entwickler.press

Agile Muster und Methoden

Agile Softwareentwicklung maßgeschneidert

Manfred Steyer

Manfred Steyer

Agile Muster und Methoden

Manfred Steyer

Agile Muster und Methoden

Agile Softwareentwicklung maßgeschneidert

entwickler.press

Manfred Steyer
Agile Muster und Methoden

ISBN: 978-3-86802-241-4

Ein Imprint der Software & Support Verlag GmbH

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.

Ihr Kontakt zum Verlag und Lektorat:
Software & Support Verlag GmbH
entwickler.press
Geleitsstraße 14
60599 Frankfurt am Main
Tel: +49(0) 69 63 00 89 - 0
Fax: +49(0) 69 63 00 89 - 89
lektorat@entwickler-press.de
http://www.entwickler-press.de

Lektorat: Sebastian Burkart
Korrektorat: Katharina Klassen, Frauke Pesch
Layout: SatzWERK, Siegen (www.satz-werk.com)
Umschlaggestaltung: Maria Rudi
Belichtung, Druck & Bindung:
M.P. Media-Print Informationstechnologie GmbH, Paderborn

Inhaltsverzeichnis

1 Einleitung

Agile Methoden verfolgen als Gegenbewegung zu klassischen Ansätzen das Ziel, die Softwareentwicklung flexibler zu gestalten, sodass jederzeit – auch gegen Ende des geplanten Projekts – auf notwendige Änderungen reagiert werden kann. Das Ziel ist es, durch ein schlankes Vorgehen sowie durch motivierte Mitarbeiter und den gezielten Einsatz agiler Techniken Softwareprojekte erfolgreich abzuwickeln.

Die agile Bewegung hat in den letzten Jahren einige Methoden hervorgebracht, darunter *Extreme Programming* in zwei Versionen, *Scrum* oder die *Crystal*-Methodenfamilie mit *Crystal Clear*, *Crystal Yellow* und *Crystal Orange*, um nur einige zu nennen. Obwohl diese Methoden gemeinsame, durch das Agile Manifesto festgelegte Werte und Prinzipien teilen, unterscheiden sie sich dennoch in einigen Aspekten, sodass sich heutzutage Softwareentwicklungsteams, die nach agilen Grundsätzen vorgehen möchten, mit unterschiedlichen Meinungen und Ansätzen konfrontiert sehen.

Die Idee hinter diesem Buch ist die Bereitstellung eines Musterkatalogs, der Projektteams eine Grundlage für die Erstellung einer individuellen agilen Methode bietet. Die Muster dieses Musterkatalogs sollen dabei auf Handlungsempfehlungen ausgewählter agiler Methoden basieren. Bei diesen handelt es sich um Extreme Programming in beiden Versionen [Beck99], [Beck04], Scrum [Schw04], Crystal Clear [Coc05], [Coc06] sowie Crystal Orange [Coc98], [Coc06]. Unterschiedliche Meinungen und Ansätze werden dabei herausgearbeitet und verglichen.

1.1 Aufbau dieses Buchs

Da agile Methoden eine Gegenbewegung zu klassischen Vorgehensmodellen darstellen, werden in Kapitel 2 zunächst zwei populäre Vertreter dieser klassischen Vorgehensmodelle vorgestellt. Nach einer Diskussion deren Vor- und Nachteile wendet sich Kapitel 3 anschließend ausgewählten agilen Methoden zu, die die zuvor aufgezeigten Nachteile zu kompensieren versuchen. Zuerst wird dazu auf das Agile Manifesto eingegangen, das die gemeinsamen Werte und Prinzipien aller agilen Methoden wiederspiegelt. Anschließend werden die agilen Methoden Extreme Programming, Scrum sowie die Crystal-Methodenfamilie mit Crystal Clear und Crystal Orange vorgestellt, da diese zur Erarbeitung des in diesem Werk beschriebenen Musterkatalogs analysiert wurden.

Aus Kapitel 4 ist der agile Musterkatalog ersichtlich. Dieser basiert auf den in Kapitel 3 vorgestellten agilen Methoden, wobei sich jedes der darin befindlichen Muster auf eine oder auf mehrere Handlungsempfehlungen dieser Methoden bezieht. Anschließend wird in Kapitel 5 gezeigt, dass die ausgewählten agilen Methoden in Hinblick auf ihre Handlungsempfehlungen durch den Musterkatalog beschrieben werden können. Dieser Umstand soll das Merkmal der Vollständigkeit des erarbeiteten Musterkatalogs sicherstellen, da somit gezeigt wird, dass dieser sämtliche Handlungsempfehlungen der betrachteten Methoden beinhaltet. Danach folgen eine Zusammenfassung in Kapitel 6 und Informationen zur Einrichtung einer agilen Arbeitsumgebung im Anhang.

2 Ausgewählte klassische Vorgehensmodelle

Da die Unzulänglichkeiten klassischer Vorgehensmodelle der Grund für die Entstehung agiler Methoden waren, werden in diesem Kapitel zwei Vertreter dieser klassischen Ansätze vorgestellt: das Wasserfallmodell sowie das Spiralmodell.

Das Wasserfallmodell wird in diesem Kapitel vorgestellt, da es auf den ersten Blick plausibel aufgebaut ist sowie den Vorgehensweisen in anderen (Ingenieurs-)Disziplinen wie dem Bau von Häusern oder Brücken ähnelt und trotzdem vollständig zu agilen Werten und Prinzipien in Wiederspruch steht. Darüber hinaus gehen mit diesem Modell einige Nachteile, die durch agile Methoden kompensiert werden sollen, einher. Somit hilft es beim Nachvollziehen der Gründe, die zur Entwicklung agiler Methoden geführt haben.

Das Spiralmodell wurde für dieses Kapitel aufgegriffen, da es einen ersten Versuch zur Kompensierung der mit dem Wasserfallmodell einhergehenden Nachteile darstellt und es bereits durch das vorsehen von Zyklen ansatzweise eine zentrale Idee agiler Methoden beinhaltet.

2.1 Wasserfallmodell

Das Wasserfallmodell, das bereits 1970 zum ersten Mal beschrieben wurde [Roy70], sieht vor, dass ein Softwareentwicklungsprojekt in Phasen unterteilt wird, die streng sequenziell auszuführen sind. Jede dieser Phasen kann somit separat geplant und mit jeweils dafür ausgebildeten Experten besetzt werden. In jeder Phase werden bestimmte Artefakte, beispielsweise Analyse- oder Entwurfsspezifikationen, erstellt, die die Grundlage für die nächste Phase darstellen. Das erweiterte Wasserfallmodell (Abb. 2.1) sieht darüber hinaus auch die Möglichkeit von Rückkoppelungen vor. Treten bei-

spielsweise im Zuge des Tests Probleme auf, können diese an die jeweiligen Entwickler rückgemeldet werden, was zur Folge hat, dass erneut mit der Phase *Coding* fortgefahren wird.

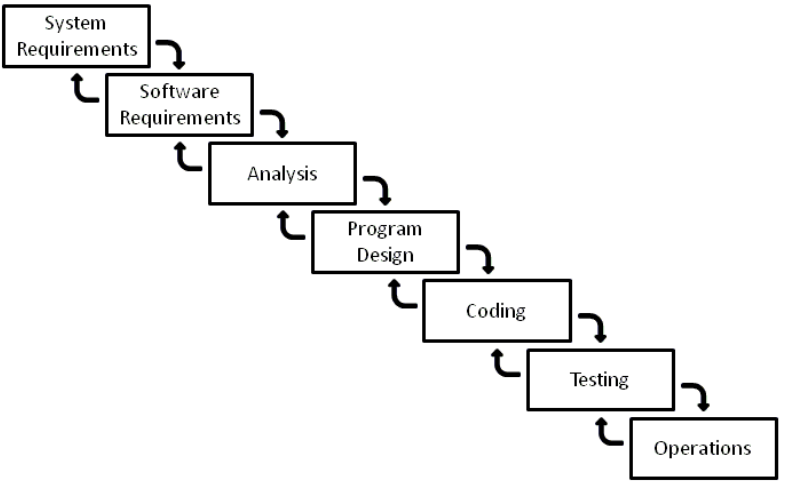

Abbildung 2.1: Erweitertes Wasserfallmodell aus [Roy70]

Diese Vorgehensweise wird als risikobehaftet angesehen, da beispielsweise ein Fehler, der im Zuge der Phase *Testing* entdeckt wird, zu einer kaskadierenden Rückkopplung bis zum Beginn des Prozesses führen kann und somit eine bis zu hundertprozentige Überziehung des Kosten- bzw. Zeitrahmens zur Folge hat. Ein Beispiel ist dazu das Entdecken von Performanceproblemen im Zuge des Testens: Dieser Umstand führt zu einer Rückkopplung zur Phase *Design*, in der gegebenenfalls der Entschluss, die Anforderungen zu überdenken, gefasst wird. Illustriert wird das durch Abbildung 2.2.

Um dieses Problem zu vermeiden, schlägt [Roy70] vor, eine erste *Programmdesign*-Phase bereits vor der Analyse stattfinden zu lassen. Darüber hinaus spricht er sich für eine massive Dokumentation in sämtlichen Phasen aus:

„The first rule of managing software development is ruthless enforcement of documentation requirements. […] Management of software is simply impossible without a very high degree of documentation."

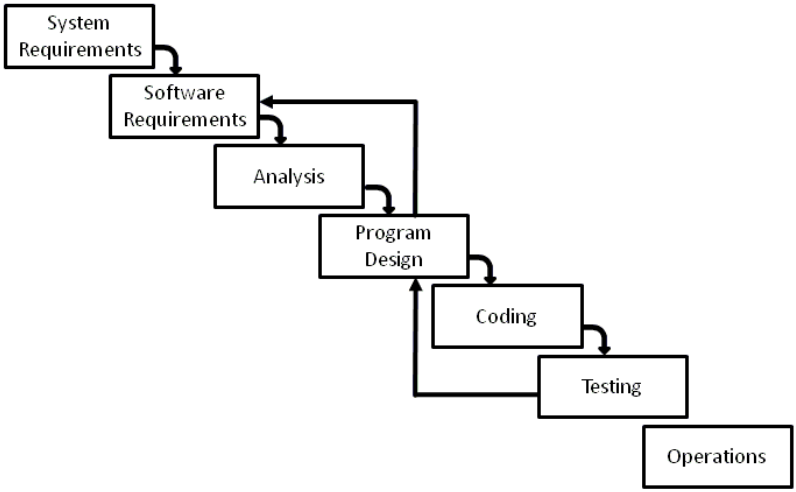

Abbildung 2.2: Rückkopplungskaskade im Wasserfallmodell aus [Roy70]

Weiterhin empfiehlt er, ein Softwareprojekt zweimal durchzuführen, bevor es an den Kunden ausgeliefert wird, wobei die erste Durchführung eine Minatur der zweiten darstellen soll und in etwa 1/4 bis 1/3 der für das Projekt beanspruchten Ressourcen dafür einzuplanen sind. Diese erste Durchführung soll als Simulation sowie zur Erprobung und Anpassung des Designs dienen und ist somit vergleichbar mit der Erstellung eines vertikalen Prototyps.

Die Vorteile dieses Modells liegen zum einen in der leichten Verständlichkeit sowie im damit einhergehenden geringen Verwaltungsaufwand bei Projekten, in denen keine Änderungen auftreten. Die Nachteile sind, dass Änderungswünsche durch die mit ihnen Hand in Hand gehenden Rückkopplungen teuer sind sowie, dass sich durch das strenge sequenzielle Vorgehen die Durchlaufzeit maximiert. Darüber hinaus besteht die Gefahr, dass Aspekte der Dokumentation zu Lasten der Entwicklung überbewertet werden. Ferner wird der Kunde erst am Ende des Projekts mit dem entwickelten Produkt konfrontiert. Sein Feedback kann somit erst im Zuge der Implementierung der nächsten Version berücksichtigt werden.

Das Wasserfallmodell scheint nur geeignet zu sein, wenn zum einen das Team in der Lage ist, endgültige Definitionen und Entwürfe im ersten Durchgang zu erstellen sowie wenn der Auftraggeber in der Lage ist, Anforderungen an das Produkt vollständig und endgültig vorzugeben. Die Erfahrung hat jedoch gezeigt, dass das nur sehr selten der Fall ist.

2.2 Spiralmodell

Beim von Böhm 1988 [Boeh88] vorgeschlagenen Spiralmodell handelt es sich um einen risikogetriebenen Ansatz zur Softwareentwicklung, der die Probleme eines wasserfallartigen Vorgehens kompensieren sollte. Dieses Modell sieht eine zyklische Planung vor, wobei sich jeder Zyklus in vier Schritte, die durch die vier Quadranten in Abbildung 2.3 visualisiert werden, teilt.

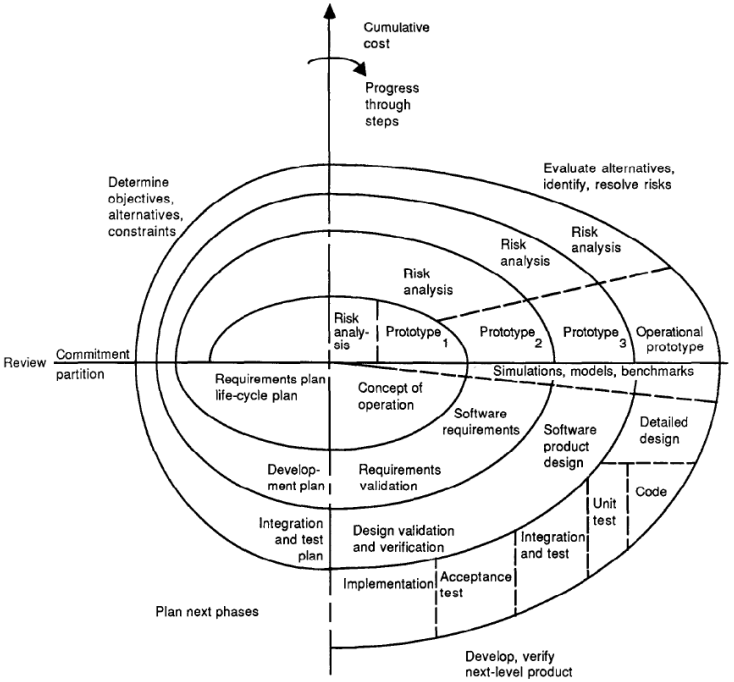

Abbildung 2.3: Spiralmodell aus [Boeh88]

Zunächst werden die Ziele des Teilprodukts, das im aktuellen Zyklus erstellt werden soll, sowie alternative Möglichkeiten zur Erreichung dieser Ziele und Randbedingungen identifiziert. Anschließend wird eine Risikoanalyse durchgeführt.

Das Ziel dieser Risikoanalyse ist das Evaluieren der identifizierten Alternativen und das Ausmachen von Risiken sowie das Finden von Wegen, um diese zu umgehen. Dazu können Prototypen, Simulationen, Modelle oder Benchmarks eingesetzt werden. Verbleiben Risiken, so wird der aktuelle Zyklus wiederholt und gegebenenfalls ein weiterer Prototyp entwickelt. Ansonsten wird mit der Planung und Durchführung des nächsten Zyklus fortgesetzt. Im Zuge der Planung wird auch das für den Folgezyklus zu verwendende Prozessmodell definiert. Auf diese Art und Weise wird pro Zyklus ein Artefakt, das mit einer Phase im Wasserfallmodell assoziiert wird, erstellt (Softwareanforderungen, Softwareproduktentwurf, Detailentwurf). Im letzten Zyklus erfolgen die Codierung und Erstellung von Modultests, die Integration inklusive Integrationstests, Abnahmetests sowie die Einführung in die vorgesehene Umgebung (Implementierung).

Zu den Vorteilen des Spiralmodells zählt, dass Fehler und Risiken frühzeitig erkannt und eliminiert werden sowie dass durch die Betrachtung von Alternativen die Wiederverwendung von Softwarekomponenten unterstützt wird. Dem steht ein hoher Managementaufwand gegenüber, da immer wieder neue Entscheidungen über den weiteren Prozessablauf zu treffen sind, weswegen dieses Modell für kleine Projekte nicht zielführend ist [Balz08]. Darüber hinaus wird auch mit diesem Modell versucht, die Planung so genau wie möglich zu gestalten, sodass anschließend das Codieren, Testen und Einführen wasserfallartig abgewickelt werden können. Somit ergeben sich die bereits vom Wasserfallmodell bekannten Nachteile.