky

Abb. 2: Von links nach rechts: Mary Poppendieck, Ken Schwaber, Autor, Mike Beedle, Jim Highsmith (nicht auf dem Foto: Ron Crocker)

Über den Autor

Robert C. Martin (auch bekannt als »Uncle Bob«) ist seit 1970 als Softwareprogrammierer tätig. Er ist Mitbegründer der Organisation *cleancoders.com*, die Online-Videotrainings für Softwareentwickler bereitstellt. Des Weiteren gründete er das Unternehmen *Uncle Bob Consulting LLC*, das Dienstleistungen in den Bereichen IT-Beratung, Training und Skill Development für große Firmen auf der ganzen Welt anbietet. Außerdem war er als Master Craftsman in dem in Chicago ansässigen IT-Consulting-Unternehmen *8th Light Inc.* beschäftigt.

Martin hat Dutzende von Artikeln in diversen Handelsmagazinen veröffentlicht und tritt regelmäßig als Redner bei internationalen Konferenzen und Kongressen auf.