

O'REILLY®

Der
US-Bestseller von
Kent Beck

Tidy First?

Mini-Refactorings für besseres
Software-Design



Kent Beck

Mit einem Vorwort von Larry Constantine
Übersetzung von Thomas Demmig

Stimmen zum Buch

Es lohnt sich immer, Kent Beck zuzuhören, und ich habe nun schon mehrere Jahrzehnte auf die Ratschläge, die in diesem Buch zu finden sind, gewartet. Das Buch unterstützt mich dabei, den Fokus bei der Softwareentwicklung von Tools und Technologie wegzubewegen hin zu dem, was wirklich wichtig ist: Design! Beim Design geht es um die Formen, die wir mit unserem Code erschaffen, und Kent hilft uns dabei, bessere Formen zu kreieren. Ein wichtiges Buch zu einem wichtigen Thema.

– *Dave Farley, Gründer und Direktor von Continuous Delivery Ltd.*

Es kann beim Entwickeln in nur schwer verständlichen Codebasen knifflig sein, herauszufinden, wo man anfangen soll. Dieses Buch gibt praktische Tipps für Entwickler jeder Erfahrungsstufe, um den Code zu verbessern, mit dem sie arbeiten.

– *Sam Newman, freier Berater, Technologe und Autor von »Building Microservices« und »Vom Monolithen zu Microservices«*

Kent Beck stellt uns Dutzende leicht verständliche Ideen dazu vor, wie man komplizierten Code in eine einfachere Form bringen kann. Die Ideen sind simpel, und doch werden Sie sich beim Lesen fragen, warum Sie nicht schon selbst darauf gekommen sind. Dieses Buch empfiehlt sich für alle, die auf sauberen und lesbaren Code Wert legen.

– *Gergely Orosz, The Pragmatic Engineer*

Seit Jahrzehnten haben sich Bücher zum Refactoring auf objektorientierte Top-down-Softwaredesign-Theorien fokussiert. *Tidy First?* bricht damit und stellt einen realistischen Ansatz vor, bei dem echter, produktiver Code inkrementell verbessert wird.

– *Maude Lemaire, Autorin von »Refactoring at Scale«*

Seien wir doch ehrlich: 99 % der Softwareentwicklung findet in Brownfield-Projekten statt. Das kann schwierig sein, insbesondere wenn der Code nicht gerade auf Lesbarkeit hin optimiert wurde. In diesem Buch verändert Kent Beck das Ganze, indem er die Beziehung zwischen Menschen durch Code priorisiert. Er zeigt Schritt für Schritt, wie sich Softwaredesign durch kleine, graduelle Änderungen verbessern lässt, sodass der Code nicht nur für Sie, sondern auch für Ihre Mitstreitenden klarer wird.

– Vlad Khononov, Autor von »Einführung in Domain-Driven Design«

Copyright und Urheberrechte:

Die durch die dpunkt.verlag GmbH vertriebenen digitalen Inhalte sind urheberrechtlich geschützt. Der Nutzer verpflichtet sich, die Urheberrechte anzuerkennen und einzuhalten. Es werden keine Urheber-, Nutzungs- und sonstigen Schutzrechte an den Inhalten auf den Nutzer übertragen. Der Nutzer ist nur berechtigt, den abgerufenen Inhalt zu eigenen Zwecken zu nutzen. Er ist nicht berechtigt, den Inhalt im Internet, in Intranets, in Extranets oder sonst wie Dritten zur Verwertung zur Verfügung zu stellen. Eine öffentliche Wiedergabe oder sonstige Weiterveröffentlichung und eine gewerbliche Vervielfältigung der Inhalte wird ausdrücklich ausgeschlossen. Der Nutzer darf Urheberrechtsvermerke, Markenzeichen und andere Rechtsvorbehalte im abgerufenen Inhalt nicht entfernen.

Tidy First?

Mini-Refactorings für besseres Softwaredesign

Kent Beck

*Deutsche Übersetzung von
Thomas Demmig*

O'REILLY®

Kent Beck

Lektorat: Ariane Hesse

Übersetzung: Thomas Demmig

Fachliche Unterstützung: Patrick Baumgartner, Technical Agile Coach und Software Crafter bei 42talents und Marco Emrich, Software Architect and Primary Consultant at codecentric AG

Korrektur: Sibylle Feldmann, www.richtiger-text.de

Satz: III-satz, www.drei-satz.de

Herstellung: Stefanie Weidner

Umschlaggestaltung: Karen Montgomery, Michael Oréal, www.oreal.de

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 978-3-96009-244-5

PDF 978-3-96010-841-2

ePub 978-3-96010-842-9

1. Auflage 2024

Translation Copyright © 2024 dpunkt.verlag GmbH

Wieblinger Weg 17

69123 Heidelberg

Authorized German translation of the English edition of *Tidy First?* ISBN 9781098151249 © 2024 Kent Beck. This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«. O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

Schreiben Sie uns:

Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: komentar@oreilly.de.

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Übersetzer noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

*Im Gedenken an Professor Barry Dwolatzky:
Geek der Extraklasse, Urgewalt und Inspiration.*

Inhalt

Vorwort	11
Einleitung	13
Einführung	19
<hr/>	
Teil I: Aufräumereien	21
1 Guard Clauses	23
2 Toter Code	25
3 Symmetrien normalisieren	27
4 Neue Schnittstelle, alte Implementierung	29
5 Lesereihenfolge	31
6 Kohäsionsreihenfolge	33
7 Deklaration und Initialisierung zusammenbringen	35
8 Beschreibende Variablen	37
9 Beschreibende Konstanten	39
10 Explizite Parameter	41
11 Anweisungen gruppieren	43
12 Hilfsroutinen extrahieren	45
13 Ein Haufen	47

14 Erläuternde Kommentare	49
15 Redundante Kommentare entfernen	51
<hr/>	
Teil II: Managen	53
16 Getrenntes Aufräumen	55
17 Verketteten	59
18 Batchgrößen	63
19 Rhythmus	67
20 Entwirren	69
21 Vorher, nachher, später, nie	71
<hr/>	
Teil III: Theorie	75
22 Vorteilhafte Beziehungen zwischen Elementen	77
23 Struktur und Verhalten	81
24 Ökonomie: der Wert der Zeit und der Optionalität	85
25 Ein Dollar heute > ein Dollar morgen	87
26 Optionen	89
27 Optionen versus Zahlungsflüsse	93
28 Reversible Strukturänderungen	95
29 Kopplung	97
30 Constantines Äquivalenz	101
31 Kopplung versus Entkopplung	105
32 Kohäsion	109
33 Zusammenfassung	111
Anhang: Kommentierte Leseliste und Referenzen	115
Index	119

Dieses schmale Büchlein, das erste einer Reihe, ist für professionelle Programmierer und Programmierinnen gedacht – für die Art von Softwareentwicklern mit einem tiefen und geekigen Interesse an ihrem Handwerk und am Verbessern ihrer Arbeit mit wenig Aufwand, aber großer Wirkung. Autor Kent Beck ist solch ein engagierter Profi, der sich um Details kümmert, aber auch einen Blick für übergeordnete Fragen und das große Ganze hat.

Praktizierende Softwareentwicklerinnen und -entwickler interessieren sich oft nur wenig für Theorie, aber Kent weiß, worüber er redet, wenn er Praxis und Theorie in einem Ratgeber zusammenbringt, um Code aufzuräumen, der auf diese Weise sowohl lesbar als auch handhabbar wird.

In der Theorie gibt es keinen Unterschied zwischen Theorie und Praxis – in der Praxis aber schon. Dieser Sinnspruch ist in diversen Variationen weit verbreitet und wurde neben anderen schon fälschlicherweise Albert Einstein und Yogi Berra zugeordnet. Nur einem streberhaften Wortschöpfer (erwischt!) dürfte es wichtig sein, ihn korrekterweise Benjamin Brewster zuzuordnen, einem Yale-Studenten, der ihn im Jahr 1882 im *Yale Literature Magazine* veröffentlicht hatte. Dank der engagierten Wortfreaks bei *QuoteInvestigator.com* kann ich dieses geekige Detail im Vertrauen auf das Publikum hier anbieten: Es kommt bei dem Beruf darauf an, die Details richtig zu machen.

Durch das Zusammenbringen von Theorie und Praxis beginnt Kent ganz unten mit kleinen Codeschnipseln und akribischer Aufmerksamkeit für die winzigen Details. Dann arbeitet er sich nach oben vor zu einem weiteren Blickwinkel, der den Prozess des Schaffens saubereren Codes in den Blick nimmt. Sauberer Code, der angesichts unvermeidbarer Änderungen und Korrekturen robuster ist. Bei der Zusammenstellung dieses Leitfadens für die Praxis bezieht sich Kent letztendlich auf die realen wirtschaftlichen Aspekte der Softwareentwicklung und die Theorie der Softwareentwicklung.

Diese Kerntheorie ist einfach: Die Komplexität von Computercode hängt davon ab, wie er aufgeteilt ist, wie gekoppelt diese Teile untereinander und wie kohäsiv sie in sich sind. Die Quelle der Theorie von Kopplung und Kohäsion wird meist dem von mir zusammen mit Ed Yourdon geschriebenen Buch *Structured Design* (Yourdon

Press 1975, Prentice Hall 1979) zugeschrieben, aber es lässt sich auch bis zu einer Konferenz in Cambridge, Massachusetts, im Jahr 1968 zurückverfolgen. Kopplung und Kohäsion haben es fast nicht in die 1979er-Auflage von Prentice Hall geschafft. Die Lektoren hatten versucht, Ed und mich davon zu überzeugen, die zwei Kapitel wegzulassen, weil »niemand an der Theorie interessiert sei«. Zum Glück für die Geschichte der Softwareentwicklung blieben die Autoren standhaft, und die Lektoren lagen falsch. Seitdem hat sich die Theorie in einem halben Jahrhundert Praxis und in Hunderten von Studien und Untersuchungen als gültig erwiesen.

Kopplung und Kohäsion sind einfache Maßstäbe für die Komplexität von Computercode – nicht aus der Perspektive des Computers, der das Programm ausführt, sondern aus der des Menschen, der versucht, den Code zu verstehen. Um ein Programm zu verstehen – sei es, um es zu erstellen, oder sei es, um es zu korrigieren oder anzupassen –, müssen Sie das Stück Code vor sich auf dem Bildschirm genauso verstehen wie die anderen Elemente, mit denen es verbunden ist, die es beeinflusst oder durch die es beeinflusst wird. Es ist einfacher, den aktuellen Code zu verstehen, wenn alles beisammen ist, als Ganzes Sinn hat und das formt, was Psychologinnen und Psychologen als Gestalt bezeichnen. Das ist Kohäsion. Es ist zudem einfacher, den Code im Hinblick auf seine Beziehungen zu anderen Codebereichen zu verstehen, wenn es nur wenige dieser Beziehungen gibt, die recht schwach sind oder stark eingeschränkt. Das ist Kopplung. Kopplung und Kohäsion sind alles, womit sich Ihr Hirn bei komplizierten Systemen beschäftigt.

Sehen Sie? Nett und ordentlich. Das ist die Theorie. Jetzt aber zu den praktischen Details und dem Untermischen von gerade so viel Theorie, dass alles Sinn ergibt. Kent Beck wird Sie auf diesem Weg leiten können.

– Larry Constantine
Rowley, Massachusetts,
9. Oktober 2023

Larry Constantine war als Professor an der Universität von Madeira (Portugal) und der University of Technology in Sydney (Australien) tätig. Er ist an mehr als 200 Artikeln und drei Dutzend Büchern beteiligt gewesen – einschließlich des Jolt-Award-Gewinners *Software for Use* (Addison-Wesley 1999), geschrieben von Lucy Lockwood – und 15 Romanen unter seinem Pseudonym Lior Samson.