

FOM Hochschule für Oekonomie & Management Essen
Standort Essen



Berufsbegleitender Studiengang
Wirtschaftsinformatik, 4. Semester

Hausarbeit im Rahmen der Lehrveranstaltung
Software Engineering

über das Thema

Analyse und Optimierung des Softwareentwicklungsprozesses eines mittelständischen Softwarehauses

Autor: Aleksandar Simic
Matrikelnr.: 396631
Stüvestraße 34
45144 Essen

Abgabe: 16. Juli 2017

Inhaltsverzeichnis

Abbildungsverzeichnis	IV
1 Einleitung	1
1.1 Themenvorstellung	1
1.2 Zielsetzung	1
1.3 Aufbau	1
2 Ausgangssituation	3
2.1 IST-Analyse	3
2.1.1 Unternehmensvorstellung	3
2.1.2 Projektablauf	3
2.1.3 Teambildung	5
2.1.4 Zuständigkeiten	5
2.1.5 Kommunikation	5
2.2 SOLL-Analyse	6
2.2.1 Anforderungen	6
2.2.2 Wünsche	6
3 Agile Softwareentwicklung	8
3.1 Grundlagen	8
3.1.1 Agilität	8
3.1.2 Agile Softwareentwicklung	8
3.2 Extreme Programming	10
3.2.1 Ablauf	10
3.2.2 Kunde vor Ort	10
3.2.3 User Stories	10
3.2.4 Planung und Entwicklung	11
3.3 Feature Driven Development	11
3.3.1 Ablauf	11
3.3.2 Allgemeines Modell	12
3.3.3 Funktionsliste erstellen	12
3.3.4 Planung, Design und Entwicklung	12
3.4 Scrum	13
3.4.1 Ablauf	13
3.4.2 Scrum-Rollen	13
3.4.3 Scrum-Fluss	14
3.4.4 Scrum-Artefakte	14

4	Optimierung des Softwareentwicklungsprozesses	16
4.1	Evaluation der Modelle	16
4.2	Auswahl einer Lösung	16
5	Schlussbetrachtung	17
5.1	Fazit	17
5.2	Ausblick	17
	Literaturverzeichnis	V

Abbildungsverzeichnis

1	Flussdiagramm eines Projektablaufs	4
2	Ablauf eines Scrum-Sprints	14

1 Einleitung

1.1 Themenvorstellung

Immer mehr Unternehmen steigen in ihrer internen Abwicklung von Projekten auf sogenannte agile Prozessmodelle um. Das in dieser Seminararbeit behandelte Unternehmen steht nun vor der Frage, ob dessen Betriebsprozesse auf eines dieser Modelle umgestellt werden sollen. Aus einem Interview mit dem Bereichsleiter für Softwareentwicklung B. R. geht hervor, dass bereits interne Gespräche stattfanden und einige Anforderungen getroffen wurden. Da die Prozesse des Unternehmens bereits veraltet sind und eindeutig Verbesserungspotential besteht, wird die Prüfung, ob eines der agilen Prozessmodelle die Anforderungen der Geschäftsleitung erfüllen kann, zur Motivation für diese Arbeit genutzt. TODO Zitat

1.2 Zielsetzung

Mit dieser Seminararbeit wird das Ziel verfolgt, die Prozesse des gewählten Unternehmens zu optimieren bzw. an die gestellten Anforderungen anzupassen. Die veralteten und suboptimalen Vorgehensweisen in der Projektabwicklung sollen aktuellen, effizienteren Prozessen weichen, um die Vorteile der Agilität im Management von Projekten maximal ausnutzen zu können. Vorstellbar wären hier unter anderem Verbesserungen im Bereich der Flexibilität, Ressourcen- und Zeitausnutzung bzw. Wirtschaftlichkeit und Kostensenkung, um nur einige zu nennen. Dem Unternehmen soll letztendlich durch Evaluation ausgewählter agiler Prozessmodelle das für ihre Bedürfnisse und Vorstellungen auf Basis der definierten Anforderungen bestmögliche Modell geliefert werden.

1.3 Aufbau

Zu Beginn wird die aktuelle Prozessstruktur aufgezeigt. Es werden alle Schritte, die von der Auftragsakquisition bis hin zur Inbetriebnahme des fertigen Softwareproduktes durchlaufen werden, analysiert. Anschließend werden anhand eines Interviews mit dem zuständigen Bereichsleiter des Unternehmens die gewünschten Anforderungen spezifiziert und priorisiert. Außerdem werden einige grundlegende Punkte zum Thema Agilität und agiler Softwareentwicklung erklärt und drei weit verbreitet agile Prozessmodelle vorgestellt, und zwar XP (Extreme Programming), FDD (Feature Driven Development) und Scrum. Diese werden zunächst grob erläutert, deren wichtigste Bestandteile und Merkmale dargestellt und daraufhin der Ablauf, welcher im Zuge eines Projekts je Prozessmodell verfolgt

wird, aufgezeigt. Um ein für das Unternehmen geeignetes Modell bestimmen zu können, werden die zuvor vorgestellten im Rahmen der definierten Anforderungen evaluiert. Sollte keines der genannten Modelle die Anforderungen hinreichend erfüllen, wird versucht, aus den gegebenen ein passendes hybrides Prozessmodell zusammenzuführen. Abschließend werden die gewonnen Erkenntnisse noch einmal reflektiert und es wird ein Ausblick darauf gewährt, inwieweit sich das gewählte Modell in Zukunft optimieren lässt und welche weiteren Anforderungen gestellt werden könnten.

2 Ausgangssituation

2.1 IST-Analyse

2.1.1 Unternehmensvorstellung

Das Unternehmen ist in mehrere Vertriebsstandorte in ganz Deutschland verteilt, die Verwaltung und die Entwicklung des Kernprodukts ist allerdings im zentralen Standort gebündelt. Von den ca. 160 Mitarbeitern sind zurzeit 35 Personen für die Entwicklung zuständig, wobei 3 davon in der Ausbildung sind. Das Kernprodukt des Unternehmens besteht aus mehreren einzelnen Komponenten, die von den Entwicklern separat implementiert werden, beispielsweise aus einer GUI (Graphical User Interface) und einer zugehörigen Hintergrundverarbeitung. Die Software wird sowohl in einem fertigen Umfang als Standardversion als auch auf Nachfrage Individualanpassung entwickelt und vertrieben und nahezu jährlich auf eine neue Version geupdatet. Ebenso ist sie modulweise aufgebaut, sodass der Interessent einzelne Module hinzubestellen kann.

2.1.2 Projektablauf

Ein Projekt beginnt stets bei der Akquisition von Kunden. Da das Hauptaugenmerk des Unternehmens bei der Akquise auf Deutschland, Österreich und der Schweiz liegt, werden den einzelnen Standorten verschiedene Bereiche zugewiesen. Die Mitarbeiter im Vertrieb versuchen nun, das Softwareprodukt an den Kunden zu bringen, indem diesen die Standardversion präsentiert wird. Besteht kundenseitig Interesse am Erwerb des Produkts, werden Anpassungswünsche entgegengenommen und an die Entwicklung zwecks Prüfung der Realisierbarkeit weitergeleitet und gegebenenfalls nochmal überarbeitet.

Sind die Anpassungen schließlich umsetzbar oder bestehen keine, wird ein Auftrag generiert und zur Planung freigegeben. Dabei werden zunächst Aufwände für die einzelnen Anpassungen geschätzt. Entsprechend der Schätzungen wird ein Fertigstellungstermin für Entwicklung und Qualitätssicherung definiert und ein Auslieferungstermin an den Kunden übermittelt. Die von dem Auftrag betroffenen Komponenten und Module werden einzelnen Spezialisten zugewiesen, welche die Anpassungen implementieren und in die Qualitätssicherung übergeben. Sollten währenddessen Fehler auftreten, werden diese an den zuständigen Entwickler zusammengefasst in einem Fehlerprotokoll zurück gemeldet und bearbeitet. Nachdem alle Module angepasst und getestet wurden, werden die Auszubildenden über die Fertigstellung informiert. Die einzelnen Module werden durch die Auszubildenden zu einem fertigen Gesamtprodukt verarbeitet, an den Kunden ausgeliefert und dort auf dessen System installiert.

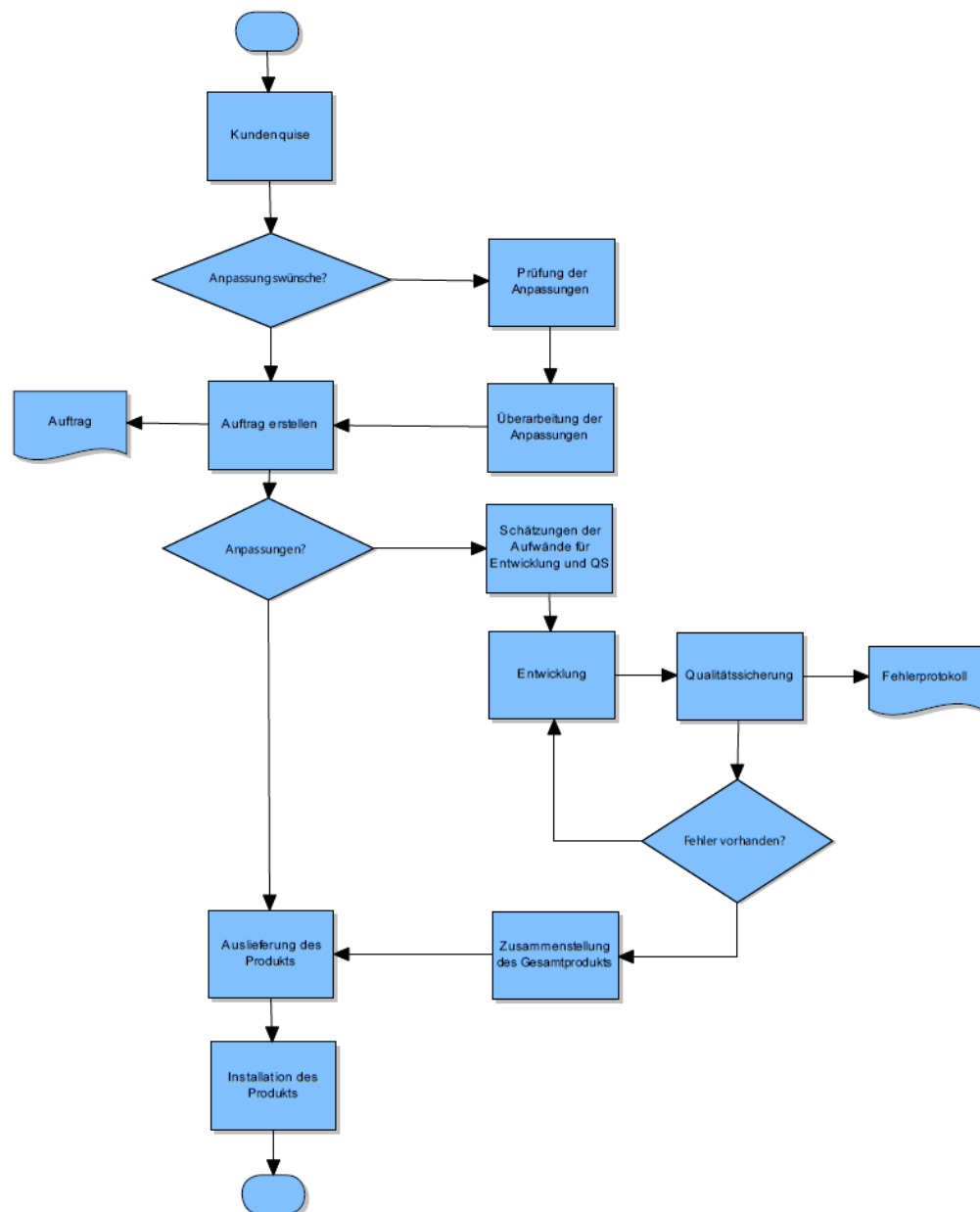


Abbildung 1: Flussdiagramm eines Projektablaufs
Quelle: Eigene Darstellung

Diese Projektart bezieht sich auf die Akquisition von Neukunden. Auch Bestandskunden können neue Projekte in Auftrag geben, wobei hier wieder zwischen zwei verschiedenen Projektgegenständen unterschieden wird. Zum einen können zu den bereits erworbenen Modulen weitere hinzugekauft werden. Dabei besteht erneut die Möglichkeit, dass der Kunde verschiedenen Anpassungen wünscht, die es zu prüfen und zu realisieren gilt. Außerdem können selbst im laufenden Betrieb Anpassungswünsche auftreten, welche durch einen sogenannten CR (Change Request) beauftragt werden. Dieser wird durch den zuständigen Kundenberater des Unternehmens in Zusammenarbeit mit einem kundenseitigen Ansprechpartner erstellt. Der Projektablauf gleicht anschließend dem beim Ersterwerb.

2.1.3 Teambildung

Für ein Projekt werden keine expliziten Teams definiert. Vielmehr existieren pro Modul ein oder zwei Spezialisten, die besonderes Wissen dazu besitzen oder sogar maßgeblich an der initialen Entwicklung des entsprechenden Moduls beteiligt waren. Die Module bestehen dabei aus einzelnen Programmen, welche wiederum einen unterschiedlichen Komplexitätsgrad aufweisen können. Daher kann es vorkommen, dass bestimmte Programme aufgrund ihrer Komplexität nur vom erfahrensten Entwickler bearbeitet werden können. Da die Entwickler aufgrund ihrer geringen Anzahl neben den Projektaufträgen auch das Tagesgeschäft erledigen müssen, werden sie nicht für die einzelnen Aufträge abgestellt. Sie müssen zeitweise bei Bedarf mehrere Aufträge gleichzeitig bearbeiten, gleiches gilt auch für die Mitarbeiter der Qualitätssicherung.

2.1.4 Zuständigkeiten

Welche Aufgaben von welchem Entwickler bzw. Tester übernommen werden, entscheidet zentral der Bereichsleiter. Er prüft die zu implementierende Anpassung, weist diese seinem Ermessen nach dem zuständigen Spezialisten zur Prüfung und schließlich zur Bearbeitung zu und bestimmt einen Tester. Dieser muss ebenfalls genügend Erfahrung mit dem Modul bzw. dem Programm besitzen, um sinnvolle Testfälle aufstellen zu können. Tritt kurzfristig ein dringliches Anliegen für ein Programm auf, dessen Spezialist bereits in einem Projekt tätig ist, muss dieser seine Arbeit unterbrechen und sich um das Anliegen kümmern. Der Tester ist zuständig für die Qualität des Programms. Er erstellt nach seinem Test ein Testprotokoll, in welchem die Testfälle und Fehler festgehalten werden, und stellt dieses für den Entwickler zur Korrektur bereit. Außerdem erstellt er die Dokumentation der getätigten Anpassung.

2.1.5 Kommunikation

Die einzelnen Entwickler, welche die verschiedenen Anpassungen eines Projekts realisieren, stimmen sich kaum untereinander ab. Treten Probleme oder Fragen auf, wird sich diesbezüglich kurz ausgetauscht. Nach Abschluss der Anpassung wendet sich unter Umständen der Tester an den Entwickler zur genaueren Erläuterung und Klärung von Verständnisfragen. Kommunikation zum Kunden besteht zunächst lediglich zu Beginn des Projekts. Hierbei stimmt der Projektleiter seitens des Unternehmens die gewünschten Anpassungen mit dem kundenseitigen Projektleiter ab. Zusätzlich kann ein Entwickler bei Fragen den Kunden selbstständig kontaktieren, es besteht somit keine geregelter Kommunikationsweg zum Kunden.

2.2 SOLL-Analyse

2.2.1 Anforderungen

Das neu einzuführende Prozessmodell soll bestimmte spezifizierte Anforderungen möglichst vollständig erfüllen können. Das Hauptaugenmerk liegt hierbei auf der optimierten Ressourcen- und Zeitausnutzung, da den Mitarbeitern aufgrund der Anzahl und Größe der Projekte mehrere Aufgaben parallel zugewiesen werden. Dies führt zu einer erhöhten Auslastung der Entwickler und Qualitätssicherer. Infolgedessen können Termingrenzen nicht immer eingehalten werden. Durch eine gezieltere Aufgabenteilung und Teambildung soll hier eine Optimierung erreicht werden, welche sowohl die Effizienz als auch die Entwicklungsdauer umfasst.

Das neue Prozessmodell soll die Bildung von abgegrenzten Projektteams vorsehen und gleichzeitig eine geringe Anzahl an Mitarbeitern pro Team favorisieren. Der Grund hierfür ist die geringe Anzahl an verfügbaren Entwicklern in Verbindung mit der parallelen Bearbeitung mehrerer Projekte. Zusätzlich sollen neue und unerfahrene Mitarbeiter an die Programme herangeführt werden und möglichst umfangreiche Kenntnisse in unterschiedlichen Fachgebieten erlangen, um das Wissen im Unternehmen maximal zu streuen. Dadurch soll der Schaffung von Spezialisten für die einzelnen Programme und Module entgegengewirkt werden.

Um zeitliche Engpässe während der gesamten Projektarbeit zu umgehen, soll im Allgemeinen durch Prozessoptimierungen und -vereinheitlichungen in den einzelnen Bereichen eine Verkürzung der gesamten Durchlaufzeit erreicht werden. Ebenfalls soll ein kunden-seitiger Ansprechpartner enger in die Entwicklung eingebunden werden. Aufgrund notwendiger Abstimmungen während der Realisierungsphase soll sowohl ein einheitlicher und direkter Kommunikationsweg zum Kunden geschaffen, als auch die Frequenz derselben erhöht werden. Dies soll bezüglich der Spezifikationen auftretende Probleme simpler und schneller lösbar machen.

2.2.2 Wünsche

Zum einen besteht der Wunsch, dass das gesamte Wissen bezüglich Entwicklung und Qualitätssicherung nicht auf einzelne Spezialisten verteilt ist, sondern jeder Entwickler und Tester jede Art von Projekt übernehmen können. Besonders Entwickler müssen in der Lage sein, Anpassungen für jedes Programm prüfen, schätzen und implementieren zu können, unabhängig vom Komplexitätsgrad. Zusätzlich wäre die Realisierung eines integrierten Risikomanagements wünschenswert. Es sollen Risiken analysiert, geprüft und

verhindert bzw. minimiert werden können, da diesbezüglich zurzeit keinerlei Aktivitäten im Unternehmen stattfinden. Dadurch sollen plötzlich auftretende Notsituationen wie Krankheitsfälle entschärft werden können.

3 Agile Softwareentwicklung

3.1 Grundlagen

3.1.1 Agilität

Definieren lässt sich Agilität unter anderem als eine Fähigkeit, sowohl Veränderungen zu schaffen als auch auf diese zu reagieren, um in einem turbulenten Geschäftsumfeld profitieren zu können. Agile Unternehmen zielen auf Veränderungen ab anstatt sie zu fürchten, da sie in der Lage sind, besser darauf zu reagieren als der Wettbewerb. Zugleich kreieren sie selbst Veränderungen, auf die Wettbewerber nur schwer antworten können und schaffen sich somit einen Vorteil. Unternehmen müssen allerdings den gewünschten Grad der Agilität bestimmen, da der dadurch erwirtschaftete Vorteil nur relativ zu den Wettbewerbern gesehen werden kann.¹

Gewandtheit und Flexibilität in Bezug auf schnelle Richtungsänderungen und die Fähigkeit, Marktänderungen vorauszusehen, sind ebenfalls relevant für ein agiles Unternehmen. Es muss auch in der Lage sein, ein Gleichgewicht zwischen Struktur und Flexibilität beizubehalten. Reine Flexibilität führt auf Dauer zu Fortschrittsproblemen, da sich laufend etwas ändert. Die Balance zwischen beiden Aspekten zu halten kann Unternehmen zum Erfolg führen.² Agile Prozesse müssen zudem sowohl leichtgewichtig als auch hinreichend sein. Die Leichtgewichtigkeit des Prozesses fördert die Flexibilität, die Hinlänglichkeit hält das Unternehmen im Spiel.³

3.1.2 Agile Softwareentwicklung

In der agilen Softwareentwicklung wird der Fokus auf andere Werte gesetzt als in der klassischen Projektdurchführung. So werden Mitarbeiter und deren Interaktion untereinander wichtiger für die Entwicklung guter Software eingeschätzt als die zu verwendenden Prozesse und Werkzeuge, die Funktionstüchtigkeit der Software wird der Dokumentation derselben übergestellt. Ebenso soll die Zusammenarbeit mit dem Kunden mehr umfassen als reine Vertragsverhandlungen, und Flexibilität in Bezug auf die Reaktion auf Veränderungen wird dem Befolgen eines festen Plans vorgezogen. Das heißt nicht, dass die zweitgenannten Aspekte unwichtig sind, sie werden lediglich gegenüber den anderen als unterlegener angesehen.⁴

¹Vgl. Highsmith (2002), Seite XXIII.

²Vgl. ebd.

³Vgl. Cockburn (2002), Seite 178.

⁴Vgl. Beck et al. (2001a).

Zu den genannten Werten werden auch einige Prinzipien verfolgt. Zum einen besteht das Hauptziel darin, die Zufriedenheit des Kunden durch die Auslieferung von qualitativ hochwertiger Software gewährleisten zu können. Zum anderen sollten funktionsfähige Versionen in festgelegten Zeitabständen an den Kunden geliefert werden, wobei diese Abstände vorzugsweise wenige Wochen oder Monate umfassen sollten. Die Funktionsfähigkeit der Software spielt bei der Beurteilung des Fortschritts die wichtigste Rolle. Um die Agilität zu fördern, sollte stets ein Augenmerk auf die Qualität des technischen Aspekts und des Designs gerichtet werden. Auch plötzliche Anforderungsänderungen in jeder beliebigen Phase der Entwicklung sollten akzeptiert werden, da durch die Agilität in genau diesen Situationen der Wettbewerbsvorteil geschaffen wird. Die agilen Prozesse ermöglichen zusätzlich eine nachhaltige Entwicklung, indem sie Auftraggeber, Benutzer und Entwickler auf einen unbegrenzten Zeitraum ein gleichmäßiges Tempo halten lassen.⁵

Bezüglich der Teams gilt es zu beachten, dass durch Selbstorganisation derselben die optimalen Anforderungen, Architekturen und Entwürfe erzeugt werden können. Dies setzt voraus, dass sich die Mitglieder des Teams regelmäßig zusammenfinden und reflektierend deren Effektivität beurteilen und dementsprechend ihr Verhalten anpassen. Die Kommunikation untereinander und zu anderen Teilnehmern des Projekts erfolgt am effizientesten in direkten Gesprächen. Entwickler und Fachexperten müssen dabei für die Dauer des Projektes täglich in Kontakt sein. Um die Projekte erfolgreich gestalten zu können, sind motivierte Mitarbeiter Pflicht. Ihnen müssen die größtmögliche Unterstützung und das Vertrauen geboten werden, sowie ein gutes Arbeitsumfeld. Außerdem ist es von großer Bedeutung, Einfachheit anzustreben.⁶

Das Ziel der agilen Softwareentwicklung ist es zudem, durch Anpassung bestimmter Faktoren sogenannte Sweet Spots zu identifizieren und diesen näher zu kommen. Sweet Spots bezeichnen in diesem Fall die Ausnutzung besonders effizienter Mechanismen der Softwareentwicklung.⁷

Einige solcher Sweet Spots sind die Erzeugung von monatlichen Inkrementen und die Bevorzugung von kleinen Teams zur Bewahrung der Agilität und Reaktionsgeschwindigkeit. Dabei sollten die Teams aus ca. zwei bis 8 Leuten bestehen und möglichst in einem gemeinsamen Raum arbeiten, wodurch sich gleichzeitig Kommunikationsweg und Aufwand verringern. Aber auch Ansprechpartner vor Ort, der Einsatz von erfahrenen Entwicklern und vollautomatisierte Regressionstest können als Sweet Spots angestrebt werden. Vor allem die letzteren steigern die Qualität der Software erheblich.⁸

⁵Vgl. Beck et al. (2001b).

⁶Vgl. ebd.

⁷Vgl. Cockburn (2002), Seite 178.

⁸Vgl. ebd., Seite 178 ff.

3.2 Extreme Programming

3.2.1 Ablauf

In XP (Extreme Programming) werden kleine Teams aus drei bis zehn Entwicklern gebildet. Zusätzlich wird ein Ansprechpartner beim Kunden vor Ort bestimmt, welcher laufend Fachwissen bereitstellen kann. Die Entwickler arbeiten im gleichen oder in benachbarten Räumen, vorzugsweise mit untereinander verbundenen Arbeitsplätzen. Es gibt lediglich halb so viele Arbeitsplätze wie Entwickler und diese sind so eingerichtet, dass alle Entwickler mit dem Rücken zueinander sitzen. Die Implementierung läuft in einzelnen Iterationen ab, welche immer einen qualitätsgesicherten Code erzeugt, der letztendlich an die Endbenutzer ausgeliefert wird.⁹

3.2.2 Kunde vor Ort

Die echten Kunden müssen in den Entwicklungsprozess mit einbezogen werden, indem sie stets für Fragen verfügbar sind, die Anforderungen liefern und Prioritäten setzen. Hierdurch wird die Zufriedenheit des Kunden durch Einbeziehung gewährleistet und Missverständnissen bezüglich der Vorgaben wird vorgebeugt.¹⁰ Dies wird dadurch erreicht, dass mindestens ein Vertreter des Kunden ins Unternehmen geholt und zu den Entwicklern gesetzt wird. Dort kann dieser auch im Zweifel weiterhin seiner Arbeit nachgehen, solange er für Fragen ansprechbar ist. Kann kein kundenseitiger Ansprechpartner ständig vor Ort sein, so sollte er zumindest für Meetings hinzugeholt oder die Entwickler für Fragen zu ihm geschickt werden.¹¹

3.2.3 User Stories

Eine User Story ist eine anwenderseitige Beschreibung, wie sich das Zielsystem verhalten soll. Sie stellen also die Anforderungen an das System dar. Die User Stories werden während der Analyse genutzt, welche allerdings nicht nur zu Beginn des Projekts durchgeführt wird, sondern laufend durch Kommunikation mit dem Kunden. Das Produkt regelmäßig an den Kunden ausgeliefert, um so möglichst schnell und frequent Feedback und u. U. neue Stories zu erhalten und diese in den Entwicklungsprozess mit einfließen zu lassen. Durch die Kommunikation über kleine User Stories kann die Analyse stetig und informell durchgeführt werden.¹²

⁹Vgl. Cockburn (2002), Seite 165.

¹⁰Vgl. Novoseltseva (2017).

¹¹Vgl. Jeffries et al. (2001), Seite 18 ff.

¹²Vgl. ebd., Seite 24.

3.2.4 Planung und Entwicklung

Zu Beginn der Entwicklungsphase wird ein Planungsmeeting veranlasst. In diesem stellt der Kunde seine erstellten User Stories den Entwicklern vor und stellt die Verständlichkeit derselben sicher. Anschließend führen die Entwickler ein Brainstorming durch, um für die einzelnen User Stories die für deren Implementierung notwendigen Aufgaben festzulegen und ein Systemdesign zu erstellen. Hierbei lassen sich noch Missverständnisse durch den Kunden aufdecken und klären. Zum Abschluss des Meetings melden sich Entwickler für eine Story an und schätzen den entstehenden Aufwand. Dieser Prozess wird wiederholt, bis alle Stories zugewiesen und geschätzt sind.¹³

Die Implementierung wird im Anschluss dergestalt vorgenommen, dass stets zwei Programmierer am selben Arbeitsplatz an derselben Aufgabe arbeiten, wobei einer die Arbeit tätigt und der anderen zuschaut und ihn unterstützt. Die Rollen des Programmierers und des Zuschauers werden regelmäßig getauscht. Dieses Vorgehen wird als Pair Programming bezeichnet.¹⁴ Durch Continuous Integration wird sichergestellt, dass der entwickelte Programmcode mehrmals am Tag auf einem zentralen System integriert und laufend getestet wird.¹⁵ Das zentrale Integrationssystem ist stets in einem konsistenten Zustand und muss folgende Kriterien erfüllen: Compilierfähigkeit, semantische Korrektheit und funktionierende Paketierung.¹⁶ Da jedes Paar während der Implementierung seiner Aufgabe auf bereits bestehende Klassen zugreift und somit jeder das Recht besitzt, jeden Teil des Codes anzupassen, muss für das gesamte Entwicklungsteam ein einheitlicher Programmierstandard beschlossen werden. Zusätzlich wird versucht, jede Lösung so weit es geht zu vereinfachen. Dieser Vereinfachungsprozess wird durch das Refactoring abgeschlossen. Danach folgt nur noch die Auslieferung der Software.¹⁷

3.3 Feature Driven Development

3.3.1 Ablauf

FDD wurde in den 90er Jahren von Jeff Luca kreiert und stellt, wie alle agilen Methodologien, einen iterativen und inkrementellen Entwicklungsprozess zur Erstellung lauffähiger Software dar. Der Fokus wird hierbei auf die Features und Funktionalitäten gesetzt, die der Kunde schätzt und erwartet.¹⁸ Aufgeteilt wird es in 5 Prozesse. Zunächst wird ein

¹³Vgl. Jeffries et al. (2001), Seite 63 f.

¹⁴Vgl. ebd., Seite 72.

¹⁵Vgl. ebd., Seite 78.

¹⁶Vgl. Pichler und Roock (2011), Seite 22.

¹⁷Vgl. Jeffries et al. (2001), Seite 72 ff.

¹⁸Vgl. Karam (2017).

allgemeines Modell aufgestellt, welches lediglich das Skelett des Produktes umfasst. Anschließend werden die Features gesammelt und in einer Feature Liste festgehalten. Diese Features werden priorisiert, Programmierern zugewiesen und von diesen in Arbeitspakete eingeteilt. Im letzten Schritt werden die Pakete umgesetzt, getestet und ausgeliefert.¹⁹

3.3.2 Allgemeines Modell

Wie bereits erwähnt, wird das Skelett des Produkts in einem allgemeinen Schema modelliert, welches noch keine Details enthält. Dieses wird von einem sogenannten Chief Architect aufgestellt. In größeren Projekten werden kleinere Modelle für bestimmte Bereiche des Produkts von Domänenteams gebildet, welche aus Modellierern und Domänenexperten bestehen. Diese werden anschließend in einem Integrationsmeeting in ein allumfassendes Modell integriert. Die einzelnen Domänenexperten liefern die zur Erstellung der Modelle nötigen Voraussetzungen, welche in der nächsten Phase in Features aufbereitet werden. Die Modellierung selbst besteht im Groben aus sieben Aufgaben: dem Aufbau eines Modellierungsteams, einer Domänenvorstellung, der Prüfung von Dokumenten, der Aufstellung des Modells bzw. der Modelle, der Aufbereitung des allgemeinen Modells, der Verfassung von Notizen über das Modell durch Chief Architect und Chief Programmers und einer Bewertung.²⁰

3.3.3 Funktionsliste erstellen

Aus dem erzeugten Modell wird durch die im vorherigen Prozess beteiligten Chief Programmers eine Funktionsliste erstellt. Dies geschieht durch die Aufteilung auf mehrere Ebenen, vom Fachbereich, über die Geschäftsaktivität bis hin zum Geschäftsaktivitätsschritt. Der Geschäftsaktivitätsschritt ist in diesem Fall das Feature, um das Verständnis für den Kunden zu vereinfachen wird hier allerdings die Business Terminologie verwendet. Die Geschäftsaktivität besteht aus ca. 10-20 Features und ist die Ebene, auf der das Feedback zum Kunden geliefert wird. Ein Feature muss so kleinschrittig definiert sein, dass es in maximal zehn Tagen fertig gestellt werden kann. Ansonsten muss es weiter gesplittet werden.²¹

3.3.4 Planung, Design und Entwicklung

Projektleiter, Entwicklungsleiter und die Chief Programmers setzen sich in einem Planungsteam zusammen und definieren die Reihenfolge, in welcher die Features umgesetzt

¹⁹Vgl. Highsmith (2002), Seite 274-278.

²⁰Vgl. ebd., Seite 274 f.

²¹Vgl. ebd., Seite 275 f.

werden sollen. Termine zur Fertigstellung werden dagegen nicht pro Feature, sondern pro Geschäftsaktivität gesetzt. Die Reihenfolge ist abhängig von verschiedenen Faktoren wie Komplexität, Risiken und Checkpoints. Zum Abschluss der Phase werden die Verantwortungen der Geschäftsaktivitäten den Chief Programmers und einzelne Klassen bestimmten Entwicklern zugewiesen.²²

In der nächsten Phase bündeln die Chief Programmers die erhaltenen Features in Arbeitspakete. Im Gegensatz zu den Geschäftsaktivitäten, die aus Features innerhalb eines Geschäftsframeworks bestehen, sind Arbeitspakete aus Features innerhalb eines technischen Konstruktionsframeworks zusammengesetzt. Zusätzlich werden Klassen und Methoden definiert. Diese werden in der letzten Phase implementiert, der Quellcode geprüft und überarbeitet, getestet und abschließend gebildet.²³

3.4 Scrum

3.4.1 Ablauf

Scrum besteht, entsprechend den übrigen agilen Prozessmodellen, aus iterativen und inkrementellen Prozessen. Zu Beginn einer Iteration von Entwicklungsaktivitäten prüft das Entwicklungsteam die ausstehenden Aufgaben und wählt die in dieser Iteration umsetzbaren Funktionalitäten aus. Das Team arbeitet autark, indem es die Anforderungen selbstständig bewertet, schätzt, Lösungsansätze selbstständig umsetzt und auf Hindernisse eigenständig reagiert, um die bestmögliche Leistung zu erbringen. Das Inkrement, welches als Ergebnis entsteht, wird vor den Stakeholdern anhand der erstellten Funktionalitäten präsentiert, um notwendige Anpassungen frühzeitig bestimmen zu können.²⁴

3.4.2 Scrum-Rollen

Dieser Rahmen wird in Scrum durch drei Rollen umgesetzt: dem Product Owner, dem ScrumMaster und dem Team. Der Product Owner trägt die Verantwortung für die Vertretung der Interessen aller Projektbeteiligten, die laufende Budgetierung des Projekts und die Erstellung von Releaseplänen. Er ist ebenfalls zuständig für die Pflege der Anforderungsliste und die damit zusammenhängende Priorisierung der einzelnen Anforderungen. Die Teams arbeiten selbstorganisiert und eigenständig und sind als Ganzes für den Erfolg einer Iteration und des Projekts verantwortlich. Zusätzlich prüfen sie gemeinsam, wie die Anforderungsliste in der nächsten Iteration als Inkrement umgesetzt werden kann. Der

²²Vgl. Highsmith (2002), Seite 276 f.

²³Vgl. ebd., Seite 277 f.

²⁴Vgl. Schwaber (2012), Seite 6.

ScrumMaster sorgt für die regelkonforme Ausführung von Scrum. Er trägt die Verantwortung für die Vermittlung und Schulung von Scrum-Inhalten und für die Integration in die Unternehmenskultur, um den erwarteten Nutzen weiterhin zu erhalten.²⁵

3.4.3 Scrum-Fluss

Zu Beginn eines Projekts steht die Vision des Kunden vom zu entwickelnden System. Diesbezüglich stellt der Product Owner ein Product Backlog auf, welches priorisierte und geschätzte funktionale und nicht funktionale Anforderungen enthält.²⁶ Die Anforderungen werden in sogenannten Sprints umgesetzt, welche jeweils eine 30-tägige Iteration darstellen. Begonnen wird jeder Sprint mit einem Sprint Planning Meeting, in welchem die in diesem Sprint umzusetzenden Anforderungen aus dem Product Backlog gewählt werden. Täglich werden Daily Scrum Meetings abgehalten, in denen innerhalb des Teams Probleme gelöst werden können und besprochen wird, was bereits umgesetzt wurde und was es heute umzusetzen gilt.²⁷ In dem darauf folgenden Sprint Review Meeting wird dem Product Owner und den Stakeholdern das Inkrement präsentiert. Abgeschlossen wird der Sprint mit einem Sprint Retrospective Meeting, in welchem das Team zusammen mit dem ScrumMaster versucht, seinen Entwicklungsprozess zu optimieren.²⁸

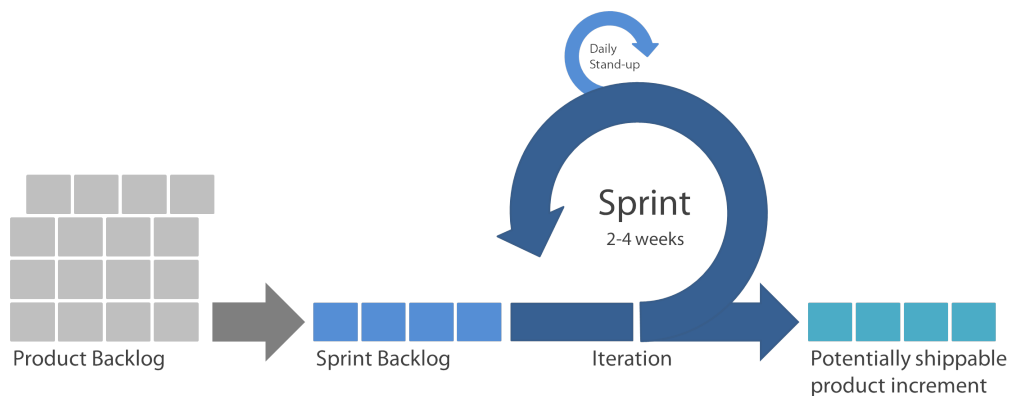


Abbildung 2: Ablauf eines Scrum-Sprints
Quelle: Costa (2016)

3.4.4 Scrum-Artefakte

Während des Scrum-Prozesses werden einige Werkzeuge bzw. Artefakte genutzt. Zum einen werden die für das im Rahmen des Projekts zu entwickelnde System notwendigen Anforderungen im Product Backlog definiert. Dieses wird vom Product Owner erstellt

²⁵Vgl. Schwaber (2012), Seite 7.

²⁶Vgl. ebd., Seite 8.

²⁷Vgl. Karam (2016).

²⁸Vgl. Schwaber (2012), Seite 9.

und kontinuierlich gepflegt. Es ist niemals vollständig, da sich die Anforderungen laufend ändern und dementsprechend muss das Dokument ebenfalls stetig angepasst werden. Zum anderen wird aus diesem Product Backlog ein Sprint Backlog definiert, welches die für den aktuellen Sprint umzusetzenden Aktivitäten enthält. Diese werden durch das Team ausgewählt, in Aufgaben mit einer Implementierungsdauer von 4-16 Stunden aufgeteilt und in eine Reihenfolge gebracht. Es stellt somit ein Echtzeitabbild der in diesem Sprint durchzuführenden Aufgaben dar. Außerdem werden am Ende eines Sprints potenziell auslieferbare Inkremente erzeugt. Damit diese Funktionalitäten kurzfristig implementierbar sind, müssen sie aus qualitativ hochwertigem und gründlich getestetem Code bestehen. Zusätzlich müssen die in diesem Inkrement implementierten Funktionalitäten in irgendeiner Form für den Anwender dokumentiert sein.²⁹

²⁹Vgl. Schwaber (2012), Seite 10-15.

4 Optimierung des Softwareentwicklungsprozesses

4.1 Evaluation der Modelle

4.2 Auswahl einer Lösung

5 Schlussbetrachtung

5.1 Fazit

5.2 Ausblick

Literaturverzeichnis

Bücher

Cockburn, Alistair: Agile software development (Vol. 177), Addison-Wesley Boston, 2002.

Highsmith, James A.: Agile software development ecosystems (Vol. 13), Addison-Wesley Professional, 2002.

Jeffries, Ron; Ann Anderson; Chet Hendrickson: Extreme programming installed Addison-Wesley Professional, 2001.

Pichler, Roman; Stefan Roock: Agile Entwicklungspraktiken mit Scrum Dpunkt. verlag, 2011.

Schwaber, Ken: Agiles Projektmanagement mit Scrum Microsoft Press, 2012.

Internet-Quellen

Beck, Kent et al., Manifest für Agile Softwareentwicklung, URL: [http : / / agilemanifesto.org/iso/de/manifesto.html](http://agilemanifesto.org/iso/de/manifesto.html), 13.07.2017, 18:42.

Beck, Kent et al., Prinzipien hinter dem Agilen Manifest, URL: [http : / / agilemanifesto.org/iso/de/principles.html](http://agilemanifesto.org/iso/de/principles.html), 13.07.2017, 18:41.

Costa, Rui, Who, Where, What, When, Why and How to use Scrum - A simple guide, URL: [https : // www . linkedin . com / pulse / who - where - what - when - why - how - use - scrum - simple - guide - rui - costa](https://www.linkedin.com/pulse/who-where-what-when-why-how-use-scrum-simple-guide-rui-costa), 15.07.2017, 14:26.

Karam, Lea, Extreme Programming; Tips & Advantages, URL: [https : // apiumhub . com / tech - blog - barcelona / scrum - sprint - explanation /](https://apiumhub.com/tech-blog-barcelona/scrum-sprint-explanation/), 29.06.2017, 22:35.

Karam, Lea, Extreme Programming; Tips & Advantages, URL: [https : // apiumhub . com / tech - blog - barcelona / feature - driven - development /](https://apiumhub.com/tech-blog-barcelona/feature-driven-development/), 29.06.2017, 22:34.

Novoseltseva, Ekaterina, Extreme Programming; Tips & Advantages, URL: [https : // apiumhub . com / tech - blog - barcelona / extreme - programming - tips - advantages /](https://apiumhub.com/tech-blog-barcelona/extreme-programming-tips-advantages/), 29.06.2017, 22:30.

Eidesstattliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe.

(Ort, Datum)

(Eigenhändige Unterschrift)