Andreas Rüping

Dokumentation in agilen Projekten

Lösungsmuster für ein bedarfsgerechtes Vorgehen

dpunkt.verlag

Dr. Andreas Rüping ist freiberuflicher IT-Berater mit dem Schwerpunkt Webapplikationen und lebt in Hamburg. Er favorisiert ein agiles Vorgehen im Projektalltag und unterstützt Unternehmen bei der Durchführung agiler Entwicklungsprojekte. Er ist aktives Mitglied der europäischen Pattern Community und Autor mehrerer Artikel und Fachbücher zu verschiedenen Themen der Softwareentwicklung, insbesondere Content Management und Dokumentation.

Andreas Rüping

Dokumentation in agilen Projekten

Lösungsmuster für ein bedarfsgerechtes Vorgehen

dpunkt.verlag

Andreas Rüping
andreas.rueping@rueping.info

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Herstellung: Birgit Bäuerlein
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie 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:
Buch 978-3-86490-040-2
PDF 978-3-86491-312-9
ePub 978-3-86491-313-6

1. Auflage 2013

Ringstraße 19 B
69115 Heidelberg

5 4 3 2 1 0

Vorwort

Agile Verfahren haben in den letzten Jahren viel frischen Wind in die Softwareentwicklung gebracht. eXtreme Programming, Scrum, Kanban und Co. haben einige Popularität erreicht und sind heute in vielen Softwareprojekten die Methode der Wahl. Im Zuge dieser Entwicklung ist eine Reihe von Praktiken in den Alltag der Softwareentwicklung eingezogen, die sich im Projektalltag immer wieder als erfolgreich herausgestellt haben. Typische Beispiele dieser agilen Praktiken sind inkrementelle Prozesse, testgetriebene Entwicklung, enge Kooperation mit dem Kunden sowie die regelmäßige Reflexion des eigenen Vorgehens.

Gute Dokumentation ist als Merkmal agiler Verfahren weniger bekannt. Eher ist das Gegenteil der Fall: Eines der Markenzeichen agiler Verfahren ist, dass sie eine kritische Sicht auf die Dokumentation im Projekt einnehmen. Im Agilen Manifest, das die Grundzüge agiler Softwareentwicklung zusammenfasst, wird die umfassende Dokumentation den weniger wichtigen Dingen im Projekt zugeordnet.

Bedeutet dies, dass Dokumentation in agil durchgeführten Projekten keine nennenswerte Rolle spielen soll? Oder gar, dass agile Verfahren empfehlen, auf Dokumentation vollständig zu verzichten? Die Antwort hierauf lautet natürlich: nein. Agile Verfahren behaupten nicht, dass Dokumentation unnötig sei. Was sie allerdings tun, ist die Bedeutung von Dokumentation kritisch zu hinterfragen. Agile Verfahren empfehlen, nur solche Dokumentation anzufertigen, die den Aufwand der Erstellung auch wert ist. Sie weigern sich, Dokumentation als Selbstzweck zu begreifen: Dokumentation muss einen echten Nutzen haben, andernfalls ist sie verzichtbar. Was das konkret bedeutet, davon handelt dieses Buch. Das Buch kann Ihnen dabei helfen, in Ihren Projekten agiler zu werden, was Fragen der Dokumentation angeht.

Das Buch ist dabei auf keine spezielle agile Methode festgelegt. Es orientiert sich an agilen Praktiken im Allgemeinen. Sie können das Buch einsetzen, unabhängig davon, welche agile Methode Sie in Ihrem Projekt anwenden. Sie können das Buch insbesondere auch dann nut-

zen, wenn Sie in Ihrem Projektalltag bislang noch keine agilen Verfahren einsetzen, aber den Wunsch haben, im Hinblick auf die Dokumentation erste Schritte in Richtung eines agilen Vorgehens zu wagen. In diesem Fall wird Ihnen einiges von dem, was dieses Buch empfiehlt, möglicherweise ungewohnt vorkommen. Lassen Sie sich davon aber nicht irritieren. Natürlich bricht dieses Buch mit einigen traditionellen Vorstellungen von Dokumentation. Es zeigt aber auch, dass Sie, um agil vorzugehen, nicht all die Dinge über Bord werfen müssen, die Ihnen an einer soliden Dokumentation wichtig sind. Im Gegenteil: Das Buch soll Ihnen helfen herauszufinden, welche Praktiken zur Dokumentation sinnvoll sind, und diese dann auch in die Tat umzusetzen.

Die Tipps, die Ihnen dieses Buch gibt, gehen auf Erfahrungen zurück, die ich selbst, aber auch viele Kollegen in Softwareentwicklungsprojekten unterschiedlicher Größe gemacht haben. Natürlich sind die Anforderungen und Bedürfnisse in jedem Projekt unterschiedlich, gerade auch was die Dokumentation angeht. In jedem Fall kann Ihnen dieses Buch praktische Hinweise darauf geben, wie Sie die Vorteile agilen Vorgehens für die Planung, Erstellung und Verwendung der Dokumentation in Ihren Projekten nutzen können.

Ich wünsche Ihnen viel Erfolg dabei, die in diesem Buch beschriebenen Praktiken in Ihren Projekten anzuwenden.

Andreas Rüping
Hamburg, April 2013

Danke

Beim Schreiben eines Buchs ist es immer hilfreich, wenn man Feedback und auch die Ideen anderer einfließen lassen kann. Ganz besonders gilt dies beim Schreiben von Mustern (Patterns), weil Muster allgemeine Erfahrungen aus der Praxis repräsentieren (und eben nicht nur eigene Ideen). Ein großer Teil dieses Buchs besteht aus Mustern, die ich über Jahre hinweg immer wieder habe beobachten können. In diesem Sinne gilt mein Dank den Kollegen in vielen Projekten, mit denen zusammen ich eine Menge über agile Entwicklung gelernt habe, wie auch über gute Dokumentation.

Mein Dank gilt ebenso den Teilnehmern der europäischen Konferenz über Software Patterns (EuroPLoP), die mir über viele Jahre hinweg wertvolles Feedback zu meinen Dokumentationsmustern gegeben haben. Dieses Feedback hat mich vor zehn Jahren dazu motiviert, ein englischsprachiges Buch mit dem Titel *Agile Documentation* zu schreiben, und auch den Lesern dieses Buchs möchte ich meinen Dank aussprechen.

Während der letzten Jahre habe ich für die Deutsche Informatik-Akademie (DIA) Seminare zum Thema *Agile Dokumentation* gehalten. Mein Dank geht an die Organisatoren bei der DIA sowie an die Seminarteilnehmer für viele interessante Fragen und Anmerkungen.

Die Idee zu diesem deutschsprachigen Buch geht auf den dpunkt.verlag zurück. Mein herzlicher Dank gilt zunächst Frau Christa Preisendanz für ihre Anregung, ein solches Buch ins Auge zu fassen, wie auch für die Übernahme des Lektorats. Bedanken möchte ich mich außerdem bei Frau Vanessa Wittmer für ihr Engagement bei der Suche nach einem schönen Cover und bei Frau Nadine Thiele für die Unterstützung bei technischen Fragen der Textverarbeitung. Ebenfalls möchte ich den (anonymen) Gutachtern des Manuskripts für ihre hilfreichen Vorschläge danken.

Schließlich habe ich beim Schreiben dieses Buchs nicht nur von fachlichem Feedback, sondern auch von persönlicher Unterstützung profitiert. Ein herzliches Dankeschön geht daher an Freunde und Familie für ihre Unterstützung, die ganz unterschiedliche Formen angenommen hat, auf die ich mich aber immer verlassen konnte.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Agile Methoden legen in Softwareentwicklungsprojekten großen Wert auf direkte Kommunikation von Angesicht zu Angesicht und nehmen gegenüber umfangreicher schriftlicher Dokumentation eine eher kritische Haltung ein. Sie erkennen aber an, dass Dokumentation sinnvoll sein kann und manchmal auch benötigt wird.

Daraus resultiert die Frage, wie die Dokumentation in einem agilen Kontext sinnvoll gestaltet werden kann. Die Möglichkeiten dafür auszuloten ist Gegenstand dieses Buchs. Das Ziel dabei ist, bei der Planung, der Erstellung und der Nutzung von Dokumentation Strategien zu entwickeln, die sich am besten mit dem Wort *bedarfsgerecht* beschreiben lassen – die sich also an den tatsächlichen Bedürfnissen eines Projekts orientieren [Rüping 2011].

Ein bedarfsgerechtes Vorgehen bezieht sich zum einen darauf, *was* in einem Projekt schriftlich dokumentiert werden sollte (und was nicht). Es betrifft zum anderen die Frage, *wie* eine angemessene Dokumentation gestaltet werden kann. Beiden Fragen werden wir in diesem Buch nachgehen.

1.2 Historie der agilen Entwicklung

Um die Sichtweise der agilen Methoden verstehen zu können und speziell auch, was zu der betont kritischen Sicht auf umfangreiche Dokumentation geführt hat, lohnt sich ein Rückblick auf die 1990er-Jahre. In dieser Zeit haben die ersten agilen Verfahren ihren Ursprung. Blicken wir also einmal auf die damals gängigen oder zumindest doch empfohlenen Methoden in der Softwareentwicklung.

Prozesslastige Methoden in den 1990er-Jahren

Die 1990er-Jahre waren geprägt von einer Reihe von Methoden, die großen Wert auf definierte Prozesse und umfangreiche Dokumentation legten. Diese Methoden waren durch das Ziel gekennzeichnet,

den Ablauf eines Projekts im vorhinein detailliert zu planen und sich bei der anschließenden Umsetzung streng am Plan zu orientieren.

Fast alle diese Methoden basieren auf dem klassischen Wasserfallmodell, das den Prozess der Softwareentwicklung in verschiedene Phasen unterteilt (Grobspezifikation, Feinspezifikation, Entwurf, Realisierung, Test, Auslieferung), die typischerweise streng sequenziell durchlaufen werden. Dokumentation spielt beim Wasserfallmodell insofern eine große Rolle, als die Ergebnisse einer Phase in der Regel gründlich dokumentiert werden und die Dokumentation als Input für die nächste Phase bereitgestellt wird.

Typische Vertreter dieser Methoden sind die folgenden:

- Das V-Modell ist ein Vorgehensmodell zur Planung und Durchführung von Softwareentwicklungsprojekten, dessen Ursprünge ins Jahr 1979 zurückgehen. Es erweitert das traditionelle Wasserfallmodell, indem es jeder der ursprünglichen Phasen eine Test- oder Abnahmephase gegenüberstellt. Während das originale V-Modell durch relativ starre Prozesse gekennzeichnet war, lässt die 2005 in Deutschland eingeführte Variante XT bereits eine gewisse Anpassung der Prozesse an spezifische Gegebenheiten zu.
- Das 1986 erstmals vorgestellte Spiralmodell ist ein Vorgehensmodell, das ebenfalls auf dem Wasserfallmodell beruht, es aber bereits um die Idee der iterativen Entwicklung erweitert. Es sieht vor, typische Phasen wie Analyse, Entwurf, Realisierung und Test immer wieder zu durchlaufen und sich so schrittweise dem Projektziel zu nähern.
- Der 1998 erstmals veröffentlichte Rational Unified Process (RUP) [Kruchten 1998] umfasst ein Vorgehensmodell zur Softwareentwicklung, das zur Modellierung die bekannte Unified Modeling Language (UML) [Rumbaugh/Jacobsen/Booch 1998] einsetzt. Auch der Rational Unified Process weicht bereits deutlich vom klassischen Wasserfallmodell ab und empfiehlt stattdessen ein iteratives Vorgehen.

Ebenso prägend für die 1990er-Jahre waren Bemühungen, durch die Verbesserung der zugrunde liegenden Prozesse die Qualität von Software zu erhöhen. Resultat dieser Bemühungen waren verschiedene Prozessframeworks, die zwar kein spezielles Verfahren vorschreiben, aber doch dazu auffordern, innerhalb eines bestimmten Rahmens Prozesse und Vorgehen zu definieren. Zu den oben genannten Softwareentwicklungsmethoden sind diese Prozessframeworks in dem Sinne orthogonal, dass sie unterschiedliche Schwerpunkte setzen und

sich mit diesen kombinieren lassen. Bekannte Prozessframeworks sind die folgenden:

- Das Capability Maturity Model Integration (CMMI) ist ein Modell zur Beurteilung des Reifegrads sämtlicher Prozesse eines Unternehmens im Zusammenhang mit der Entwicklung und dem Betrieb von Software. Ziel ist die Definition, Planung, Implementierung und Qualitätssicherung dieser Prozesse. Das relativ starre ursprüngliche Modell (CMM) wurde 2003 durch den flexibleren Nachfolger (CMMI) abgelöst.
- Die Normenreihe ISO 9000 ff. legt Mindestanforderungen für die Qualitätssicherung fest. Die Normen beziehen sich auf Produktherstellung und Dienstleistungen generell und werden gelegentlich auch auf die Softwareentwicklung angewendet. Seit Mitte der 1990er-Jahre sind manche Softwareunternehmen bestrebt, sich in Bezug auf ihre Qualitätsstandards nach ISO 9000 zertifizieren zu lassen.

Sowohl die genannten Softwareentwicklungsmethoden als auch die Prozessframeworks zur Qualitätssicherung sind im Laufe der Zeit weiterentwickelt worden. In einigen Fällen drückt sich dies in der Namensgebung aus. So trat zum Beispiel an die Stelle des ursprünglichen V-Modells die Variante XT, und CMM wurde durch CMMI abgelöst.

Die traditionellen Entwicklungsmethoden, die ursprünglich auf dem Wasserfallmodell basierten, haben sich dabei tendenziell auf ein stärker iteratives Vorgehen hin bewegt. Die Prozessframeworks haben im Laufe der Zeit an Flexibilität gewonnen und tragen mittlerweile der Tatsache Rechnung, dass Projekte individuell geprägt sind und dass Prozesse an die individuellen Gegebenheiten angepasst werden müssen. Dementsprechend befinden sich Prozessframeworks wie beispielsweise CMMI heutzutage nicht mehr unbedingt im Widerspruch zu agilen Methoden [Glazer/Dalton/Anderson/Konrad/Shrum 2008].

Ende der 1990er-Jahre stellte sich die Situation allerdings noch anders dar. Die genannten Entwicklungsmethoden waren damals noch bestrebt, allgemeingültige Modelle für die Softwareentwicklung aufzustellen, und begannen sich nur langsam vom Wasserfallmodell zu lösen. Sie wurden zunehmend als starr, schwergewichtig und wenig flexibel wahrgenommen. Mit dem Aufkommen der Prozessframeworks rückten Prozessdefinitionen noch mehr in den Mittelpunkt der Softwareentwicklung, was den Eindruck von Schwergewichtigkeit und mangelnder Flexibilität weiter verstärkte.

Die Kombination aus wasserfallbasierten Entwicklungsmethoden und vergleichsweise starren Prozessdefinitionen hat in der Praxis immer wieder zu massiven Problemen geführt, die in manchen Fällen auch für das Scheitern großer Projekte ursächlich waren:

- Der Overhead, der durch die Prozesslastigkeit und die häufig damit verbundene Bürokratie dieser Methoden entsteht, kann immens sein und kann die Kreativität und Produktivität des Entwicklungsteams regelrecht ersticken. In manchen, nach schwergewichtigen Verfahren durchgeführten Projekten wurde mehr Aufwand darin investiert, den formalen Kriterien des Softwareentwicklungsprozesses zu genügen als tatsächlich Software zu entwickeln.
- Der Wunsch, jede noch so kleine Anforderung, aber auch jeden Entwicklungsschritt im Projekt zu dokumentieren, hat häufig zu Bergen von Dokumentation geführt, die niemand mehr hat lesen können.
- Die exakte Planung eines Projekts für mehrere Monate oder gar Jahre in die Zukunft ist schwierig bis unmöglich. Das Geschehen im Projekt ist schlecht vorhersehbar. Anforderungen ändern sich im Laufe der Zeit, und auch eine gutgemeinte Planung hat sich oft als unzuverlässig herausgestellt.
- Die häufig mit prozesslastigen Verfahren einhergehende sequenzielle Abfolge der einzelnen Projektphasen führt dazu, dass der Kunde die entstandene Software erst gegen Ende des Projekts zu sehen bekommt. Zu einem so späten Zeitpunkt ist es sehr schwer, das Feedback des Kunden auf die entwickelten Systeme noch zu berücksichtigen und möglicherweise noch eine Kurskorrektur vorzunehmen.

Als Konsequenz aus diesen Problemen kam gegen Ende der 1990er-Jahre immer häufiger die Forderung auf, sich von starren, schwergewichtigen Prozessen zu lösen und stattdessen mehr Wert auf Flexibilität zu legen.

Forderung nach mehr Flexibilität

Unter Flexibilität wird dabei die Fähigkeit verstanden, sich an wechselnde Bedingungen und Anforderungen anzupassen. In der Evolutionsbiologie ist diese Fähigkeit eine Grundvoraussetzung für das Überleben von Arten. Analog dazu ist diese Flexibilität auch in Entwicklungsprojekten ein entscheidendes Kriterium, das für den Erfolg oder Misserfolg eines Projekts ausschlaggebend sein kann. Im gleichen Zug wurde gefordert, die eigentliche Erstellung der Software wieder mehr in den Mittelpunkt des Projektgeschehens zu stellen und hingegen Prozesse und Methoden als Mittel zum Zweck zu betrachten und

immer wieder auf ihren Sinn und Zweck hin zu überprüfen. Die Idee eines agilen Vorgehens war geboren.

Agiles Manifest

In der Folge haben einige der frühen »Agilisten« begonnen, Praktiken agilen Vorgehens zu entwickeln und in ihrer alltäglichen Projektpraxis anzuwenden. Im Februar 2001 trafen sich 17 dieser Experten zu einem Workshop mit dem Ziel, ihre Erfahrungen mit dem neuen Vorgehen auszutauschen. Bei den Diskussionen im Rahmen dieses Workshops hat sich dann das herauskristallisiert, was heute als die Prinzipien agiler Entwicklung verstanden wird. Das Ergebnis des Workshops ist später als das Agile Manifest bekannt geworden [Agile Alliance 2001]. Abbildung 1–1 zeigt die Kernaussage des Agilen Manifests in der deutschen Übersetzung.

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 Veränderungen
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.

Abb. 1–1
Agiles Manifest

Neben diesen grundlegenden Prinzipien benennt das Agile Manifest auch eine Reihe von Kernpraktiken, die sich für die konkrete Anwendung in der täglichen Projektpraxis eignen. Einige davon sind die folgenden:

- Software wird inkrementell entwickelt. Das Team liefert kontinuierlich die jeweils entwickelten Inkremente an den Kunden aus.
- Die Menge funktionierender Software dient als primäres Fortschrittsmaß.

- Fachexperten und Entwickler kooperieren eng miteinander, idealerweise auf täglicher Basis.
- Die wichtigste Kommunikationsform der Projektbeteiligten ist das Gespräch von Angesicht zu Angesicht.
- Einfachheit ist ein Prinzip. Einfachen Lösungen wird der Vorzug vor komplizierten Lösungen gegeben.
- Das Team reflektiert regelmäßig sein Vorgehen und passt das Vorgehen gegebenenfalls an.

Wenngleich das Agile Manifest diese und eine Reihe weiterer Praktiken empfiehlt, so beschreibt es dennoch keine spezifische Entwicklungsmethode. Durch die Formulierung grundlegender Prinzipien gibt es vielmehr den Rahmen vor, in dem sich die agile Softwareentwicklung bewegt. Für die konkrete Ausgestaltung einzelner Methoden bleibt dabei noch viel Spielraum. Dieser Spielraum ist auch notwendig, weil Projekte sehr unterschiedlich sind, zum Beispiel im Hinblick auf Anzahl der beteiligten Personen, Laufzeit, Umfang der Funktionalität, Komplexität, Technologie und Kritikalität. Nicht alle Projekte können nach derselben Methode durchgeführt werden. Verschiedene agile Methoden interpretieren daher den Rahmen, den das Agile Manifest vorgibt, auf durchaus unterschiedliche Art und Weise. Viele der Methoden weisen durchaus Ähnlichkeiten auf und können auch gut miteinander kombiniert werden, dennoch gibt es auch eine Vielzahl von Unterschieden.

Verschiedene agile Methoden

Im Laufe der Zeit hat sich eine Reihe von agilen Methoden zur Softwareentwicklung etabliert, die in ihrer konkreten Ausgestaltung unterschiedliche Akzente setzen und auch unterschiedliche Verbreitung erfahren haben.

- Eine der frühesten agilen Methoden ist eXtreme Programming, häufig auch einfach als XP bezeichnet [Beck 2000; Wolf/Roock/Lippert 2005]. Seit dem Jahr 2000 hat XP einige Popularität erreicht. XP basiert auf einer Reihe bewährter Praktiken, zu denen beispielsweise inkrementeller Entwurf, testgetriebene Entwicklung sowie kontinuierliche Integration gehören. Bekannt geworden ist XP vor allem für die Einführung von Pair Programming.
- Bei Crystal [Cockburn 2002] handelt es sich um eine Familie von Methoden, die abhängig von der Anzahl der Projektbeteiligten und von der Kritikalität des Projekts verschiedene Strategien zur Kommunikation und zur Qualitätssicherung vorschlagen.
- Scrum ist mittlerweile die bekannteste agile Methode, zumindest im deutschsprachigen Raum [Schwaber/Beedle 2008; Pichler 2007; Cohn 2010; Wolf/van Solingen/Rustenburg 2010; Gloger 2011;

Wirdemann 2011; Röpstorff/Wiechmann 2012]. Kennzeichen von Scrum ist ein iteratives Vorgehen, das sich in regelmäßigen Intervallen, den sogenannten Sprints, ausdrückt. In diesen Sprints werden jeweils Inkremente der Software entwickelt und ausgeliefert. Am Ende jeden Sprints steht unter anderem eine Retrospektive zur Überprüfung des eigenen Vorgehens.

- Feature-Driven Development (FDD) ist ein leichtgewichtiges Verfahren, das ausgehend von einem Gesamtmodell eine Liste einzelner Features entwickelt und die gesamte Entwicklung dann in die Implementierung der einzelnen Features herunterbricht [Palmer/Felsing 2002]. FDD ist vielleicht weniger »revolutionär« als beispielsweise XP oder Scrum, befindet sich aber durchaus im Einklang mit den Prinzipien agiler Entwicklung.
- Eine der neueren agilen Methoden ist Kanban [Anderson 2011], das seinen Ursprung in den Prinzipien des Lean Development hat [Poppendieck/Poppendieck 2003]. Kanban versucht durch die Reduktion paralleler Arbeit den gesamten Fluss aller Projekttätigkeiten zu erhöhen.

Einige dieser Methoden haben mittlerweile einen großen Bekanntheitsgrad erlangt und gehören zum Standardrepertoire heutiger Softwareentwicklung. Insbesondere mit XP, Scrum und Kanban sind bereits viele Projekte erfolgreich durchgeführt worden [Wolf 2011].

Neben diesen Methoden haben noch weitere Techniken aufgrund ihres agilen Charakters eine gewisse Bekanntheit erlangt. Zu nennen sind hier insbesondere die folgenden:

- Bei testgetriebener Entwicklung (Test-Driven Development) handelt es sich um eine Strategie, die das Testen von Software in den Vordergrund stellt (nachdem es über lange Zeit hinweg in vielen Projekten nur ein Schattendasein gefristet hatte). Das Prinzip ist dabei, der Entwicklung bestimmter Funktionen immer die Entwicklung entsprechender Tests vorausgehen zu lassen [Westphal 2005]. Testgetriebene Entwicklung ist als Einzeltechnik Bestandteil vieler agiler Verfahren.
- Agile Modeling ist ein Verfahren, das sich auf die Modellierungsaspekte in Projekten konzentriert und dafür leichtgewichtige und durch starke Interaktion geprägte Prozesse empfiehlt [Ambler 2002].
- DevOps ist ein vergleichsweise neuer Ansatz, der das Ziel verfolgt, ein agiles Vorgehen auf den Betrieb von Software auszudehnen [Peschlow 2011]. Schwerpunkt ist dabei die enge Kooperation zwischen Entwicklern und Betriebsabteilung. Durch die regelmäßige

Auslieferung von Software geht deren Inbetriebnahme in einen einfachen, erprobten und reproduzierbaren Prozess über.

Zum Thema Dokumentation gibt es im Agilen Manifest keine Vorschriften, was das konkrete Vorgehen im Projekt angeht. Hingegen vermittelt das Agile Manifest eine grundsätzliche Haltung, die zwar den Wert von Dokumentation anerkennt, diesen Wert aber auch gegenüber anderen Projektzielen relativiert. Gerade vor dem historischen Hintergrund ist dies nur zu verständlich.

Die schwergewichtigen Methoden der 1990er-Jahre hatten allesamt ein gehöriges Maß an Dokumentation mit sich gebracht. Ziel war damals gewesen, jegliche Information, die in einer bestimmten Projektphase benötigt wird, vorher schriftlich zu fixieren. In manchen Projekten wurden über einen Zeitraum von Jahren nur Spezifikationen und Konzeptpapiere erstellt, bevor überhaupt eine einzige Zeile Code programmiert wurde. Dabei sind in der Regel große Mengen an Dokumentation entstanden, die in ihrer Fülle gar nicht gelesen werden konnten und außerdem schneller veraltet sind, als es jedem Projektbeteiligten recht sein konnte. Insider sprechen in einem solchen Fall von *write-only documentation*.

Dokumentation als Mittel zum Zweck

Ein gemeinsames Merkmal aller agiler Methoden ist, dass sie derartige Szenarien vermeiden wollen. Der Grund hierfür liegt in der Erkenntnis, dass Dokumentation nicht automatisch zum Verständnis der Materie führt, wie auch Abbildung 1–2 andeutet. Beim Schreiben wandert nur ein Teil des in den Köpfen vorhandenen Wissens tatsächlich auch ins Dokument, manches hingegen bleibt unausgesprochen. Gleichermaßen erfassen auch gründliche Leser nicht immer alles, was in einem Dokument beschrieben ist.

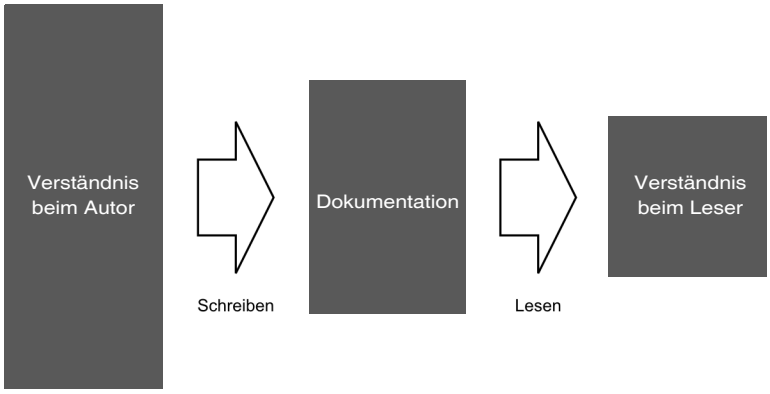

Abb. 1–2
Dokumentation vs. Verständnis