|  |
| --- |
| 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 19

4. Vergleichskriterien für die Methoden 20

5. Wahl der Tools 21

6. Wahl des Projektes 22

7. Vorbereitung der Durchführung 27

8. Durchführung des Projektes 28

8.1 Crystal Clear 28

8.2 Scrum 28

8.3 Kanban 28

9. Ergebnisse 29

9.1 Analyse/Ursachenforschung 29

9.1.1 Crystal Clear 29

9.1.2 Scrum 29

9.1.3 Kanban 29

9.2 Auswertung der Tools 29

9.3 Gegenüberstellung der agilen Prozesse 29

10. Fazit 30

Abbildungverzeichnis

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

Tabellenverzeichnis

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

Abkürzungsverzeichnis

# Einleitung

In der Softwareentwicklung setzt man seit einigen Jahren immer mehr auf agile Entwicklungsprozesse. Diese versprechen durch vermehrten Kontakt mit dem Kunden und hohe Anpassungsfähigkeit an Produktänderungen ein schnelleres lieferbares Ergebnis. Der große Nutzen gegenüber linearen Prozessen wie Wasserfall und RUP lässt sich nicht mehr bestreiten. Agil ist jedoch ein sehr allgemeiner Begriff und so kann man nicht generell von dem agilen Prozess sprechen. Angefangen mit dem Extrembeispiel „Extreme Programming“ über die sehr häufig genutzte Variante Scrum, wurden durch diese agile Bewegung bis heute viele Prozesse entwickelt, die sich oft nur in wenigen Eigenschaften unterscheiden.

Gerade durch die Prozessvielfalt fällt es oft schwer eine Entscheidung zu treffen, welcher Prozess am besten zum Projekt passt. Aus diesem Grund ist es Ziel dieser Arbeit verschiedene agile Prozesse zu betrachten, auszuwerten und zu vergleichen. Es soll herausgefunden werden, in wie weit sich die einzelnen Prozesse tatsächlich unterscheiden und in welchen Fällen man einen bestimmten Prozess vorziehen sollte.

Hierzu wurden drei agile Prozesse ausgewählt:

* Scrum
* Crystal Clear
* Kanban

Als eine sehr verbreitete und beliebte Variante darf Scrum in diesem Vergleich nicht fehlen, vor allem weil es auch oft an den Hochschulen als „die“ agile Methode vorgestellt wird. Außerdem besteht unter den Teammitgliedern bereits Erfahrung in der Durchführung von Scrum, da es in einigen Unternehmen angewandt wird. Daneben wurde ein eher unbekannter von Alistair Cockburn entworfener Prozess namens Crystal Clear gewählt, da er sich speziell für kleine Teams, wie es bei dieser Arbeit der Fall ist, eignet und eine abgeschwächte Form von XP sein soll (Cockburn, 2005). Darüber hinaus hat man sich für den Entwicklungsprozess Kanban entschieden, der eigentlich aus der Produktion stammt und speziell für sein Kanban-Board bekannt ist. Daher wäre es interessant den eigentlichen Prozess hinter dem Namen kennen zu lernen.

Die Arbeit wurde nach einem bestimmten Schema gegliedert, welches das zeitliche Vorgehen während der Durchführung widerspiegelt. Bevor auf den tatsächlichen Vergleich eingegangen wird, werden im ersten Teil die agile Softwareentwicklung und die drei genannten Prozesse vorgestellt, um einen ersten Einblick in das Thema zu vermitteln. In den darauf folgenden Kapiteln werden einige Vorüberlegungen beschrieben. Dazu gehört einerseits die Festlegung der Vergleichskriterien, welche bereits eine erste Gegenüberstellung auf Basis der Recherchen liefern soll. Andererseits wird im Anschluss darauf ein Beispielprojekt definiert, auf welches die Prozesse angewandt werden sollen, um dadurch selbst erhobene Vergleichsdaten führen zu können. Ein weiterer Teil dieser Arbeit ist es die Verwendbarkeit von bestimmten Projektmanagementtools in Bezug auf die drei Prozesse zu testen. Dazu werden aus einem Pool von Tools nach einer ausführlichen Bewertung zwei ausgewählt und während des Beispielprojekts angewandt. Die Durchführung des Beispielprojekts mit drei Prozessen erweist sich jedoch als durchaus schwierig, aus diesem Grund wird hierzu eine Strategie entworfen und vorgestellt.

Das darauf folgende Kapitel beschreibt die Durchführung der Prozesse und soll speziell die Vorgehensweise und die selbst gewonnenen Erfahrungen dokumentieren. Es beinhaltet wie die Prozesse umgesetzt wurden und zeigt einige Ergebnisse aus Meetings, Reflexionen und Dokumentation. Hierbei soll der Fokus nicht auf dem zu entwickelnden Produkt sondern auf dem eigentlichen Vorgehen liegen. Danach folgt die eigentliche Analyse der Prozesse. Dabei wird für jeden Prozess auf die Frage eingegangen, was funktioniert oder nicht funktioniert hat und was wohl die Ursachen dafür waren. Außerdem werden die angewandten Tools ausgewertet und anschließend eine Gegenüberstellung der drei agilen Prozesse auf Basis der gewonnenen Erfahrungen aufgestellt. Hierbei werden die Punkte aus dem Vorabvergleich vom Anfang aufgegriffen und mit den selbst erhobenen Informationen verglichen.

Zum Schluss dieser Arbeit sollen auch mögliche Kombinationen der Entwicklungsprozesse abgewägt und Empfehlungen für bestimmte Projektgegebenheiten gegeben werden.

# 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.



Abbildung 1

### 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“[[1]](#footnote-1)

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".

### 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. Um agil zu sein müssen nicht alle diese Praktiken ausgeführt werden, jedoch helfen sie dabei die agilen Werte im Projekt zu leben.

Aus diesem Grund gibt es viele verschiedene agile Prozesse, die auf unterschiedliche Weise versuchen die Prinzipien in den Projekten durchzuführen. Es gibt also nicht nur einen agilen Prozess, sondern viele verwandte Modelle, die sich von Projekt zu Projekt unterscheiden können.

Im Folgenden soll explizit auf die ausgewählten Prozesse Crystal Clear, Scrum und Kanban eingegangen und die wichtigsten Merkmale dargestellt werden.

## 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 Vison müssen das angestrebte Ergebnis des Projekts, der Grund für die Durchführung, der Nutzen des Projekts, der Einsatzbereich, die Branche , das Budget und der Zeitplan geklärt werden. Die Projektvision ist nicht technisch geprägt. Die Vision wird vom Product Owner in Zusammenarbeit mit dem Kunden erstellt um dem Scrum Master und dem Team ein fest definiertes Ziel zur Verfügung zu stellen

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 Product Backlog

Die Liste von Anforderungen und Funktionen aus der Vision User Story, mit deren Prioritäten, die im Release Planning Meeting festgelegt wurden, wird Produkt Backlog genannt. User Stories sind vage Anforderungen aus Benutzersicht welche einen sichtbaren Mehrwert für die Kunden darstellt. Dabei sollten User Stories diese Eigenschaften aufweisen:

* Independent: unabhänig voneinander
* Negotiable: verhandelbar
* Valuable: Wert für den Kunden
* Estimatable: schätzbar
* Small: klein
* Testable: testbar

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

In Ersten Sprint Planning Meeting wird die Softwarearchitektur im Groben festgelegt um Abhängigkeiten bei der Planung zu berücksichtigen. Während des Sprint Planning Meetings werden die User Stories in so genannte Tasks unterteilt, 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. Der Sprint Backlog dient dem Team sowie dem Scrum Master zur Übersicht wer welche Aufgabe gerade bearbeitet oder was evtl. kritisch wird mit der Umsetzung in diesem Sprint

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 1-4 Wochen dauert“, organisiert sich das Entwicklerteam selbst. Dabei weißt sich das Team selbst Aufgaben zu und klärt die jeweiligen Verantwortlichkeiten. Während eines Sprints arbeitet das Team komplett eigenverantwortlich. „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. Zudem Überwacht er die Einhaltung des Sprint Ziels. Zusätzlich ist der Scrum Master dafür verantwortlich den Product Owner vom Entwicklungsteam während eines Sprints fernzuhalten.

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, sowie eventuell der Kunde, zukünftige Anwender und der Geschäftsführer, teil. In diesem werden die umgesetzten User Stories des Sprints präsentiert. Dabei werden schnell Missverständnisse oder evtl. Fehlfunktionen aufgedeckt, welche dann im nächsten Sprint korrigiert werden müssen. Dabei sollte die Präsentation eine Live Demo sein. „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 nur mit dem Team und dem Scrum Master statt, da es dabei um interne Strukturen geht, die nicht direkt mit dem Produkt zu tun haben müssen. Dabei werden Fragen wie: was war schlecht, oder was war gut gestellt. Hier geht es um Verbesserungen welche die Produktivität des Teams steigern, auch im Hinblick auf weitere Sprints und Projekte. Dabei wird eine Liste mit konkreten Verbesserungsmaßnahmen erstellt.

## Kanban

# Vergleichskriterien für die Methoden

# Wahl der Tools

# Wahl des Projektes

Um den Vergleich der drei ausgewählten agilen Softwareentwicklungsprozesse auf eigene praktische Erfahrungen stützen zu können, sollen sie an realen Projekten durchgeführt werden. Die Prozesse können so anhand der Durchführung analysiert und verglichen werden.

Nun stellt sich die Frage, wie man drei Prozesse so durchführt, dass ein optimales Vergleichsergebnis erarbeitet werden kann. Die optimalste Lösung wäre es für jeden Softwareentwicklungsprozess ein eigenes Projektteam zu definieren und jedem Team die gleiche Aufgabenstellung zu geben. Wenn man davon ausgeht, dass die Qualifikationen der Teams ausgeglichen sind, besteht somit für den Vergleich die ideale Basis: gleiches Projektziel, vergleichbares Team, identischer Zeitraum und generell gleiche Gegebenheiten.

Im Rahmen dieser Studienarbeit kann der Ansatz jedoch nicht realisiert werden, da das Projektteam nur aus drei Personen besteht, die gerade die Mindestanzahl für ein Projekt darstellt. Aus diesem Grund werden im Folgenden einige Lösungsansätze erörtert, die sich mit der Problemstellung befassen, wie drei Prozesse zeitgleich von einem einzigen dreiköpfigen Team durchgeführt werden können.

Lösungsansätze

Eine Möglichkeit wäre jeweils ein Projekt für jeden Prozess durchzuführen. Dadurch kann die Technologie unabhängig von den anderen Projekten definiert werden. Außerdem beeinflussen sich so die Projekte nicht, da sie unterschiedliche Aufgabenstellungen haben und die Teammitglieder ihre Erfahrungen aus den anderen Projekten nur bedingt benutzen können. Die Projekte sind aus diesem Grund aber auch schwerer vergleichbar, denn die Projektgegebenheiten unterscheiden sich, wodurch keine einheitliche Vergleichsbasis existiert. Darüber hinaus ist das Management von drei separaten Projekten sehr aufwendig.

Um die Nachteile des ersten Ansatzes zu verringern wurde eine zweite Lösung entworfen, bei der ein Projekt in drei Module unterteilt wird und jedes dieser Module mit einem anderen Prozess bearbeitet wird. Dadurch wird das Management etwas leichter aber problematisch wird hierbei die Projektfindung, da man durch die Trennbarkeit in drei Module sehr eingeschränkt ist. Selbst bei diesem Ansatz ist keine gute Vergleichbarkeit gegeben, da die Module sich auch trotz des gemeinsamen Ziels in den Aufgabengebieten unterscheiden.

Da sich die Wahl von drei separaten Projekten oder Modulen nicht als vergleichbar erweist, ist es eher ratsam sich auf die Wahl von einem Projekt zu beschränken. Ein denkbarer Ansatz wäre die drei Prozesse gleichzeitig auf ein Projekt anzuwenden. Aber bereits hier stellt sich die Frage, wie dies möglich sein soll. Ein Projektteam, das drei Prozesse gleichzeitig auf genau dasselbe Projekt durchführen sollen, wäre mit Sicherheit überfordert, da es schwierig ist die Prozesse zu trennen. Darüber hinaus würden sich die Prozesse gegenseitig beeinflussen oder sogar behindern. Ein Vergleich wäre somit unmöglich.

Aus diesem Grund hat man sich entschieden zwar nur ein Projekt zu definieren, dieses jedoch separat für jeden Prozess durchzuführen. Dadurch werden für die definierten Projektanforderungen drei getrennte Lösungen entwickelt, die sich sehr gut als weiteres Mittel für den Vergleich eignen. Ein Nachteil ist, dass die Projekte sich gegenseitig beeinflussen, da durch die Einteilung der gleichen Teammitglieder in jedem Projekt ein Wissenstransfer entsteht. Dieser Faktor kann aber weitgehend ignoriert werden, denn der dadurch simulierte Erfahrungsaustausch könnte in normalen Umständen auch durch Berater stattfinden. Mit dieser Variante ist somit die beste Vergleichsbasis geboten.

Vorgaben

Bei der Wahl des Projektes müssen einige Vorgaben beachtet werden. Einerseits sollte es machbar sein, den Projektscope in 6-8 Wochen abzuarbeiten, um so den zeitlichen Rahmen der Studienarbeit nicht zu sprengen. Andererseits sollte das Projekt aber auch genügend Umfang bieten, um die Prozesse ausführlich anwenden und vergleichen zu können. Darüber hinaus sollen Projektanforderungen so gewählt sein, dass sie die Teammitglieder zwar fordern, aber nicht zu viele neue Kenntnisse verlangen. Denn aus Zeitmangel kann keine lange Einarbeitungsphase eingeplant werden.

Technologie

Aus diesem Grund hat man sich für eine Webanwendung entschieden, welche mit Hilfe von JSF (Java Server Faces) implementiert werden soll. Vorteil dieser Technologie sind die bereits bestehenden Kenntnisse der Teammitglieder in der Programmierung mit Java und HTML, welche die Hauptbestandteile von JSF darstellen. Als Alternative lässt sich auch die Verwendung von PHP nicht ausschließen, jedoch wäre die Einarbeitungsphase bei dieser Programmiersprache wesentlich aufwendiger.

Um JSF optimal nutzen zu können, soll die Entwicklungsumgebung Eclipse eingerichtet werden, welche mit weiteren Frameworks wie Hibernate erweitert wird, um das Datenbank Mapping zu realisieren. Für die Datenbank wird die OpenSource Lösung MySQL verwendet. Darüber hinaus kann das in Eclipse integrierte Maven Tool genutzt werden, welches einen automatischen Build ermöglicht. Eine weitere wichtige agile Praktik ist das Testen, das mit dem Unit-Test Framework JSFUnit umgesetzt werden kann. Aufgrund der Inkompatibilität von JSFUnit mit anderen Servern, muss für das Deployment der Webapplikation ein JBoss Server eingerichtet werden. Da das Produkt des Projektes nicht direkt einem Kunden ausgeliefert werden muss, genügt eine lokale Implementierung der Applikation.

Projektanforderungen

Ziel des gewählten Projektes ist es, ein Spiel als Webapplikation umzusetzen. Bei dem Spiel handelt es sich um eine gitterartige Oberfläche auf der verschiedenfarbige Spielsteine verteilt sind. Der Spieler hat die Möglichkeit Spielsteine, die in einer Gruppierung von gleichartigen Steinen vorliegen, auszuwählen und zu löschen. Beim Löschen werden alle Spielsteine der Gruppierung gelöscht und der Spieler erhält proportional zur Anzahl der gelöschten Objekte Punkte. Danach werden die entstandenen freien Plätze durch Aufrücken der Objekte von oben nach unten wieder gefüllt.

An dieser Stelle wurden drei Varianten erarbeitet wie ein Level abgeschlossen werden kann, um so für einen geringen Unterschied zwischen den Projekten zu sorgen, welcher sich aber nicht auf den Vergleich auswirken sollte. Generell ist ein Spiel zu Ende, wenn keine gleichfarbigen Gruppierungen von Spielsteinen mehr vorliegen und der Spieler somit nicht in der Lage ist, einen weiteren Spielzug durchzuführen. Im weiteren Verlauf sprechen wir hierbei von dem generellen Abbruchkriterium.

|  |  |
| --- | --- |
| Variante 1: Zeit | Auf der einen Seite ist es möglich das Spiel nach einer vordefinierten Zeit, welche pro Level gesetzt wird, zu beenden. Der User muss somit in einer bestimmten Zeit genügend Punkte sammeln, um die Mindestpunktzahl zu erreichen und im Level aufzusteigen. Hierbei wird das Spielfeld nach jedem Zug wieder mit neuen Spielsteinen aufgefüllt, so dass der Spieler immer einen Spielzug durchführen kann. |
| **Variante 2: Punkte** | Andererseits kann man das Spiel auch so implementieren, dass die durch das Löschen entstehenden Lücken bestehen bleiben und höchstens bei kompletten leeren Zeilen die Blöcke aufgerückt werden. Dazu wird wieder eine Mindestpunktanzahl festgelegt, die der Spieler erreichen muss, um ein neues Level zu erreichen. Der Spieler hat danach jedoch die Möglichkeit weitere Spielzüge durchzuführen, um noch mehr Punkte sammeln zu können. Das Spiel ist also erst dann beendet, wenn das generelle Abbruchkriterium erreicht wird. |
| **Variante 3: Blöcke** | Eine weitere Variante wäre es eine Mindestblockanzahl zu definieren, welche zum erfolgreichen abschließen eines Levels führt. Der Spieler muss es also schaffen, so viele Gruppierungen wie möglich zu löschen, damit zum Schluss nur noch die definierte Anzahl an Blöcken vorhanden ist. Auch in diesem Fall ist das Spiel erst dann fertig, sobald das generelle Abbruchkriterium gilt, um dem Spieler die Möglichkeit zu geben noch mehr Punkte zu erreichen, da diese später in der Rangliste relevant sind. |

Ein Spieler kann ein Level so lange durchspielen, bis er das nächste Level erreicht hat. Außerdem soll eine Rangliste implementiert werden, welche die Spielergebnisse aller bisheriger Spieler vergleicht und dem Spieler so Feedback über seine Leistung gibt. Generell kann sich in einer Session immer nur ein Spieler anmelden. Es muss also nur ein Einzelspielermodus erarbeitet werden. Die Benutzer können sich mit ihrem Namen und dem Ergebnis in der Rangliste eintragen, aber eine Registrierung muss nicht stattfinden. Die Anwendung beginnt somit für jeden Spieler bei einer neuen Session beim ersten Level und er muss sich von neuem hocharbeiten.

# 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

1. Quelle: <http://www.heise.de/developer/meldung/10-Jahre-Agiles-Manifest-zur-Geburt-agiler-Softwareentwicklung-1188299.html> [↑](#footnote-ref-1)