|  |
| --- |
| Vergleich der drei agilen Softwareentwicklungsprozesse Crystal, Scrum und Kanban  STUDIENARBEIT  für die Prüfung zum  Bachelor of Engineering/Bachelor of Science  des Studiengangs Informatik Studienrichtung Angewandte Informatik  an der  Dualen Hochschule Baden-Württemberg Karlsruhe  von  Vorname Nachname  Abgabedatum  Matrikelnummer Matrikelnummer  Kurs Kursbezeichnung  Ausbildungsfirma Firmenname, Stadt  Betreuer [der Ausbildungsfirma] Titel Vorname Nachname  [Gutachter der Studienakademie Titel Vorname Nachname] |

Erklärung

gemäß § 5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.

Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Ort, Datum Unterschrift

Zusammenfassung

Abstract

Inhaltsverzeichnis

Zusammenfassung 3

Abstract 3

Inhaltsverzeichnis 4

Abbildungverzeichnis 5

Tabellenverzeichnis 5

Abkürzungsverzeichnis 5

1. Einleitung 6

2. Agile Softwareentwicklung 7

2.1 Allgemein 7

2.1.1 Einführung 7

2.1.2 Das agile Manifest 9

2.1.3 Agile Prinzipien/Methoden 10

2.2 Crystal Clear 12

2.3 Scrum 15

2.4 Kanban 18

4. Vergleichskriterien für die Methoden 19

5. Wahl der Tools 20

6. Wahl des Projektes 20

7. Vorbereitung der Durchführung 20

8. Durchführung des Projektes 21

8.1 Crystal Clear 21

8.2 Scrum 21

8.3 Kanban 21

9. Ergebnisse 22

9.1 Analyse/Ursachenforschung 22

9.1.1 Crystal Clear 22

9.1.2 Scrum 22

9.1.3 Kanban 22

9.2 Auswertung der Tools 22

9.3 Gegenüberstellung der agilen Prozesse 22

10. Fazit 23

Abbildungverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

Tabellenverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

Abkürzungsverzeichnis

# Einleitung

# Agile Softwareentwicklung

## Allgemein

### Einführung

Agil stammt vom Lateinischen Wort agilis und bedeutet soviel wie „von großer Beweglichkeit zeugend; regsam und wendig“ (1). Mit Softwareentwicklung ist die „Verbesserung vorhandener oder Erarbeitung neuer Software“ (1) gemeint.

In Kombination wird der Begriff der agilen Softwareentwicklung meist als Gegensatz zur traditionellen Softwareentwicklung verwendet und bezieht sich im Allgemeinen auf den gesamten Entwicklungs- und Managementprozess. Hierbei wird oft hauptsächlich ein Vergleich mit dem relativ starren Wasserfallmodell vorgenommen, welches 1970 im Artikel „Managing the Development of Large Software Systems“(2) von Dr. Winston Royce erstmalig formell beschrieben wurde. Dabei erklärt Royce bereits, dass lineares Arbeiten für Softwareentwicklung ungeeignet ist. Stattdessen empfiehlt er einen iterativen Prozess, der heute in verschiedensten Ausführungen in allen Beispielen für Agile Softwareentwicklung zu finden ist.

Das Abheben von alten, starren Modellen ist jedoch nicht das Hauptziel der Agilen Entwicklung. Prinzipiell soll der gesamte Prozess flexibler gestaltet werden, um die Probleme aus klassischen Modellen zu verringern oder ganz zu vermeiden. Zu diesen Problemen zählen beispielsweise die Überdokumentation, die fehlende Reaktionsfähigkeit gegenüber sich ändernden Anforderungen und Ressourcen, gesetzlichen Rahmenbedingungen oder spät erkennbaren Risiken. Weitere Probleme sind große Fehler bei Zeitschätzungen am Anfang des Projekts sowie mangelnde Kommunikation und Wissens- und Erfahrungsaustausch zwischen festgelegten Rollen innerhalb des Projekts.

Abbildung 1 gibt einen Überblick über die verschiedenen Werte wie Transparenz oder Anpassungsfähigkeit, aber auch unterschiedliche Methoden und Hilfsmittel der Agilen Softwareentwicklung. Das Hauptziel der schnelleren Lieferung von „Working Software“ ist ebenso klar erkennbar wie die kontinuierliche Evaluation des Prozesses und der Ergebnisse.



### Das agile Manifest

Im Februar 2001 haben 17 Softwareentwickler die Richtlinien der agilen Softwareentwicklung im agilen Manifest formuliert, welches bis heute Gültigkeit besitzt. Dadurch sollte eine kleine Revolution in der Softwareentwicklung angestoßen werden.

„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.“

Individuen und Interaktionen stehen für das selbstbestimmt Arbeiten an einem Projekt und die persönliche Kommunikation, welche oft falsche Verständigung minimiert.

Funktionierende Software hat beim agilen Prozess eine deutlich höhere Priorität als eine Umfassende Dokumentation, denn das stellt den Kunden vorrangig zufrieden.

Die enge Zusammenarbeit mit dem Kunden ermöglicht eine Schnelle Rückfrage bei Unklarheiten und hilft Fehler frühzeitig zu erkennen und zu vermeiden.

Das Reagieren auf Veränderungen bezieht sich unter anderem auf eine Anpassung der Anforderungen während der Entwicklung und auf das Anpassen der Arbeitsweise.

Zusätzlich zum agilen Maifest wurden 12 Prinzipien zur agilen Softwareentwicklung aufgestellt. „Sie erklären detaillierter die Werte und Prinzipien der "Agilisten". (http://www.heise.de/developer/meldung/10-Jahre-Agiles-Manifest-zur-Geburt-agiler-Softwareentwicklung-1188299.html)

Darin enthalten sind unter anderem, die Zusammenarbeit von Fachexperten und Entwicklern, das selbständige Arbeiten, die regelmäßige Reflektion der Arbeit, die möglichst frühe Auslieferung einzelner funktionierender Softwareteile und die Anpassung von Veränderungen in das Projekt zur Wettbewerbssteigerung.

### Agile Prinzipien/Methoden

Neben den abstrakten Werten kann man die Prinzipien eher als die „gelebten“ Werte bezeichnen (Lundak, 2009). Während die Werte sehr stark im Unternehmen verankert sein müssen, werden die Prinzipien am besten von der Basis erarbeitet, um somit für diejenigen, die sie ausführen, eine höhere Akzeptanz zu schaffen.

In dem agilen Manifest haben sich die Autoren auf insgesamt zwölf Prinzipien geeinigt. Die höchste Priorität haben die regelmäßigen Lieferungen von hochwertiger Software an den Kunden, wobei die Lieferungszyklen so kurz wie möglich sein sollten. Dabei wird der ausgelieferte funktionierende Code als Fortschrittsmaßstab verwendet (Cockburn, 2003). Agil bedeutet außerdem, dass mögliche Änderungen an den Anforderungen positiv aufgenommen werden, um so durch die Einbindung von neuen Technologien und Geschäftsprozessen ein Wettbewerbsvorteil zu erlangen. Nur ein hochqualitatives und ständig überprüftes Design lässt sich problemlos an Änderungen anpassen. Nicht nur die Anforderungen können sich ändern, sondern auch die Arbeitsgewohnheit des Teams: mit Hilfe von regelmäßigen Reflexionen soll die Arbeitsweise Schritt für Schritt geprüft und verbessert werden, bis das Team am effektivsten arbeitet. Darüber hinaus wird eine enge Zusammenarbeit zwischen Entwicklern und Anwendern gefordert, um schnelles Feedback über die Funktionsweise der Software zu erhalten. Am effektivsten ist ein Projektteam, wenn es sich aus motivierten Mitarbeitern zusammensetzt, die sich selbst organisieren können und von außen gefördert werden. Das Unternehmen kann sich so auf gute Arbeit verlassen und muss nicht in das Geschehen durch festgelegte Methoden eingreifen. Im agilen Manifest wird auch gerade die Kommunikation als treibende Kraft für effektive Zusammenarbeit vorgestellt. Ein weiteres essenzielles Prinzip ist die Einfachheit – „die Kunst, die Menge nicht getaner Arbeit zu maximieren“ (Beck, et al., 2001). Das bedeutet, dass zum Beispiel durch weniger Arbeit auf Grund von einfachen und klaren Prinzipien trotzdem höhere Leistung erzielt werden kann.

Neben den vom agilen Manifest definierten Prinzipien empfiehlt es sich jedoch auch eigene zu erarbeiten, damit sich die Mitarbeiter besser damit identifizieren können (Lundak, 2009).

Zur Unterstützung und Umsetzung der Prinzipien wurden einige Praktiken und Techniken entworfen, sogenannte „Best Practices“. Zu den bekanntesten gehören Pair Programming, Reflexion, Refactoring, Test-Driven Development, kontinuierliche Code-Integration und kontinuierliche Tests.

## Crystal Clear

Die Crystal Familie

Jedes Projekt ist unterschiedlich und benötigt andere Methoden, um erfolgreich abgeschlossen zu werden. Aus diesem Grund hat Alistair Cockburn, einer der Urheber des agilen Manifests, eine Methodikfamilie entworfen namens Crystal. Sie enthält viele verschiedene Methodiken für unterschiedliche Projektarten, doch alle diese Methodiken haben einen „gemeinsamen genetischen Code“ (Cockburn, 2005). Mit Hilfe des Codes können Unternehmen ein neues Familienmitglied erzeugen, welches an die Bedürfnisse ihrer Projekte angepasst ist.

Die einzelnen Methodiken der Familie werden durch Teamgröße und Kritikalität charakterisiert, welche mit Hilfe von Farbe und Härtegrad angegeben werden. Desto dunkler die Farbe ist, umso größer ist das Projektteam. So wird Crystal Clear zum Beispiel für Teams mit ein bis sechs Mitgliedern ausgeführt.

Gemeinsam verfolgen die Crystal Methodiken alle dieselben Ziele: der positive Projektausgang soll sichergestellt werden, eine effiziente Entwicklung wird angestrebt und das Team soll sich mit den Konventionen wohlfühlen (Cockburn, 2005). Darüber hinaus legte Alistair Cockburn für die Crystal Familie fest, dass der Detailierungsgrad der Dokumentation von den Projektgegebenheiten abhängt und nicht für jedes Projekt gleich sein muss. Als Ausgleich legt Crystal aber sehr viel Wert auf kurze und ergiebige Kommunikationspfade und regelmäßige Abstimmungen der Arbeitsgewohnheiten, um die Zusammenarbeit flexibel verbessern zu können.

Gute und effiziente Kommunikation ist eines der wichtigsten Prinzipien, aus diesem Grund gibt es speziell für kleine Teams ein auf osmotische (enge) Kommunikation spezialisiertes Familienmitglied namens Crystal Clear. Alistair Cockburn stellt bei der Definition der Methodik klar, dass Crystal Clear „nicht vollständig festgeschrieben“ ist, da sich auch alle Projekte unterscheiden (Cockburn, 2005). Die Methodik soll während eines Projektes Schritt für Schritt an das Projekt und das Team angepasst werden. Deshalb möchte er das Team nicht durch festgeschriebene Techniken und Methoden einengen, sondern versucht eher Empfehlungen zu geben. Alistair Cockburn schreibt, dass Crystal Clear ein „einfacher und toleranter Regelsatz sein soll, der das Projekt in sicheres Fahrwasser bringt“ (Cockburn, 2005).

Eigenschaften

Bei den Befragungen der besten Teams haben sich sieben Eigenschaften herausgestellt, die ein Projekt zu einem erfolgreichen Abschluss bringen. Crystal Clear erfordert die ersten drei Eigenschaften. Ein Crystal Clear Team kann jedoch noch mehr der empfohlenen Eigenschaften einhalten, „um weiter in den sicheren Bereich zu gelangen“ (Cockburn, 2005).

Zu den drei essenziellen Eigenschaften zählen regelmäßige Lieferungen, verdichtete (osmotische) Kommunikation und reflektierte Verbesserungen. Vor allem die osmotische Kommunikation wird bei Crystal Clear in den Vordergrund gestellt, weil die kleinen Teams auf engem Raum arbeiten können und somit eine viel effektivere Kommunikation stattfindet. Durch den großen Grad an Nähe können viele Informationen auch im Hintergrund aufgenommen werden und auch das Feedback erhöht sich, da der Kommunikation keine Hindernisse im Weg stehen, wie zum Beispiel über den Flur laufen zum nächsten Büro, um dann festzustellen, dass die gefragte Person nicht am Arbeitsplatz sitzt. Ebenso wie die Kommunikation im Team, muss jedoch auch die Kommunikation zum Kunden oder Endanwender durch regelmäßige Lieferungen gehalten werden. Dadurch wird sichergestellt, dass beide Parteien Feedback erhalten und das gewünschte Produkt entwickelt wird. Um die Effizienz zu steigern, müssen auch regelmäßig die Arbeitsweisen reflektiert und angepasst werden. Da Crystal Clear keine bestimmten Strategien und Techniken vorschreibt, sondern viele optional zur Verfügung stellt, liegt es an dem Team, die für sie geeignete Arbeitsweise durch Experimentieren herauszufinden.

Prozesse

„Crystal Clear verwendet geschachtelte zyklische Prozesse“ (Cockburn, 2005) und jedes Projekt besteht aus den folgenden Zyklen:

Damit Crystal Clear eingehalten wird, müssen mindestens zwei Lieferungszyklen mit tatsächlichen Lieferungen an den Kunden ausgeführt werden.

Die Rollen

Ein Crystal Clear Projekt definiert vier obligatorische Rollen und weitere vier zusätzliche Rollen (Cockburn, 2005).

Zum einen gibt es den *Auftraggeber*, der für die finanziellen Mittel des Projektes zuständig ist. Er weist dem Team die Richtung, indem er die Arbeitseinheiten vor jeder Iteration priorisiert und dabei Änderungen in den Geschäftsprozessen einfließen lässt.

Zum anderen benötigt das Team einen *erfahrenen Anwender*, der sich mit den Vorgängen und dem eingesetzten System auskennt und für regelmäßige Rücksprachen zur Verfügung stehen sollte. Er ist derjenige, der die ausgelieferte Software ausprobiert und den Entwicklern Feedback gibt.

Darüber hinaus wird ein *Chefdesigner* definiert, der vom fähigsten Designer ausgeführt wird und technischer Leiter des Projektes ist. Er übernimmt Aufgaben wie Projektmanagement und Förderung der Teammitglieder und fungiert als Bindeglied zwischen Auftraggeber und Projektteam.

Bei Crystal Clear wird die Rolle des *Designers* und des *Programmierers* kombiniert, da die Programmierung immer ein Design voraussetzt und die beiden Rollen somit nicht getrennt ausgeführt werden können.

Zusätzliche Rollen wie *Koordinator, Fachexperte, Tester* oder *Autoren* können wahlweise hinzugefügt werden, um die anderen Rollen zu entlasten.

Die Arbeitsergebnisse

Auch bei den Arbeitsergebnissen gilt, dass weder alle erforderlich noch alle optional sind (Cockburn, 2005). Aus diesem Grund sind durch aus äquivalente Ersetzungen, Variationen und Anpassungen der Arbeitsergebnisse möglich. Gerade bei Crystal Clear kann die Anzahl und Notwendigkeit von Zwischenergebnissen stark reduziert werden, da diese durch interpersonelle Kommunikation, Anmerkungen auf den Whiteboards oder Lieferungen teilweise ersetzt werden. Das schließt die Dokumentation aber nicht vollkommen aus. Alle Projektrelevanten Arbeitsergebnisse müssen in irgendeiner Form dokumentiert werden. Hierbei ist der Formalismus eher nebensächlich, da auch Ergebnisse in Form von Whiteboard-Aufschrieben und Flipcharts angemessen sind. Jedes Dokument wird einer oder mehreren Rollen zugeordnet, damit allen bewusst ist, wer für welches Ergebnis verantwortlich ist.

## Scrum

Der Begriff Scrum hat seinen Ursprung beim Rugby, wo dieser Begriff für „Gedränge“ steht.

Beim Scrum als agiles Vorgehensmodell geht es eher um das selbst Organisieren sowie um Eigenverantwortung.

„Scrum ist ein Management-Rahmenwerk zur Entwicklung komplexer Produkte“, welches besonders häufig in der Softwareentwicklung eingesetzt wird. „Das wichtigste an ‚Scrum ist, das man offen bleibt gegenüber neuen Einsichten oder Ideen, um Prioritäten und auch das Produkt anzupassen.“

Gerade in der Softwareentwicklung stehen zum Anfang der Planung und Umsetzung noch viele Unklarheiten im Raum, welche eine Änderung des Produktes und der Planung auslösen. Zudem ergeben sich während der Umsetzung meist komplexere Funktionen oder Probleme, welche in der Planung nicht berücksichtigt wurden.

Bei den agilen Softwareentwicklungsprozessen muss im Gegensatz zu den klassischen Ansätzen kein Neustart des Projekts oder der Funktion erfolgen, sondern die Anforderungen können vor oder nach einem Sprint geändert werden.

Die Vision

Wie bei jedem Projekt steht auch bei Scrum die Vision am Anfang. In der Version müssen das angestrebte Ergebnis des Projekts, der Grund für die Durchführung und der Nutzen des Projekts geklärt werden.

Das Release Planning Meeting

Während des Release Planning Meetings wird eine Liste von Anforderungen/Funktionen erstellt, welche das Projekt enthalten soll. Anschließend wird jeder Anforderung eine Priorität zugeordnet. „Hauptfunktionen, welche die höchste Priorität besitzen und welche meist schon zu Anfang am klarsten formuliert sind, sollten auch am frühesten abgearbeitet werden“. Weiter muss im Release Planning Meeting die Definition von „done“ festgelegt werden, dabei wird geregelt, welche Anforderungen erfüllt sein müssen um den Eintrag im Produkt Backlog als „Erledigt“ zu markieren.

Der Produkt Backlog

Die Liste von Anforderungen und Funktionen mit deren Prioritäten, die im Release Planning Meeting festgelegt wurden, wird Produkt Backlog genannt.

Der Produkt Owner

Der Produkt Owner ist das Bindeglied zwischen dem Auftraggeber und der Softwarefirma, er ist unter anderem für die Priorisierung der Funktionen zuständig. „Zudem ist es die Aufgabe des Produkt Owners für Fragen jederzeit zur Verfügung zu stehen und die Interessen der Steakholder zu priorisieren und die dem Entwicklern näher zu bringen“.

Dazu muss der Produkt Owner Kenntnisse darüber besitzen, was:

„der Kunde benötigt und worin der höchste Wert liegt“

„der Markt und die Konkurrenz anbieten“

„die Entwickler an Informationen benötigen“

Das Sprint Planning Meeting

Während des Sprint Planning Meetings werden die Aufgaben, so genannte Tasks priorisiert, welche in einem Sprint abgearbeitet werden. Dabei muss abgeschätzt werden in welcher Zeit das Team die einzelnen Tasks abarbeiten kann. „Kurze Zyklen mit fester Zeitdauer führen innerhalb kürzester Zeit zur besseren Einschätzung, was in dieser Zeit möglich ist.“ Dadurch ist es einfacher dem Kunden einen ungefähren Zeitplan zu vermitteln.

Der Sprint Backlog

die einzelnen Tasks die innerhalb eines Sprints abgearbeitet werden sollen, werden in den Sprint Backlog eingetragen. Jede Aufgabe innerhalb eines Tasks sollte an einem Tag abzuarbeiten sein.

Während eines Sprints sind die Einträge im Sprint Backlog fixiert und es dürfen keine fundamentalen Änderungen an ihnen vorgenommen werden.

Der Sprint

Während eines Sprints, „welcher in der Regel variabel ist, jedoch in der Regel 2-4 Wochen dauert“ organisiert sich das Entwicklerteam selbst. Dabei weißt sich das Team selbst Aufgaben zu und klärt die jeweiligen Verantwortlichkeiten. „Am Ende eines Sprints steht eine funktionierende Funktion, welche der Kunde ausführlich Testet“.

Das Daily Scrum Meeting

An jedem Tag des Sprints steht ein Daily Scrum Meeting an, welches einen Zeitrahmen von 15 Minuten nicht überschreiten und jeden Tag zum selben Zeitpunkt stattfinden sollte. „Daily Scrum Meetings finden am besten im Stehen statt, das jeder beteiligte aktiv bleibt“. Während eines Daily Scrums sollte jeder beteiligte am Sprint folgende drei Fragen beantworten um dem Team den momentanen Entwicklungsstand zu erläutern:

„Was habe ich seit dem Letzten Meeting erreicht?“

„Was werde ich bis zum nächsten Meeting erreichen?“

„Was blockiert mich?“

Der Scrum Master

Der Scrum Master achtet auf die Einhaltung des ‚Daily Scrums sowie auf dessen Struktur und regelt den Ablauf.

Das Sprint Burndown Chart

Während eines Sprints werden alle Tasks mit geschätzter Zeit und fertige Tasks mit benötigter Zeit versehen. Diese Zeiten werden dann im Burndown Chart grafisch dargestellt, dies dient dazu, dass das Team den Vorschritt des Sprints verfolgen kann und auch sieht ob die Zeitplanung eingehalten werden kann.

Der Sprint Review

Am Ende eines Sprints steht der Sprint Review, daran nimmt das Team, der Scrum Maser und der Produkt Owner teil. In diesem werden die Ergebnisse des Sprints präsentiert. Dabei werden schnell Missverständnisse oder evtl. Fehlfunktionen aufgedeckt, welche dann im nächsten Sprint korrigiert werden müssen. „Aus ökonomischer Sicht beinhaltet jede Änderung eine Verbesserung und daher sind diese Änderungen gewünscht.“

Das Retrospektiv Meeting

Es erfolgt im Anschluss des Print Reviews, dabei wird der Sprint im Hinblick auf Probleme und Verbesserungen durchleuchtet. Das Meeting findet ohne den Produkt Owner statt, da es dabei um interne Strukturen geht, die nicht direkt mit dem Produkt zu tun haben müssen. Hier geht es um Verbesserungen auch im Hinblick auf weitere Sprints und Projekte.

## Kanban

# Vergleichskriterien für die Methoden

# Wahl der Tools

# Wahl des Projektes

# Vorbereitung der Durchführung

# Durchführung des Projektes

## Crystal Clear

## Scrum

## Kanban

# Ergebnisse

## Analyse/Ursachenforschung

### Crystal Clear

### Scrum

### Kanban

## Auswertung der Tools

## Gegenüberstellung der agilen Prozesse

# Fazit