Clean Agile. Back to Basics

von Robert C. Martin (Buchzusammenfassung)

Patrick Bucher

05.09.2021

Inhaltsverzeichnis

Hinweis zur Übersetzung: Hierbei handelt es sich um die deutschsprachige Übersetzung einer englischsprachigen Buchzusammenfassung, die über das englischsprachige Original geschrieben worden ist. Manche Begriffe wurden frei auf Deutsch übersetzt, andere im englischen Original belassen. Auf eigene Übersetzungen folgt beim ersten Auftreten jeweils der Originalbegriff in Klammern.

Im Englischen wurde der Begriff "agile software development" mit "Agile" abgekürzt. Deshalb steht auch in dieser Übersetzung der substantivierte Begriff "Agile" geschrieben, der englisch auszusprechen ist, wo "agile Softwareentwicklung" gemeint ist.

1 Einführung in Agile

Das *Manifest für Agile Softwareentwicklung* (*Agile Manifesto*) ist das Ergebnis eines Treffens von 17 Experten für Software anfangs 2001 als Reaktion auf schwergewichtige Prozesse wie Wasserfall (*Waterfall*). Seither erfreute sich Agile weiter Verbreitung und wurde auf verschiedene Arten erweitert – leider nicht immer im Sinne der ursprünglichen Idee.

1.1 Geschichte von Agile

Die grundlegende Idee von Agile – die Arbeit mit kleinen Zwischenzielen, wobei der Fortschritt gemessen wird – könnte so alt sein wie unsere Zivilisation. Es ist auch möglich, dass agile Praktiken in den Anfängen der Softwareentwicklung verwendet worden sind. Die Idee des wissenschaftlichen Managements (*Scientific Management*), welche auf dem Taylorismus basiert,

von oben herab organisiert ist und auf eine detaillierte Planung setzt, war zu dieser Zeit weit verbreitet in der Industrie, wodurch sie in Konflikt zu den vor-agilen (*Pre-Agile*) Praktiken war, die zu dieser Zeit in der Softwareentwicklung so weit verbreitet waren.

Wissenschaftliches Management war für Projekte geeignet, bei denen Änderungen teuer waren und zu denen es eine genau definierte Problemdefinition mit extrem spezifischen Zielen gab. Vor-agile Praktiken andererseits eigneten sich gut für Projekte, bei denen Änderungen günstig, das Problem nur teilweise definiert und die Ziele informell spezifiziert waren.

Leider gab es zu dieser Zeit keine Diskussion darüber, welcher Ansatz für Softwareprojekte der bessere war. Stattdessen fand das Wasserfallmodell weite Verbreitung, das ursprünglich von Winston Royce in seinem Fachartikel *Managing the Development of Large Software Systems* als Strohmannargument aufgebaut worden war, um dessen Unzulänglichkeit zu demonstrieren. Das Wasserfallmodell mit seinem Fokus auf Analyse, Planung und genaues Einhalten von Plänen war ein Abkömmling des wissenschaftlichen Managements, nicht von vor-agilen Praktiken.

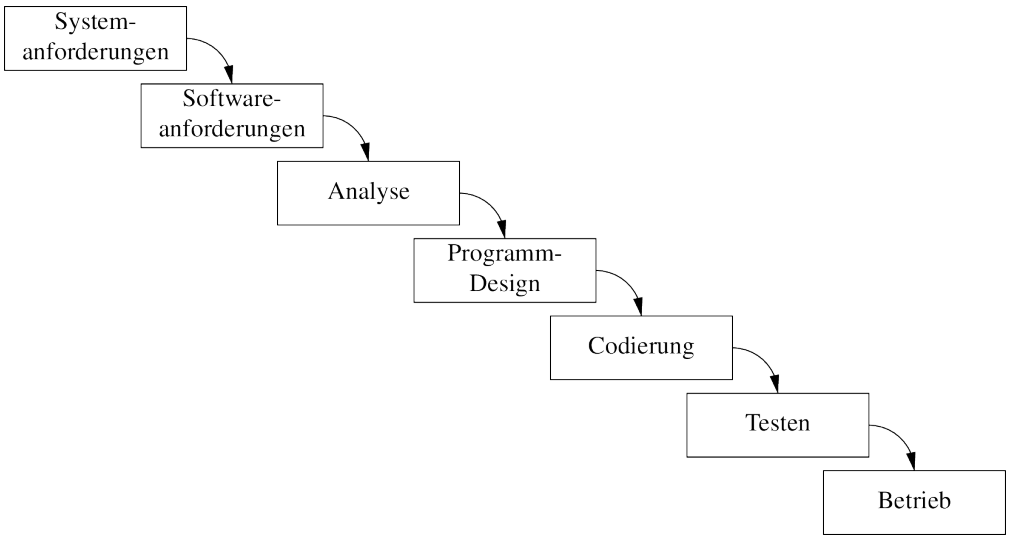

Abbildung 1: Das Wasserfallmodell

Das Wasserfallmodell dominierte die Industrie ab den 1970er-Jahren für fast 30 Jahre. Seine aufeinanderfolgenden Phasen von Analyse, Design und Umsetzung sah vielversprechend aus für Entwickler, welche in endlosen "Programmieren und Korrigieren"-Zyklen (*"code and fix" cycles*) arbeiteten, und dabei nicht einmal die vor-agile Disziplin aufbrachten.

Was auf dem Papier gut aussah – und zu vielversprechenden Ergebnissen nach der Analyse- und Design-Phase führte – scheiterte oft kläglich in der Umsetzungsphase. Diese Probleme wurden jedoch auf eine schlechte Ausführung geschoben, und der Wasserfall-Ansatz selber wurde nicht kritisiert. Stattdessen wurde dieser Ansatz so dominant, dass auf neue Entwicklungen in der Software-Industrie wie strukturierte oder objektorientierte Programmierung bald

die Disziplinen der strukturierten und objektorientierten Analyse und des strukturierten und objektorientierten Designs folgten – und so perfekt zur Wasserfall-Denkweise passten.

Einige Befürworter dieser Ideen begannen jedoch das Wasserfallmodell mitte der 1990er-Jahre in Frage zu stellen, wie z.B. Grady Booch mit seiner Methode des objektorientierten Designs (OOD), die Entwurfsmuster-Bewegung (*Design Pattern movement*), und die Autoren des *Scrum*-Papers. Kent Becks Ansätze des *Extreme Programming* (XP) und der testgetriebenen Entwicklung (*Test-Driven Development, TDD*) der späten 1990er-Jahre waren eine klare Abkehr vom Wasserfallmodell hin zu einem agilen Ansatz. Martin Fowlers Gedanken zum *Refactoring* mit dessen Betonung von kontinuierlicher Verbesserung passt sicherlich schlecht zum Wasserfallmodell.

1.2 Das Manifest für Agile Softwareentwicklung

17 Vertreter verschiedener agiler Ideen – Kent Beck, Robert C. Martin, Ward Cunningham (XP), Ken Schwaber, Mike Beedle, Jeff Sutherland (Scrum), Andrew Hunt, David Thomas ("Pragmatic Programmers") und weitere – trafen sich anfangs 2001 in Snowbird, Utah, um ein Manifest zu erarbeiten, dass die gemeinsame Essenz all dieser leichtgewichtigen Ideen erfassen sollte. Nach zwei Tagen konnte ein breiter Konsens erreicht werden:

> 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.

Das *Manifest für Agile Softwareentwicklung* wurde nach dem Treffen auf agilemanifesto.org veröffentlicht, wo es noch immer unterschrieben werden kann. Die 12 Prinzipien hinter dem Agilen Manifest wurden nach den beiden Wochen, die auf das Treffen folgten, in gemeinsamer Arbeit verfasst. Dieses Dokument erläutert die vier Werte, die im Manifest aufgeführt sind, und verleiht ihnen eine Richtung; es legt dar, dass diese Werte wirkliche Konsequenzen haben.