FERDINAND PORSCHE
FERN FH

Methoden und Kriterien zur Auswahl agiler Vorgehensmodelle in der Softwareentwicklung

Bachelorarbeit

eingereicht von: **Julia Zeilermayr**
Matrikelnummer: 01141163

im Fachhochschul-Bachelorstudiengang Wirtschaftsinformatik (0470)
der Ferdinand Porsche FernFH

zur Erlangung des akademischen Grades einer

Bachelor of Arts in Business

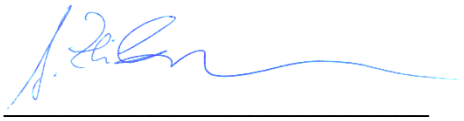
Betreuung und Beurteilung: Mag. Dr. Michael Derntl

Wiener Neustadt, Mai 2022

Ehrenwörtliche Erklärung

Ich versichere hiermit,

1. dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Inhalte, die direkt oder indirekt aus fremden Quellen entnommen sind, sind durch entsprechende Quellenangaben gekennzeichnet.
2. dass ich diese Bachelorarbeit bisher weder im Inland noch im Ausland in irgendeiner Form als Prüfungsarbeit zur Beurteilung vorgelegt oder veröffentlicht habe.



Wien, Julia Zeilermayr, 14.05.2022

Unterschrift

Die Kapitel 2 bis 4 der vorliegenden Bachelorarbeit wurden im Rahmen der Lehrveranstaltung „Bachelor Seminar 1“ eingereicht und am 07.03.2022 als Bachelorarbeit 1 angenommen.

Kurzzusammenfassung: Methoden und Kriterien zur Auswahl agiler Vorgehensmodelle in der Softwareentwicklung

Diese Arbeit beschäftigt sich mit agilen Vorgehensmodellen und ihren Konzepten und stellt im Zuge dessen auch die Unterschiede zwischen klassischen Vorgehensmodellen und der agilen Welt dar. Es werden die gängigsten agilen Vorgehensmodelle Scrum, Extreme Programming, Kanban und Crystal Family vorgestellt. Dabei wird sowohl auf die Rollen, Charakteristiken und Besonderheiten der einzelnen Modelle eingegangen, um den Leser:innen eine Vorstellung der Anwendung dieser Modelle zu vermitteln. Anschließend werden Kriterien gezeigt, welche bei der Wahl eines passenden Vorgehensmodells unterstützen können.

Es werden zwei reale Teams innerhalb eines IT Großkonzerns vorgestellt, für die ein passendes agiles Vorgehensmodell gefunden werden soll. Sowohl das Projektumfeld als auch die Eigenschaften der Teams werden dafür betrachtet. Anhand einer zuvor definierten, qualitativen Beurteilungsmethode wird mittels Punktesystem geprüft, welches Modell sich für das jeweilige Team eignet. Abschließend folgt eine Interpretation der Ergebnisse

Schlagwörter:

Agile Vorgehensmodelle, Scrum, Kanban, Crystal Family, Extreme Programming, XP

Abstract: methods and criteria for choosing agile process models in software development

This thesis deals with agile process models and the agile methodology itself. Outlining the differences between classic and agile process models, four of the most common agile process models are presented: Scrum, Extreme Programming, Kanban and Crystal Family. The focus herein lies on the roles and characteristics of the different models. A list of potential criteria is introduced to help identify the most suitable agile process model for a given team.

Subsequently, a case study based on two teams within a globally operating IT company has been conducted, in order to validate the identified list of selection criteria. Both teams are working in the same IT company. Therefore the background of the company including the project the teams are working for is presented jointly for both teams. By using a predefined evaluation method, the four agile process models are evaluated by cross-checking them with the previously defined criteria. This evaluation is done for both teams so that a well-fitting agile process model can be determined for each team.

Keywords:

Agile process models, scrum, kanban, crystal family, extreme programming, XP

Inhaltsverzeichnis

1. Einleitung

1.1 Ausgangslage

Software ist dazu da, uns das Leben zu erleichtern. Sie soll uns bei Abläufen des täglichen Lebens unterstützen, Aufgaben automatisieren und uns dadurch Zeit- und Kostenersparnisse verschaffen.

Bevor das jedoch geschehen kann, muss die Software erst einmal entwickelt werden. Dies ist mit einer Menge an komplexen Prozessen verbunden, die durchlaufen werden müssen, bevor die Software ausgeliefert werden kann. In dieser Arbeit möchte ich mich mit Methoden beschäftigen, die uns dabei helfen, sowohl die Softwareentwicklung als auch den IT Betrieb effizienter und strukturierter abzuhandeln – den Vorgehensmodellen.

Mein persönlicher beruflicher Werdegang in der IT hat im Bereich IT Betrieb begonnen. Obwohl der IT Betrieb maßgeblich durch operative Tätigkeiten geprägt ist, konnte ich dennoch feststellen, dass Projekte auch hier eine große Rolle spielen. Der Spagat, der notwendig ist, um beide Bereiche unter einen Hut zu bekommen, stellt eine besondere Herausforderung dar. Aus diesem Grund habe ich bereits vor einiger Zeit begonnen, mich mit den unterschiedlichen Vorgehensmodellen zu beschäftigen, die sich sowohl in der Softwareentwicklung als auch im IT Betrieb einsetzen lassen.

Um ein geeignetes Vorgehensmodell für einen Anwendungsfall wählen zu können, müssen zu allererst die zur Verfügung stehenden Vorgehensmodelle evaluiert und auf ihre Gemeinsamkeiten und Unterschiede analysiert werden. In der Praxis ist es in den meisten Fällen so, dass ein geeignetes Vorgehensmodell immer an die jeweiligen Gegebenheiten des Unternehmens angepasst werden muss.

Ich möchte in dieser Arbeit erforschen, nach welchen Kriterien vorgegangen werden kann, um das im individuellen Fall beste Vorgehensmodell herauszufinden.

Als Analyseobjekte zur Evaluierung der jeweiligen Vorgehensmodelle werden zwei reale Teams herangezogen, die im selben Unternehmen arbeiten. Bei diesem Unternehmen handelt es sich um einen Großkonzern in der IT Branche, der weltweit agiert und dessen Kernkompetenz in der Entwicklung und dem Betrieb von Versicherungssoftware liegt. Die unterschiedlichen Teams des Unternehmens sind auf der ganzen Welt verstreut und arbeiten interdisziplinär zusammen.

Das erste Team, das betrachtet wird, befasst sich mit Deployment-Automatisierung von IT Services. Es wird im Folgenden „Team Entwicklung“ genannt, da die Hauptaufgabe in der Entwicklung von Skripts liegt, die vollautomatisiert das Deployment von IT Services ausführen sollen.

Das zweite Team ist für den IT Betrieb derselben IT Services bei mehreren Kunden verantwortlich. Es wird daher „Team Betrieb“ genannt. Neben klassischen Betriebstätigkeiten werden zusätzlich kleinere Projekte abgehandelt.

Im Laufe meiner beruflichen Entwicklung hatte ich mit beiden Teams zu tun, hauptsächlich in koordinativer Tätigkeit. Dabei wurde mir bewusst, dass ein Vorgehensmodell nicht universell auf verschiedene Teams anzuwenden ist, sondern es Kriterien geben muss, nach denen sich die Wahl für ein geeignetes Vorgehensmodell treffen lässt. Aus diesen Überlegungen hat sich letztendlich die Fragestellung ergeben, die ich mit dieser Arbeit untersuchen möchte.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es zu zeigen, welches agile Vorgehensmodell sich im jeweiligen Anwendungsfall am besten eignet. Vor allem IT Manager:innen oder auch Teamleiter:innen stehen in der Praxis oft vor dem Problem, wie sie ihre Teams am besten organisieren sollen, um die höchste Produktivität zu erreichen. Bei der großen Anzahl an Vorgehensmodellen und ihren Varianten ist es oft schwer, einen Überblick zu behalten. Daher ist es ebenso ein Ziel der Arbeit, die unterschiedlichen agilen Methoden strukturiert zu beschreiben, um sie anschließend besser miteinander vergleichbar zu machen. Es soll ein Kriterienkatalog geschaffen werden, anhand derer sich die Wahl eines Vorgehensmodell bestmöglich treffen lässt.

Des Weiteren sollen die Gemeinsamkeiten und Unterschiede zwischen dem Entwicklungsteam und dem Betriebsteam herausgearbeitet werden. Den Lesenden soll nachvollziehbar erläutert werden, aus welchen Gründen sich eine Methode für ein Team eignet, für ein anderes aber weniger. Ein weiterführendes Ziel soll sein, dass die Lesenden dieser Arbeit die Kriterien auch bei anderen Teams anwenden können.

1.3 Forschungsfrage

Die konkrete Forschungsfrage dieser Arbeit lautet wie folgt:

Welches agile Vorgehensmodell eignet sich am besten für das „Team Entwicklung“ bzw. das „Team Betrieb“ unter Berücksichtigung der verschiedenen Zielsetzungen?

1.4 Hypothese

Meine Hypothese besagt, dass sich für das „Team Entwicklung“ das Vorgehensmodell Scrum am besten eignet und für das „Team Betrieb“ die Kanban Methode.

Der Grund für diese Theorie liegt in meiner persönlichen Erfahrungen. Mit dem „Team Entwicklung“ habe ich selbst bereits nach der Scrum Methode gearbeitet und in meiner Rolle als Scrum Master beobachtet, dass sich die Methode für das Team eignet. Es ist

jedoch zu erwähnen, dass das Team auch bisher keine anderen agilen Vorgehensmodelle versucht hat.

Beim „Team Betrieb“ stützt sich meine Theorie darauf, dass dieses Team häufig mit ad hoc Anforderungen und Betriebstätigkeiten umgehen muss, und daher eine gewisse Flexibilität gegeben sein muss. Die Kanban Methode lässt diese Flexibilität aus meiner heutigen Sicht zu, und erscheint mir daher als gut geeignet.

1.5 Aufbau der Arbeit

Am Beginn der Arbeit werden grundlegende IT Begriffe und IT Prozesse erklärt. Es wird der Begriff der Softwareentwicklung definiert und der Software-Lebenszyklus erläutert. Anschließend wird auf den IT Betrieb näher eingegangen.

Der nächste Abschnitt behandelt die Definition von Vorgehensmodellen sowie die Unterscheidung zwischen klassischen und agilen Vorgehensmodellen. Hierfür wird vor allem auf den Begriff der Agilität näher eingegangen, der anhand des agilen Manifestes verdeutlicht werden soll.

Anschließend werden vier agile Vorgehensmodelle beispielhaft näher beschrieben: Scrum, Extreme Programming, Kanban und Crystal Family. Neben deren Entstehung soll der Grundgedanke jeden Modelles vermittelt werden, sowie die darin vorkommenden Rollen und Praktiken.

Im letzten Abschnitt von Teil 1 werden verschiedene Kriterien herausgearbeitet, die dazu dienen können, ein geeignetes Vorgehensmodell auszuwählen.

Im zweiten Teil der Arbeit wird zuerst eine Bewertungsmethode vorgestellt, die anhand der beschriebenen Kriterien beurteilen lässt, wie gut sich ein Vorgehensmodell für das betrachtete Team eignet.

Als Nächstes wird das Unternehmen und das Projektumfeld behandelt, in welchem beide Teams agieren. Dadurch ergeben sich die Rahmenbedingungen, die für beide Teams berücksichtigt werden müssen.

Darauffolgend wird das erste Team („Team Entwicklung“) näher vorgestellt, inklusive seiner Aufgabenbereiche und Teamzusammensetzung. Die Methode wird bei „Team Entwicklung“ angewandt und ausgewertet. Selbiges wird ebenso beim zweiten Team, „Team Betrieb“, durchgeführt.

Nach einer ausführlichen Interpretation der Ergebnisse folgt die Conclusio inklusive Beantwortung der Forschungsfrage, sowie eine Zusammenfassung und ein Ausblick.

2. Vorgehensmodelle in der Softwareentwicklung

Um Vorgehensmodelle in der Softwareentwicklung genauer betrachten zu können, müssen zunächst die Begriffe Softwareentwicklung und IT Betrieb definiert werden.

2.1 Begriffsdefinitionen

2.1.1 Softwareentwicklung

Softwareentwicklung ist ein Prozess, der alle Tätigkeiten und Ressourcen umschließt, die notwendig sind, um Software herzustellen. Da Softwareentwicklung in den wenigsten Fällen von einzelnen Personen, sondern meist in größeren Gruppen bzw. Organisationsstrukturen erfolgt, wird sie meistens in Form von Projekten abgehandelt. (Klinger 2010)

Gemäß DIN69901 ist ein Projekt „ein Vorhaben, das im Wesentlichen durch Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist“. Es hat immer einen Anfang, ein Ende und eine konkrete Zielsetzung, die es zu erfüllen gilt. Das Projektmanagement wird durch Projektleiter:innen übernommen, welche unter anderem für Termin- und Ressourcenplanung, die Koordination der Arbeitsaufträge, die Steuerung und Überwachung sowie für das Projektreporting zuständig ist. (Klinger 2010)

Die Softwareentwicklung selbst gliedert sich und mehrere Phasen, die im sogenannten Software Lebenszyklus dargestellt werden. Dieser setzt sich aus folgenden Phasen zusammen (Derntl, Schikuta und Wanek 2019):

Anforderungsanalyse

In dieser Phase werden die konkreten Anforderungen analysiert, welche die Software erfüllen soll. Es ist vor allem wichtig, die Hintergründe der Anforderungen aus Sicht der Anwender:innen zu verstehen und den Ist-Zustand zu beleuchten, um eine möglichst zufriedenstellende Lösung produzieren zu können.

Design

Hier werden die Anforderungen konkret ausformuliert und die grundlegende Softwarearchitektur, die Datenstrukturen sowie die notwendigen Schnittstellen definiert. Das Resultat wird anschließend in Form von Modellen, Diagrammen und zusätzlichen textuellen Spezifikationen dargestellt.

Implementierung

Bei der Implementierung findet die eigentliche Programmierung bzw. Erstellung des Quellcodes statt.

Testen

In der Testphase erfolgt die Überprüfung, ob die entwickelte Software den gewünschten Anforderungen entspricht. Diese Phase hat einen besonderen Einfluss auf die Qualität der Software, und wird leider in vielen Projekten auf Zeitgründen stiefmütterlich behandelt. Erst wenn das Testen abgeschlossen und die Fehler behoben wurden kann die Software freigegeben werden.

Release

Die Release bezeichnet die Freigabe und Produktivsetzung von Software. Da Software in den meisten Fällen nicht nur einmal released, sondern stets weiterentwickelt wird, ist es von besonderer Wichtigkeit, dass die jeweilige Softwarekomponente versioniert und gut dokumentiert ist.

Wartung

Bereiche der Wartungsphase sind einerseits Fehlerbehebungen, Adaptierungen an neue Umgebungen, Hardware oder Betriebssysteme, und Aktualisierungen. Diese Phase dauert so lange an, bis die Software außer Betrieb genommen wird.

2.1.2 IT Betrieb

Der Begriff des IT Betriebs ist so umfangreich, dass es wichtig ist, ihn für diese Arbeit abzugrenzen bzw. zu definieren. Der IT Betrieb (auch engl. IT Operations genannt) ist in den meisten Unternehmen eine abgegrenzte Organisationseinheit. Die Aufgabe des IT Betriebes ist es, „die Hardware und die zum Betrieb der Hardware erforderliche Software in angemessenem Umfang zur Verfügung zu stellen und störungsfrei zu betreiben. … Im Störungsfall dient der IT Betrieb als zentrale Ansprechstelle der Anwender, bei Ausfällen hat er für die möglichst kurzfristige Wiederherstellung der Verfügbarkeit Sorge zu tragen.“ (Hofer 2022)

Um einen reibungslosen IT Betrieb sicherzustellen, sind gut definierte und funktionierende Prozesse unabdingbar. Aus diesem Grund wurde die Information Technology Infrastructure Library (ITIL) entwickelt, welche eine Sammlung an Prozess-Best-Practices zur Verfügung stellt, die im IT Betrieb notwendig sind. (Pfitzinger und Jestädt 2016)

Ein großer Unterschied zwischen Softwareentwicklung und IT Betrieb liegt in der Planbarkeit. Während Softwareentwicklung in der Regel gut in Projekten abgehandelt werden kann, ist der IT Betrieb weniger gut planbar. Ein beim Endanwender auftretender Fehler, ein Netzwerkausfall oder sonstige betriebsstörende Vorfälle müssen im Normalfall rasch behandelt werden und unterbrechen die Arbeit an Projekten. Somit ist der IT Betrieb stark von ad hoc Anforderungen geprägt, die sich nicht durch Projekte planen lassen.

Es gibt allerdings auch im IT Betrieb Aktivitäten, die sich durch Projekte abwickeln lassen. Beispielsweise können geplanter Hardwareaustausch, Software Versionsupgrades oder Infrastruktur Rolloutprojekte in Form von Projekten organisiert werden. Die Tatsache, dass der IT Betrieb somit gleichzeitig in Betriebsprozessen als auch in Projekten involviert ist, stellt eine besondere Planungsschwierigkeit dar.

Ein Bereich des IT Betriebs, der in dieser Arbeit eine besondere Rolle spielen soll, ist die Bereitstellung von IT Services (engl. Deployment). Deployment ist der Prozess der Installation und Konfiguration von Software auf PCs oder Servern. (AROCOM GMBH 2022)

Seit der Entwicklung von Cloud Technologien hat sich dieser Bereich stark verändert. Während Deployments früher mit vielen manuellen Schritten verbunden waren, lassen sich diese dank moderner Cloud Technologien gut automatisieren. Aus diesem Grund kommt auch dem Bereich des IT Betriebes eine neue Rolle zu: Neben seinen betrieblichen Tätigkeiten kommen für ihn nun auch Entwicklungstätigkeiten hinzu, zum Beispiel zur Automatisierung von Deployments.

Diese Automatisierung kann wiederum, ebenso wie die klassische Softwareentwicklung, in Form von Projekten abgehandelt werden. Um Projekte effizient durchführen zu können, ist wiederum die Auswahl des richtigen Vorgehensmodells äußerst wichtig und trägt maßgeblich zum Erfolg des Projektes bei.

2.2 Vorgehensmodelle

Ein Vorgehensmodell kann als eine Sammlung an Methoden und Verhaltensweisen angesehen werden, auf die sich eine Organisation innerhalb eines Projektes einigt, um effizient zusammen zu arbeiten. Es beschreibt sowohl die Prozesse, Rollen als auch die Artefakte innerhalb des Projektes. (Klinger 2010)

Unterschiedlichste Vorgehensmodelle sind im Laufe der Zeit durch Erfahrungen in Projekten entstanden und wurden aufgegriffen und weiterentwickelt. Diejenigen, die sich in der Softwareentwicklung am meisten bewährt haben, werden in den folgenden Abschnitten vorgestellt. Grundsätzlich kann bei Vorgehensmodellen zwischen klassischen Modellen und agilen Modellen unterschieden werden.

Der Vorteil der Anwendung von Vorgehensmodellen im Allgemeinen besteht darin, die Kommunikation zwischen Entwickler:innen, Designer:innen, Tester:innen, Anwender:innen und dem Management zu verbessern. Des Weiteren soll das Vorgehensmodell zum Verständnis der komplexen Softwareentwicklungsprozesse beitragen und die Planbarkeit erhöhen. Zusätzlich kann durch die Anwendung des richtigen Vorgehensmodells die Qualität des Projektes gesteigert werden, vor allem durch strukturierte Prozesse und Dokumentation. (Klinger 2010)

2.3 Klassische Vorgehensmodelle

Das bekannteste Beispiel für klassische Vorgehensmodelle ist das Wasserfallmodell. Zusätzlich zum Wasserfallmodell haben sich noch weitere Modelle etabliert, wie beispielsweise das V-Modell, das Spiralmodell oder das Rapid Prototyping Modell. Das Wasserfallmodell wird hier daher stellvertretend für die klassischen Vorgehensmodelle näher ausgeführt.

Klassische Vorgehensmodelle bestehen aus sogenannten Phasen, die im Laufe des Projektes nacheinander durchlaufen werden. Im Beispiel der Softwareentwicklung sind diese Phasen, wie bereits oben beschrieben, die Anforderungsanalyse, das Design, die Implementierung, das Testen, der Release und die Wartung. Diese Phasen können jedoch je nach Autor:in leicht variieren bzw. anders benannt werden. (Derntl, Schikuta und Wanek 2019)

Das Wasserfallmodell wurde 1970 von Winston Royce entwickelt. Charakteristisch für dieses Modell ist, dass eine Phase vollständig abgeschlossen sein muss, bevor mit der nächsten Phase begonnen werden kann. Aufgrund dieser strikten Abfolge ergibt sich auch der Name Wasserfallmodell, da alle Phasen nach dem Top-down Prinzip in einer exakten Reihenfolge nacheinander abgearbeitet werden. Am Abschluss jeder Phase liegt ein Meilenstein, der vorgibt, zu welchem Zeitpunkt die Ziele der jeweiligen Phase erreicht sein müssen. (Klinger 2010)

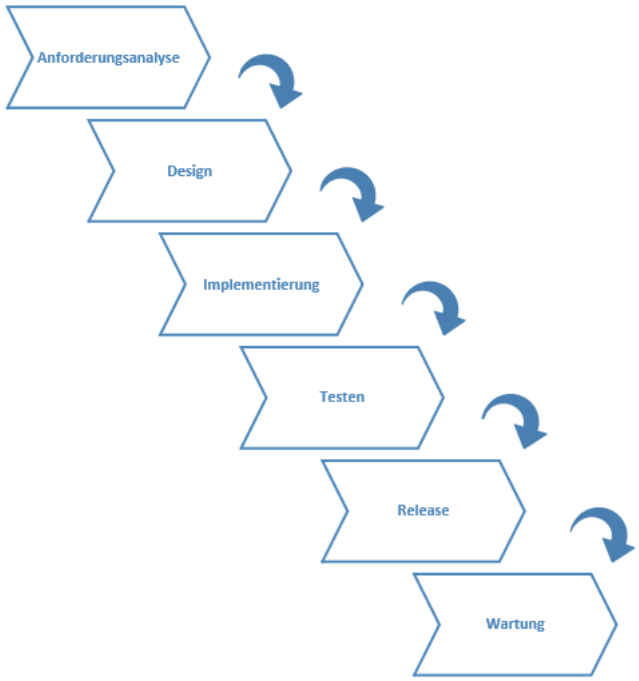


Abbildung 1 Wasserfallmodell

In moderneren bzw. verbesserten Versionen des Wasserfallmodells sind zusätzlich Rückkoppelungen in Form von Informationsfluss zur davorliegenden Phase vorgesehen.

Wie unschwer zu erkennen ist, bedarf es bei der Anwendung dieses Modells einer umfangreichen und genauen Planung am Beginn des Projektes. Aus diesem Grund eignet sich dieses Modell auch nur für Projekte, bei denen die Anforderungen am Beginn des Projektes bereits vollständig bekannt sind und sich später nicht mehr ändern – das ist in der Softwareentwicklung nur in Ausnahmefällen der Fall. Die Einbeziehung der Anwender:innen ist beim Wasserfallmodell nur am Anfang des Projektes, in der Anforderungsanalyse, vorgesehen.

Eine besondere Herausforderung in der Softwarebranche ist, dass stets flexibel und schnell reagiert werden muss, um rechtzeitig auf die Bedürfnisse der Kund:innen bzw. des Markes eingehen zu können, bevor es jemand anderes tut. Genau diese Flexibilität ist bei klassischen Vorgehensmodellen nicht gegeben. Bei Projekten mit einem fixen Enddatum kommt ein weiterer Nachteil hinzu: Wenn sich die frühen Phasen des Projektes verzögern, ergibt sich das Risiko, dass die eigentliche Implementierung inkl. Testen am Schluss nicht fertiggestellt werden kann und am Ende gar kein Softwareprodukt zur Verfügung steht.

2.4 Agile Vorgehensmodelle

Agile Vorgehensmodelle entstanden erstmals Anfang der 1990er Jahre. Der Bedarf an veränderten Herangehensweisen ergab sich aus dem Wunsch nach mehr Flexibilität. Klassische Modelle waren nicht mehr in der Lage, sich schnell genug an stetigen Wandel und sich verändernde Markt- und Kundenbedürfnisse anzupassen. (Engel 2011)

Ziel der agilen Vorgehensmodelle war es, wie der Name schon sagt, die Softwareentwicklung „beweglicher“ zu machen. Es sollte schneller auf Änderungen von Anforderungen reagiert werden können und das Ziel einer funktionierenden Software im Mittelpunkt stehen. Voraussetzungen dafür sind gute Kommunikation, Flexibilität und dynamische Abläufe. (Klinger 2010)

Um agile Vorgehensmodelle genauer erläutern zu können, muss zuerst „Agilität“ an sich definiert werden.

„Agilität ist die Fähigkeit einer Organisation, flexibel, aktiv, anpassungsfähig und mit Initiative in Zeiten des Wandels und Unsicherheit zu agieren.“ (Onpulson-Lexikon 2021)

Diese Flexibilität bezieht sich sowohl auf geänderte Anforderungen, Technologien als auch den Markt. Veränderungen sollen nicht als Bedrohung, sondern vielmehr als Chance angesehen werden, die agile Bewegung steht dem Wandel also positiv gegenüber.

Softwareentwicklung kann nach Jim Highsmith als agil angesehen werden, wenn sie folgende zwei Bedingungen erfüllt (Highsmith 2002):

1. Wandel wird akzeptiert und bewältigt
2. Wandel wird genutzt und zum eigenen Vorteil ausgelöst

Nach Dogs und Klimmer definiert sich die agile Methodik durch folgende vier Eigenschaften, wodurch evaluiert werden kann, ob sie agil ist oder nicht:

„Damit eine Methodik als agil bezeichnet wird,

- muss mit ihr Software inkrementell (Stück für Stück der Reihe nach) entwickelt werden,
- sie muss eine ausgeprägte Kommunikation unter allen Projektbeteiligten forcieren,
- sie muss unkompliziert und leicht erlernbar sein,
- und ebenso muss sie adaptiv sein, d.h. sie muss selbst zu einem späten Zeitpunkt noch größere Änderungen an den Arbeitsprodukten ermöglichen.“ (Dogs und Klimmer 2005)

Grundsätzlich finden sich in der agilen Softwareentwicklung die gleichen Phasen wieder, wie auch schon in den klassischen Vorgehensmodellen (Anforderungsanalyse, Design, Implementierung, Testen, Release und Wartung). Jedoch werden sie hier in viel kürzen Zyklen bzw. Iterationen durchlaufen, die sich immer wieder wiederholen. Da alle Phasen in einer Iteration durchlaufen werden (inklusive Inbetriebnahme), ergibt sich auch am Ende jeder Iteration bereits ein funktionsfähiges Produkt – im Gegensatz zu den klassischen Modellen muss darauf also nicht bis zum Ende des Projektes gewartet werden. Die Entwicklung erfolgt somit inkrementell, sodass nach jeder Iteration ein Stück lauffähiger Software hinzukommt.

Durch diese Herangehensweise ergibt sich auch die schnelle Reaktionsfähigkeit und Flexibilität: Es muss immer nur so viel festgelegt und entschieden werden, wie für die kommende Iteration notwendig ist. Anschließend kann eine erneute Evaluation der Situation erfolgen.

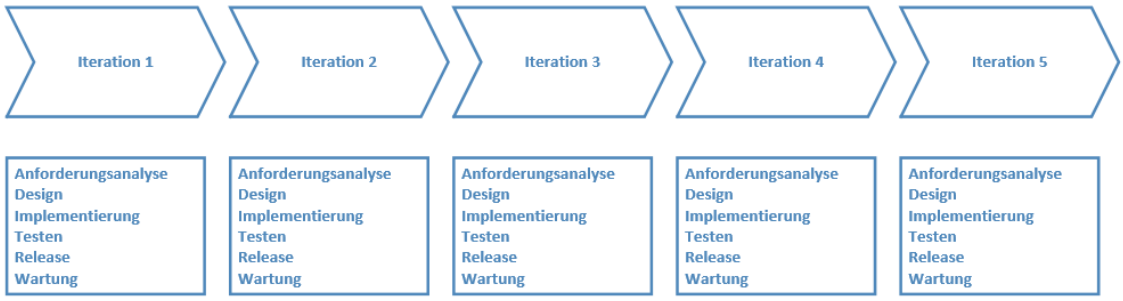


Abbildung 2 Phasen und Iterationen bei agilen Vorgehensmodellen

2.4.1 Das Agile Manifest

Das Agile Manifest entstand bei einer Zusammenkunft in Snowbird, Utah, im Februar 2001, wo 17 Softwareentwickler:innen bzw. Vertreter:innen der damals bestehenden agilen Methoden und Ansätze zusammentrafen. Sie gründeten die Agile Allianz und

erstellten das Agile Manifest, welches von allen anwesenden Softwareentwickler:innen unterzeichnet wurde. Das Agile Manifest setzt sich aus den 4 Werten und 12 Prinzipien agiler Softwareentwicklung zusammen.

Die agilen Werte definieren sich wie folgt:

„Wir erschließen bessere Wege, Software zu entwickeln,
indem wir es selbst tun und anderen dabei helfen.
Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

Individuen und Interaktionen mehr als Prozesse und Werkzeuge
Funktionierende Software mehr als umfassende Dokumentation
Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung
Reagieren auf Veränderung mehr als das Befolgen eines Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden,
schätzen wir die Werte auf der linken Seite höher ein.“
(Beck, et al. 2001)

Hier ist vor allem zu beachten, dass es nicht um ein entweder oder geht, und somit auch die Werte auf der rechten Seite nicht völlig außer Acht gelassen werden dürfen. Es soll vielmehr betont werden, dass die Wichtigkeit und Priorität auf den Werten der linken Seite liegt.

Die 12 Prinzipien ergeben sich wie folgt (Beck, et al. 2001):

„Wir folgen diesen Prinzipien:

1. Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.
2. Heisse Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
3. Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.
4. Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.

5. Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.
6. Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.
7. Funktionierende Software ist das wichtigste Fortschrittsmaß.
8. Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.
9. Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
10. Einfachheit -- die Kunst, die Menge nicht getaner Arbeit zu maximieren -- ist essenziell.
11. Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.
12. In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.“

Zusammenfassend kann festgehalten werden, dass die kontinuierliche Lieferung funktionsfähiger Software, sowie die Kundenzufriedenheit im Mittelpunkt stehen. Es geht weniger darum, über welchen Weg ein Ziel erreicht wird, als darum, ans Ziel zu kommen. Die Kund:innen sollen als Partner:innen innerhalb des Projektes angesehen werden, mit denen permanent Kontakt gehalten wird. Abgesehen davon wird auch grundsätzlich im gesamten Projektumfeld der Fokus auf direkte, persönliche Kommunikation gelegt.

3. Vertreter agiler Vorgehensmodelle

3.1 Scrum

3.1.1 Entstehung

Das wohl bekannteste und in der Wirtschaft vielfach eingesetzte Bespiel agiler Vorgehensmodelle ist Scrum. Inspiriert von den Japanern Ikujiro Nonaka und Hirotaka Takeuchi, die im Jahre 1987 die ersten Ansätze von Scrum beschrieben, gelten letztendlich Ken Schwaber und Jeff Sutherland als Begründer von Scrum in der agilen Softwareentwicklung, welche dies erstmals in den 1990er Jahren präsentierten. (Klinger 2010)

Der Begriff „Scrum“ stammt ursprünglich aus dem Bereich des Rugby Sports und wird wörtlich mit „Gedränge“ übersetzt. Beim Rugby beschreibt Scrum eine Situation, in der sich alle Teammitglieder dicht gedrängt an- und hintereinander reihen, um mit vereinten Kräften die gegnerische Mannschaft von ihrer Position abdrängen zu können. Diese Symbolik haben Schwaber und Sutherland aufgegriffen, da auch beim Vorgehensmodell Scrum vor allem der Teamgeist im Vordergrund steht.

3.1.2 Grundgedanke

Scrum geht, wie die meisten agilen Vorgehensmodelle, von sich oft ändernden Anforderungen aus. Aus diesem Grund wird in kurzen Iterationen geplant, sogenannten Sprints. Ein Sprint kann zwischen einer und vier Wochen andauern, in vielen Fällen werden 2- oder 3-wöchige Sprints gewählt. Da nur für die Dauer eines Sprints konkret geplant wird, kann schnell auf geänderte Rahmenbedingungen reagiert werden.

Der Fokus von Scrum liegt auf dem Management von Teams, wobei keine konkreten Entwicklungspraktiken vorgegeben werden. Es wird außerdem großer Wert auf Selbstorganisation der Teams gelegt, Intervention des höheren Managements ist nur in Ausnahmefällen nötig. Ein Scrum Team plant eigenständig und verpflichtet sich somit selbst zur Lieferung der selbst geplanten Anforderungen. (Engel 2011)

3.1.3 Rollen

Scrum Team

Das Scrum Team besteht aus ca. 5 bis 10 Personen und setzt sich aus Entwickler:innen, Designer:innen, Tester:innen und gegebenenfalls auch weiteren Expert:innen zusammen. Das Team soll eigenständig in der Lage sein, Software-Inkremente zu liefern, und dabei möglichst wenig Abhängigkeiten zu anderen Teams haben. Aus diesem Grund soll das Team interdisziplinär besetzt sein und End-to-End alle benötigten Fähigkeiten abdecken.

Scrum Master:in

Scrum Master:innen unterstützen das Team beim Erfüllen seiner Tätigkeiten. Sie müssen über sehr gute Kenntnisse der Scrum Methoden verfügen und sorgen dafür, dass diese im Team eingehalten werden. Sie nehmen an allen Scrum Zeremonien teil und moderieren diese. Außerdem ist es die Aufgabe der Scrum Master:innen, Hindernisse aus dem Weg zu räumen, die das Entwicklungsteam beim Erfüllen seiner Aufgaben hindert. Scrum Master:innen können auch mehr als ein Scrum Team betreuen, sind aber pro Team durch nur eine Person vertreten.

Product Owner:in

Product Owner:innen sind verantwortlich für den Product Backlog. Dieser ist eine Liste an Anforderungen, die das zu entwickelnde Softwareprodukt erfüllen soll. Product Owner:innen sorgen dafür, dass alle Anforderungen in den Product Backlog eingepflegt werden. Sie stehen somit auch in regem Austausch mit den Kund:innen, und priorisieren in Abstimmung mit ihnen die Anforderungen. Pro Team sind Product Owner:innen nur durch eine Person verteten.

Kund:in

Kund:innen, welche durch eine oder auch mehrere Personen vertreten sein können, stehen dem Entwicklungsteam bei Rückfragen zur Verfügung und legen mit den Product Owner:innen die Prioritäten der Anforderungen fest.

3.1.4 Zeremonien

Scrum gibt eine Reihe an Zeremonien vor, die innerhalb eines Sprints vom Team abgehalten werden. In Abbildung 3 werden diese bildlich zusammengefasst. (Kriegisch 2022)

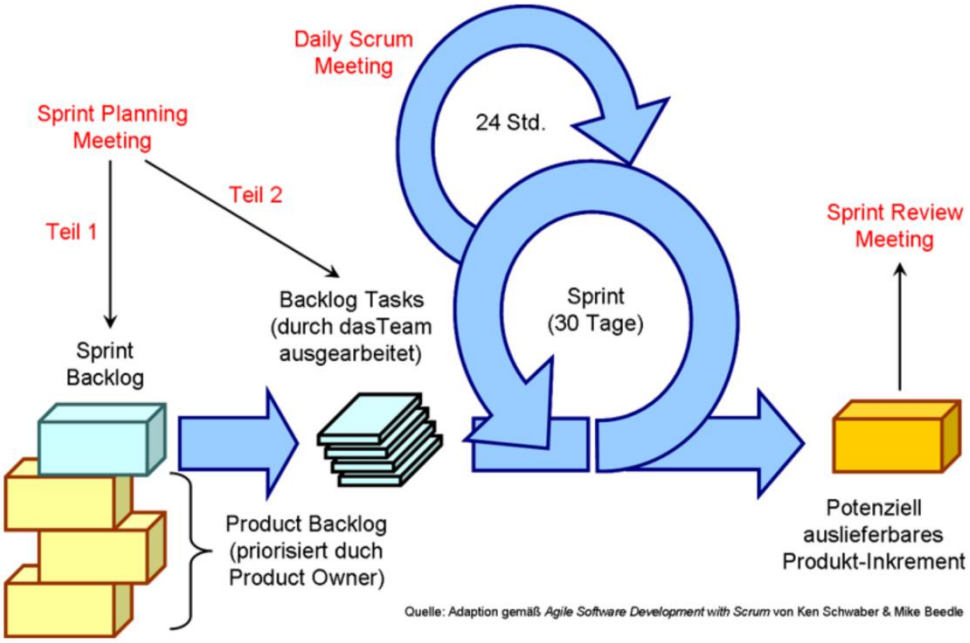


Abbildung 3: Zeremonien und Vorgangsweise in Scrum (Kriegisch 2022)

Daily Scrum

Das Daily Scrum, auch Daily Stand up Meeting genannt, findet, wie der Name schon sagt, täglich statt. Es dient dazu, dass die Teammitglieder sich über den aktuellen Status ihrer Tätigkeiten austauschen. Konkret sollen pro Teammitglied drei Fragen beantwortet werden: „Was habe ich seit dem letzten Daily Scrum erledigt?“, „Was werde ich bis zum nächsten Daily Scrum erledigen?“ und „Was behindert mich an meiner Arbeit?“. Das Meeting sollte im Stehen abgehalten werden und maximal 15 Minuten dauern. (Scrum.org 2022)

Sprint Planning

Das Sprint Planning findet immer am Beginn eines Sprints statt und dient dazu, die Aufgaben für den kommenden Sprint zu planen. Teilnehmer:innen sind sowohl das Scrum Team, die Scrum Master:innen als auch die Product Owner:innen, welche die Prioritäten für den nächsten Sprint vorgeben. Das Ergebnis des Sprint Plannings ist ein klar definierter Umfang an Aufgaben für den nächsten Sprint, welcher vom gesamten Team vereinbart wurde.

Sprint Review

Das Sprint Review findet jeweils am Ende eines Sprints statt. Hier präsentiert das Scrum Team, welche Funktionalitäten es im letzten Sprint entwickelt hat. Sowohl Kund:innen, Product Owner:innen als auch das Management sind eingeladen daran teilzunehmen.

Sprint Retrospektive

Abschließend zu jedem Sprint findet die Sprint Retrospektive statt, bei der das Scrum Team sich rückwirkend damit auseinandersetzt, wie der letzte Sprint gelaufen ist. Es wird diskutiert, was gut und was schlecht war und was in Zukunft verbessert werden sollte. Dieses regemäßige Meeting trägt somit stark zur kontinuierlichen Verbesserung bei.

Backlog Refinement

Beim Backlog Refinement werden die Anforderungen im Product Backlog vom Scrum Team genauer betrachtet. Es wird überprüft, ob die Anforderungen klar und verständlich definiert sind und gegebenenfalls die Aufgaben dazu verfeinert. Außerdem werden die Anforderungen hinsichtlich ihres Aufwandes bzw. ihrer Komplexität geschätzt, um später beim Sprint Planning besser abschätzen zu können, wieviel im Sprint umgesetzt werden kann.

3.1.5 Artefakte

Product Backlog

Der Product Backlog enthält alle Anforderungen an das Produkt. Er wird laufend geändert und ergänzt und durch die Product Owner:innen gewartet.

Sprint Backlog

Der Sprint Backlog ist jener Teil des Product Backlogs, der in einem Sprint abgehandelt wird. Sobald der Umfang des Sprint Backlogs für den nächsten Sprint definiert wurde, soll dieser im Laufe des Sprints nicht mehr geändert werden.

User Stories und Epics

Die Anforderungen werden in so genannte User Stories aufgeteilt. In einer User Story sind einerseits die geforderten Funktionalitäten aus Sicht der Benutzer:innen beschrieben und andererseits nach welchen Kriterien eine User Story als erledigt angesehen werden kann. Eine Gruppierung aus mehreren User Stories, die inhaltlich zusammen gehören, nennt sich Epic.

3.1.6 LeSS und SAFe

Wie bereits oben beschrieben soll ein Scrum Team aus maximal 10 Teammitgliedern bestehen. Um Scrum auch in größeren Firmen und Konzernen verwirklichen zu können, haben sich Methoden entwickelt, die es ermöglichen, Scrum zu skalieren.

LeSS (Larman und Vodde 2014)

LeSS steht für „Large Scaled Scrum”. Die Praktiken und Ideen entsprechen im Großen und Ganzen der Scrum Methodik, jedoch arbeiten hier mehrere Teams an ein und demselben Produkt. Es kann unterschieden werden zwischen LeSS, das auf bis zu 8

Teams mit je bis zu 8 Personen ausgelegt ist, und LeSS Huge, bei dem bis zu wenige Tausend Personen an einem Produkt arbeiten.

Bei LeSS gibt es einen einzigen gemeinsamen Product Backlog für alle Teams gemeinsam, der von einem einzigen Product Owner betreut wird. Die Teams arbeiten alle in einem gemeinsamen Sprint, sodass am Ende ein auslieferbares Produkt geliefert werden kann.

SAFe (Scaled Agile, Inc. 2020)

SAFe steht für „Scaled Agile Framework“ und ist ebenfalls eine Methode um Scrum in einem größeren Rahmen zu leben. Auch in SAFe gibt es mehrere agile Teams. Übergeordnet dazu findet sich hier jedoch noch eine weitere Organisationsebene, deren Aufgabe es ist, die unterschiedlichen agilen Teams aufeinander abzustimmen, um letztendlich das für die Kund:innen beste Produkt liefern zu können. Dadurch entstehen in SAFe weitere agile Rollen wie Product Manager:innen, System Architect:innen und Release Train Engineers (RTE), auf die an dieser Stelle nicht weiter eingegangen werden soll.

3.2 Extreme Programming (XP)

3.2.1 Entstehung

Extreme Programming wurde im Jahr 1996, vorwiegend von Kent Beck und Ward Cunningham entwickelt, die später ebenso Mitbegründer des Agilen Manifestes waren. Es handelt sich um ein agiles Vorgehensmodell, das sich vorrangig für kleine bis mittelgroße Teams eignet.

Der Name Extreme Programming (auch XP genannt) zeigt bereits, dass sich diese Methodik vorrangig auf die Entwicklung und weniger auf die begleitenden Management Prozesse konzentriert. Bei diesem Vorgehensmodell wird in kurzen Entwicklungszyklen sehr intensiv entwickelt, sodass sich das Team in diesem Zeitraum fokussiert der Entwicklung widmen kann. Durch eine sehr enge und unmittelbare Zusammenarbeit mit den Kund:innen werden Leerzeiten vermieden, da Unklarheiten sofort mit den Kund:innen abgeklärt und beseitigt werden können. (Klinger 2010)

3.2.2 Grundgedanke

Extreme Programming basiert auf fünf Werten, in denen sich die grundlegenden Prioritäten der Methode widerspiegeln. (Engel 2011)

Kommunikation:

Schnelle und persönliche Kommunikation, sowohl unter den Entwickler:innen als auch mit den Kund:innen ist der wichtigste Schlüssel zum Erfolg. Aus diesem Grund wird bei Extreme Programming auch empfohlen, dass alle Beteiligten sich an einem Ort aufhalten, um die Kommunikation zu erleichtern.

Einfachheit:

Das entwickelte Produkt bzw. die geforderte Funktionalität soll auf die einfachste Art und Weise implementiert werden und keine zusätzlichen Features beinhalten, die nicht explizit gefordert wurden. Auch der Quellcode soll so einfach wie möglich gehalten werden.

Feedback:

Feedback wird als positives Werkzeug angesehen, um die Qualität des Produktes zu verbessern. Die Mitarbeiter:innen werden daher ermutigt, ihre Meinungen zu Ideen kundzutun oder beispielsweise auch Rückmeldung zur Qualität des Quellcodes zu geben.

Mut:

Hierdurch soll betont werden, dass Mitarbeiter:innen neue und eigene Ideen einbringen sollen, und nicht vor neuartigen Herausforderungen zurückschrecken sollen. Dazu zählt auch, neue Technologien auszuprobieren und altbewährte Ansätze auch einmal in Frage zu stellen.

Respekt:

Respekt und Achtung voreinander ist ebenso eine Grundbedingung für gute Zusammenarbeit. Respekt ist außerdem eine Voraussetzung, um konstruktives Feedback geben und entgegen nehmen zu können. In Extreme Programming wird auch betont, dass jedes Teammitglied gleich viel wert ist.

3.2.3 Rollen

Programmierer:in

Eine wichtige Rolle spielt natürlich das Entwicklungsteam bzw. die Programmierer:innen. Sie sind für die Architektur, sowie für die Erstellung des Quellcodes und entsprechender Tests zuständig. Eine Besonderheit bei Extreme Programming liegt in der Tatsache, dass bereits vor Erstellung des Quellcodes Testfälle geschrieben werden, um im Nachgang überprüfen zu können, ob das entwickelte Inkrement die geforderten Funktionalitäten erfüllt. Außerdem sind die Programmierer:innen für Aufwandsschätzungen und die Planung der konkreten Umsetzung zuständig. (Wells 2013)

Kund:in

Neben dem Entwicklungsteam spielen auch die Kund:innen eine sehr wichtige und zentrale Rolle. Anders als in anderen Vorgehensmodellen werden sie als Teil des Teams angesehen und sind ständig präsent und involviert. Die Hauptaufgabe ist es, die Anforderungen zu definieren und zu priorisieren, bzw. die Reihenfolge vorzugeben, in welcher sie zu implementieren sind. Es wird empfohlen, dass die Kund:innen sich direkt vor Ort beim Entwicklungsteam befinden, sodass Unklarheiten zu den Anforderungen so schnell wie möglich aus dem Weg geschafft werden können. Die Rolle der Kund:innen kann entweder durch eine Person oder auch durch ein Team von Personen vertreten sein. (Klinger 2010)

Teamleiter:in bzw. Coach

Die Aufgabe der Teamleiter:in ist es vor allem, die Anwendung der Methodik und die Kommunikation im Team zu beobachten und bei Bedarf unterstützend zur Seite zu stehen.

Manager:in

Manager:innen leiten und überwachen das Team und sind verantwortlich für das Erreichen der Ziele. (Engel 2011)

„Big Boss“

Als Big Boss werden die Verantwortungstragenden des Gesamtprojektes bezeichnet, was im Großteil der Fälle die Geschäftsführung ist. Ihre Aufgabe liegt darin, die grundlegende Richtung vorzugeben und von Zeit zu Zeit die Erreichung der Ziele zu kontrollieren. (Engel 2011)

Tester:in

Die Rolle der Tester:innen soll den Kund:innen bei der Erstellung von Testfällen unterstützen, als auch dem Entwicklungsteam in puncto Testtechnologien zur Seite stehen.

Des Weiteren finden sich in der Literatur teilweise auch die Rollen Consultant sowie Terminmanager:in oder Tracker:in. Letztere müssen jedoch nicht zwangsläufig durch eine dezidierte Person besetzt sein, sondern können beispielsweise von einer anderen Rolle miterfüllt werden. Die Rolle der Consultants kann bei Bedarf das Team vor allem bei technologischen Herausforderungen unterstützen.

3.2.4 Prozess

Der Ablauf bei Extreme Programming setzt sich aus sechs verschiedenen Phasen zusammen. (Engel 2011)

In der ersten Phase, der Explorationsphase, werden alle Beteiligten mit der Methodik vertraut gemacht. Dies ist vor allem auch bei den Kund:innen wichtig, da von ihnen

erwartet wird, im Optimalfall ständig erreichbar zu sein. Die Entwickler:innen können sich bereits mit den geforderten Technologien beschäftigen und Architekturvorschläge machen. Die Explorationsphase kann einige Woche oder auch wenige Monate dauern.

In der zweiten Phase, der Planungsphase, findet das sogenannte „Planning Game“ statt, welches ein bis maximal zwei Tage dauert. Hier definieren die Kund:innen die Anforderungen (auch User Stories genannt) und schreiben sie auf Story Cards. Diese werden anschließend von den Entwickler:innen hinsichtlich ihres Aufwands geschätzt. Daraufhin legen die Kund:innen die Priorität der User Stories fest. Die Reihenfolge, in der diese abgearbeitet werden sollen, ergibt sich aus der Priorität in Zusammenschau mit der Aufwandsschätzung. Sollten sich im Laufe des Projektes die Anforderungen maßgeblich ändern, kann das Planning Game wiederholt werden.

Als Nächstes erfolgt die Entwicklungsphase. Diese startet direkt mit dem Erstellen von Unit Tests. Dies ist die Vorrausetzung, dass im Nachhinein überprüft werden kann, ob die geforderte Funktionalität erfüllt ist. Diese Praktik nennt sich Test-Driven-Development. Anschließend wird in kurzen Iterationen entwickelt, die meistens ca. eine Woche bis maximal ein Monat andauern. Es steht den Kund:innen jedoch auch während einer Iteration jederzeit frei, die Anforderungen zu ändern bzw. zu adaptieren. Die Iterations-Zyklen können sich je nach Bedarf wiederholen und werden jeweils durch Akzeptanztests der Kund:innen abgeschlossen.

In der vierten Phase, der Freigabephase, wird das Produkt formal von Seiten der Kund:innen nochmal getestet und überprüft, und nach Freigabe in den Betrieb übernommen.

Es folgt die Wartungsphase im Betrieb, wo weitere Fehler behoben werden bzw. Funktionalitäten angepasst werden können.

Erst in den letzten Phase, der Endphase, erfolgt die Dokumentation der Software durch die Entwickler:innen. Diese soll die wichtigsten Informationen enthalten, jedoch dennoch einfach gehalten werden.

3.2.5 Praktiken

Das Vorgehensmodell weist eine Reihe Praktiken auf, die verwendet werden können, jedoch nicht zwangsläufig verwendet werden müssen. Im Zuge dieser Arbeit werden nun drei für Extreme Programming markante Praktiken beispielhaft beschrieben. (Engel 2011)

Pair Programming:

Beim Pair Programming sitzen zwei Entwickler:innen an einem Arbeitsplatz und arbeiten gemeinsam am selben Quellcode. Dabei schreibt immer ein Entwickler bzw. eine Entwicklerin am Code, der bzw. die andere kontrolliert und denkt währenddessen über die nächsten Schritte nach. Die Rollen werden in regelmäßigen Abständen gewechselt.

Ziel davon ist neben der schnelleren Entwicklung auch der Erfahrungsaustausch unter den Kolleg:innen. Die Teams beim Pair Programming werden außerdem regelmäßig durchmischt.

40-Stunden-Woche:

Es wird empfohlen, dass die Entwickler:innen keine Überstunden machen, um jederzeit ausgeruht ihrer Arbeit nachgehen zu können und die Motivation aufrecht erhalten bleibt. Falls nicht anders möglich, sollen die Überstunden auf ein Minimum reduziert werden und sich nicht auf einen längeren Zeitraum erstrecken.

Refactoring:

Das Überarbeiten von bestehendem Code in einfachere und strukturiertere Form unterstreicht den Wert der Einfachheit, der beim Extreme Programming groß geschrieben wird.

3.3 Kanban

3.3.1 Entstehung

Die Kanban Methode kommt ursprünglich aus der Automobilproduktion und wurde erst einige Zeit später auch in der Softwareentwicklung angewandt. Entwickelt wurde sie im Jahre 1947 von Taiichi Ohno bei Toyota. Ziel war es, den Fluss durch die gesamte Produktion zu optimieren, und Lagerbestände gering zu halten. Er erarbeitete eine Methode, bei der anhand von Karten der Materialbedarf eines Prozessschrittes signalisiert wird. Es soll immer nur dann Material vom vorherigen Prozessschritt angefordert werden, wenn es auch gleich sofort verarbeitet werden kann und die notwendigen Kapazitäten vorhanden sind. Die Methodik bedient sich des sogenannten Pull-Prinzips: Ein Prozessschritt „holt“ sich sozusagen die Materialien vom vorherigen Schritt. Kanban bedeutet aus dem Japanischen übersetzt „Signalkarte“. (Krach 2012) In Abbilung 2 wird Kanban in der Produktion mittels Pull-Prinzip dargestellt. (Pfeffer 2019)

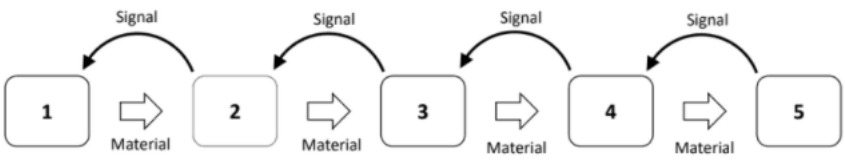


Abbildung 4: Kanban in der Produktion (Pfeffer 2019)

Anfang der 2000er Jahre fand Kanban auch in der Softwareentwicklung Anwendung. Als Begründer von Kanban in der Softwareentwicklung gilt David Anderson, welcher die Methode in diesem Kontext erstmals im Jahr 2007 präsentierte. (Krach 2012)

3.3.2 Grundgedanke

Das Kanban Modell soll bei der Visualisierung der Arbeitsschritte und deren Auslastung dienlich sein. Aus diesem Grund wird ein sogenanntes Kanban-Board angewendet, auf welchem transparent die einzelnen Aufgaben inklusive ihrem Status abgebildet werden. Es soll erst mit einem Arbeitsschritt begonnen werden, wenn dafür auch eine entsprechende Kanban Karte bereitsteht. Die Anzahl an Kanban Karten ist beschränkt und muss mit der verfügbaren Kapazität im Einklang sein – bei einer Anzahl von X Programmierer:innen ist beispielsweise eine Beschränkung von X Karten im Status „In Progress“ sinnvoll. (Cron 2013)

Des Weiteren sollen Prozessschritte klar abgegrenzt sein. Dies kann durch sogenannte „Definition of Ready“ bzw. „Definition of Done“ erreicht werden, wo festgelegt wird, nach welchen Kriterien mit einer Aufgabe gestartet werden kann, bzw. ab wann sie als beendet gilt.

3.3.3 Charakteristika

Die Kanban Methodik zeichnet sich durch folgende vier Eigenschaften aus: (Krach 2012)

Pull-Prinzip

Das Pull-Prinzip besagt, wie bereits oben beschrieben, dass eine Aufgabe aktiv in einen Prozessschritt hineingezogen werden muss und nicht dorthin geschoben werden kann. Dadurch wird sichergestellt, dass die Kapazitäten im jeweiligen Prozessschritt nicht überlastet werden können.

Mengen limitieren

Es dürfen immer nur eine limitierte Anzahl an Kanban Karten im Umlauf sein. Diese Anzahl muss für jedes Projekt anfänglich evaluiert werden und basiert auf den jeweiligen Kapazitäten. Somit gibt es eine Obergrenze an Aufgaben, an denen gleichzeitig gearbeitet werden kann.

Informationen veröffentlichen

Durch die transparente Visualisierung mittels Kanban Board, ist es jedem Teammitglied jederzeit möglich, den Status der einzelnen Aufgaben einzusehen, sowie wer aktuell an welcher Aufgabe arbeitet. Diese Transparenz ist notwendig, um dem Team eine eigenständige und eigenverantwortliche Vorgehensweise zu ermöglichen.

Aus diesem Grund werden auch tägliche Status Meetings, sogenannte „Daily Stand-ups“ durchgeführt, wie sie auch in anderen agilen Vorgehensmodellen wie zum Beispiel Scrum üblich sind.

Kontinuierliche Verbesserung

Prozesse sollen kontinuierlich re-evaluiert und verbessert werden. Bei der Einführung von Kanban wird nicht mit der Neuschaffung von Prozessen begonnen, stattdessen werden bestehende Prozesse herangezogen und laufend optimiert. Aus diesem Grund sollen auch regelmäßig Retrospektiven durchgeführt werden, bei denen rückblickend ein gewisser Zeitabschnitt betrachtet und auf Verbesserungspotenzial geprüft werden soll.

Um kontinuierliche Verbesserung sicherstellen zu können ist es wichtig, Methoden zur Messung und Steuerung zu entwickeln, um Verbesserung auch quantifizierbar zu machen.

3.3.4 Rollen

Anders als bei anderen Vorgehensmodellen werden bei Kanban keine Rollen definiert. Nach David Anderson soll sich Kanban den aktuellen Gegebenheiten anpassen, wonach Rollen individuell definiert werden können. Die Kanban Methodik konzentriert sich vorrangig auf die Erfüllung der in Abschnitt 3.3.3 beschriebenen Eigenschaften, ohne dabei bestehende Prozesse und Strukturen abändern zu müssen. (Cron 2013)

3.3.5 Artefakte

Kanban Board

Das Kanban Board ist das zentrale Instrument in der Kanban Methode, das zur Visualisierung des Arbeitsflusses dient. Es besteht aus mehreren Spalten, in denen sich die jeweiligen Kanban Karten befinden. Für jede Spalte ist ein Maximum an Kanban Karten definiert, das nicht überschritten werden darf.

Die Spalten, die in der Softwareentwicklung zumeist verwendet werden, sind „Backlog“, „To-Do“, „In Progress“ (oder „In Development“), „In Testing“ und „Done“. Dies kann jedoch auch von Projekt zu Projekt variieren bzw. abgeändert werden.

Neben der Verwendung von physischen Kanban Boards, auf denen Kanban Karten in Form von Post-it’s o.ä. abgebildet werden, gibt es heutzutage auch viele Softwareprodukte, die die Verwendung von digitalen Kanban Boards unterstützen.

Durch die visuelle Darstellung verschafft das Kanban Board einen guten Überblick über die Arbeit. Darüber hinaus zeigt es die Engpässe des Projektes auf: Häufen sich in einer Spalte die Karten bis ans Limit, ohne vom nächsten Prozessschritt entgegen genommen werden zu können, zeigt dies die Schwachstellen bzw. Bottlenecks des Prozesses auf.

Kanban Karte

Eine Kanban Karte enthält folgende Informationen (Cron 2013):

- Die Aufgabennummer, welche die Aufgabe eindeutig identifiziert
- Den Titel der Aufgabe
- Die Beschreibung der Aufgabe
- Das Eintrittsdatum, welches den Eintritt in das Kanban Board zeigt
- Das Austrittsdatum, welches den Abschluss der Aufgabe zeigt
- Die Deadline, bis wann die Aufgabe spätestens erledigt sein soll
- Auftraggeber:in oder Autor:in, sodass Rückfragen gestellt werden können
- Bearbeiter:in, welche an der Aufgabe arbeitet
- Ev. weitere Informationen über den Umfang der Aufgabe

3.4 Crystal Family

3.4.1 Entstehung und Konzept

Das Konzept der Crystal Family wurde im Jahre 1992 von Alistor Cockburn entwickelt. Es handelt sich hierbei nicht um eine einzelnes Vorgehensmodell, sondern, wie der Name bereits vermuten lässt, um eine Gruppe an Modellen. Je nach Anwendungsfall bzw. nach Art des Projektes ist auszuwählen, welches Modell sich im konkreten Fall am besten eignet. (Klinger 2010)

Die Crystal Modelle setzen sich aus zwei Dimensionen zusammen. Die erste Dimension ist die Farbe. Sie gibt an, für wie viele Projektteammitglieder die Methode gedacht ist. Je nach Teamgröße lässt sich damit das richtige Modell auswählen. Es gibt die Modelle Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Orange Web, Crystal Red, Crystal Maroon, Crystal Blue, Crystal Violet, Crystal Diamond und Crystal Sapphire. (Kumar, Matche und Maheshwary 2019)

Es wurden bisher jedoch nur die Varianten Crystal Clear, Crystal Orange und Crystal Orange Web näher definiert, weshalb auf diese im Weiteren näher eingegangen wird. (Engel 2011)

Die zweite Dimension, durch welche die Methoden kategorisiert werden können, bezieht sich auf die „Härte“. Das sprechende Bild der Farbe und Härte (von Kristallen) lässt ein gutes Verständnis der beschriebenen Systematik zu.

Die Härte beschreibt die Kritikalität des Projektes. Hier gibt es vier Stufen:

- C (Comfort) beschreibt unkritische Projekte, deren Scheitern lediglich Einbußen beim Komfort der Benutzer:innen mit sich tragen würde.
- D (Discretionary Money): Hierzu zählen Projekte, deren Scheitern den Verlust von Geld bedeuten.

- E (Essential Money): Hiermit ist ebenfalls ein Verlust von Geld verbunden, wobei es sich um deutlich höhere Geldsummen handelt.
- L (Life) würde bei Scheitern des Projektes eine Gefahr für Menschenleben bedeuten. Konkret können hierzu Softwareprojekte für medizinische Einrichtungen bzw. -Geräte, oder auch Softwarelösungen in Flugzeugen gezählt werden. (Engel 2011)

Die einzelnen Modelle der Crystal Family setzen sich jeweils aus einer Farbe (bzw. der Anzahl an Personen des Projektteams) und einer Härte zusammen. Beispielsweise steht D6 aus der Familie Crystal Clear für ein Projekt mit durchschnittlich 6 Projektmitgliedern in einem Projekt mit der Kritikalität D. Sofern nicht eindeutig entschieden werden kann, welche Methode auf das jeweilige Projekt am besten passt, wird empfohlen, im Zweifelsfall die kleinere zu wählen. (Engel 2011)

3.4.2 Grundgedanke

Grundsätzlich steht in der Methodenfamilie Crystal der Mensch im Mittelpunkt. Prozesse werden eher als zweitrangig gesehen, was auch den Umstand mit sich bringt, dass die Methode hinsichtlich Prozessen eine gewisse Flexibilität bietet. Der Fokus liegt auf der Team-Zusammenarbeit, der Kommunikation und der Förderung von Talenten.

Die Crystal Methodik verfolgt zwei Ziele: das Liefern einer funktionsfähigen Software, und den Lerneffekt für zukünftige Projekte. Die Teams sollen sich dabei möglichst selbst organisieren, auf übermäßige Kontrolle soll verzichtet werden. (Hruschka, Rupp und Starke 2004)

3.4.3 Charakteristika

Die Dauer der Entwicklungszyklen bzw. einer Iteration in den Methoden der Crystal Family kann zwischen einer Woche und zwei Monaten variieren. Eine Besonderheit stellt hierbei zusätzlich der Begriff der Episode dar, welche die eigentliche Programmierarbeit innerhalb einer Iteration beschreibt.

Die restlichen Abläufe und Prozesse innerhalb einer Iteration können in der Crystal Family sehr flexibel gestaltet werden. Fixe Bestandteile sind jedoch die Planung am Beginn, tägliche Stand-Up Meetings, sowie ein Abschlussritual am Ende der Iteration. Das Abschlussritual soll vor allem den Teamgeist stärken, es kann sich dabei um eine kleine gemeinsame Feier oder Ähnliches handeln, oder auch um Reflexionsworkshops.

Durch die flexible Gestaltung der Methode kann die Crystal Family gut mit Praktiken aus anderen agilen Vorgehensmodellen wie Scrum oder Extreme Programming kombiniert werden.

Beachtenswert bei der Crystal Family ist, dass ein äußerst hoher Wert auf Dokumentation gelegt wird. Dies geschieht vor dem Hintergrund, dass diese für zukünftige Projekte hilfreich sein kann. (Klinger 2010)

3.4.4 Rollen

Je nach Modell bzw. Projektgröße werden unterschiedliche Rollen definiert. Im Folgenden werden einige der üblichen Rollen erläutert. Diese gibt es in allen Crystal Modellen, wobei sich bei „ansteigender Farbe“ des Modells auch die Anzahl der vorhandenen Rollen erhöht.

Die folgenden vier Rollen müssen mindestens in jedem Crystal Projekt besetzt sein: (Hollenstein und Rutz 2003)

Sponsor:in bzw. Auftraggeber:in:

Sponsor:innen geben das Projekt in Auftrag und finanzieren es. Sie sind außerdem maßgeblich für wirtschaftliche Entscheidungen verantwortlich. (Engel 2011)

Designer:in / Programmierer:in:

Designer:innen und Programmierer:innen sind für die eigentliche Entwicklung der Software verantwortlich. Alistair Cockburn unterscheidet hier zwischen drei Entwicklungsstufen der jeweiligen Positionen, auf die im Zuge dieser Art nicht näher eingegangen werden soll. (Cockburn 2003)

Leitende Designer:innen / Programmierer:innen:

Die leitenden Designer:innen und Programmierer:innen erfüllen eine Schlüsselposition im Projekt, da diese Personen über die meiste Erfahrung verfügen und außerdem auch für die Koordination und Führung der anderen Designer:innen und Programmierer:innen verantwortlich sind. (Engel 2011)

Anwender:in:

Auch den Anwender:innen kommt eine bedeutende Rolle zu, da diese mit den notwendigen Funktionalitäten am besten vertraut sind. Es wird empfohlen, dass sie als Vertreter:innen der Kund:innen ebenfalls vor Ort sind. (Klinger 2010)

Zusätzlich zu den vier genannten Rollen gibt es weitere vier Rollen, die in keiner Variante der Crystal Familie fehlen dürfen. Diese müssen jedoch nicht Vollzeit besetzt sein, sondern können von Teammitgliedern zusätzlich ausgeführt werden. (Engel 2011)

Koordinator:in:

Die Koordinator:innen sind für die Organisation von Meetings sowie für das Projektreporting verantwortlich.

Fachexpert:in:

Die Fachexpert:innen geben dem Team nähere Einblicke in die fachlichen Hintergründe der zu entwickelnden Software.

Tester:in:

Die Test:innen können als vollbeschäftige Personen zur Verfügung stehen, vor allem bei kleinen Teams wird diese Rolle jedoch von den Entwickler:innen mitgemacht.

Autor:in:

Der Rolle der Autor:innen (bzw. auch Schreiber:innen genannt) kommt eine besondere Bedeutung zu: Anders als in anderen agilen Vorgehensmodellen spielt die Dokumentation in der Crystal Methodologie eine sehr große Rolle, nicht zuletzt um das Ziel verfolgen zu können, für zukünftige Projekte entsprechend vorbereitet zu sein.

4. Auswahlkriterien zur Wahl von Vorgehensmodellen

Aus den oben beschriebenen Modellen ergeben sich eine Reihe von Aspekten bzw. Kriterien, welche sich zur Auswahl eines passenden Vorgehensmodells eignen könnten. Im folgenden Abschnitt werden nun diese Aspekte angeführt, deren Anwendung schließlich in Teil 2 der Arbeit näher betrachtet wird.

4.1 Mögliche Aspekte

4.1.1 Teamgröße bzw. Projektgröße

Die Teamgröße stellt einen wesentlichen Faktor dar, der für die Auswahl des Vorgehensmodells entscheidend ist. Wie aus Kapitel 3 deutlich wird, eignen sich manche Vorgehensmodelle besser für kleinere Teams, wohingegen andere ebenso in größeren Teams bzw. Projekten angewandt werden können. Die Frage der Teamgröße bzw. Anzahl der Teams sollte also bei der Wahl eines geeigneten Vorgehensmodells berücksichtigt werden.

4.1.2 Skalierbarkeit

Bei der Skalierbarkeit geht es darum, ob ein Projekt in dem gewählten Vorgehensmodell wachsen kann oder nicht. Es stellt sich somit eine ähnliche Problemstellung wie bei der Teamgröße dar: Wenn ein Vorgehensmodell nur auf kleine Teams angewandt werden kann, kann das Vorgehensmodell nicht oder nur schwer beibehalten werden, wenn das Projekt größere Dimensionen annimmt. Die Skalierbarkeit bezieht sich jedoch nicht nur auf die Größe von Teams, sondern kann auch in Bezug auf den Umfang des Projektes gesehen werden.

4.1.3 Verfügbarkeit der Kund:innen

Wie bereits im agilen Manifest ersichtlich, spielt die Zusammenarbeit mit Kund:innen bei agilen Vorgehensmodellen eine große Rolle. Die Rollendefinition bzw. die Aktivitäten der Kund:innen sind jedoch von Modell zu Modell unterschiedlich. Beispielsweise sind die Kund:innen bei Extreme Programming ein aktiver Teil des Teams und sollten daher immer präsent sein. Im Gegensatz dazu spielen Kund:innen zwar auch bei Scrum eine wichtige Rolle, allerdings werden diese hier weniger gefordert, da die Priorisierungen der einzelnen Anforderungen durch die Product Owner:innen erfolgen. Wenn Product Owner:in und Kund:in gut abgestimmt sind, ist die Anwesenheit der Kund:innen daher nicht so stark erforderlich.

4.1.4 Änderungen der Anforderungen / Grad der Flexibilität

Dieses Kriterium beschreibt, wie genau die Anforderungen am Anfang einer Iteration bekannt sind und ob sich diese im Laufe der Iteration verändern können. Je kürzer eine

Iteration andauert, desto öfter besteht auch die Möglichkeit, neue Anforderungen einzubringen bzw. bestehende Anforderungen abzuändern. Wenn in längeren Iterationen geplant wird, verringert sich diese Flexibilität. Bei Projekten, bei denen der Umfang an Anforderungen am Anfang des Projektes noch sehr unklar sind, muss das Vorgehensmodell entsprechend gewählt werden.

4.1.5 Interdisziplinarität der Teams

Die Interdisziplinarität beschreibt, welche unterschiedlichen Fähigkeiten im Team vorhanden sind. In vielen Aufbauorganisationen von Unternehmen werden die Teams nach Fähigkeiten aufgegliedert: ein Team aus Entwickler:innen, ein Team aus Designer:innen, ein Team aus Softwarearchitekt:innen, usw. In agilen Projekten kann es jedoch sinnvoller sein, alle notwendigen Fähigkeiten in einem Team zu kombiniert, um somit effektiver zu arbeiten. Da dies nicht in allen Unternehmen ohne weiteres möglich ist, muss vor der Wahl des Vorgehensmodells evaluiert werden, wie die Teams zusammengesetzt werden können, und welches Vorgehensmodell sich daher am besten eignet.

4.1.6 Qualifikation der Teammitglieder

Selbstständigkeit der Teammitglieder ist in agilen Projekten ein wichtiges Erfolgskriterium. Allerdings stehen nicht in allen Unternehmen gleich gut qualifizierte Mitarbeiter:innen zur Verfügung. Es stellt sich somit die Frage, ob beispielsweise alle Entwickler:innen eigenständig arbeiten können, oder ob ein leitende Entwickler:innen zur Führung notwendig sind. In der Methodik der Crystal Family sind leitende Entwickler:innen vorgesehen, bei anderen Modellen nicht.

Ein weiterer Aspekt der Qualifikation der Mitarbeiter:innen betrifft auch Kenntnisse über das Vorgehensmodell selbst. So ist etwa bei Scrum das Wissen über die Scrum Methodologien für die Scrum Master:innen äußerst wichtig.

4.1.7 Wichtigkeit von Dokumentation

Die Wichtigkeit der Dokumentation variiert mitunter stark zwischen den einzelnen Vorgehensmodellen. Während die Crystal Family großen Wert auf Dokumentation legt, wird dieser Bereich bei Extreme Programming weniger intensiv behandelt. Dieser Aspekt sollte in die Auswahl eines passenden Vorgehensmodells unbedingt einfließen, um die Vorgaben bzw. Bedürfnisse des jeweiligen Unternehmens zu berücksichtigen.

4.1.8 Natur der Anforderung

Die Art des Softwareprojektes kann auch Einfluss auf die Eignung des Vorgehensmodells haben. Ist das Projekt beispielsweise höchst sicherheitsrelevant (Software zur Flugzeugsteuerung, Software für medizinische Geräte), so sind ausführliche Dokumentation und mehrere Kontrollinstanzen zwingend notwendig, um die

Qualität der Software zu gewährleisten. In diesen Fällen ist sorgsam abzuwägen, ob und welches agile Vorgehensmodell angewandt werden kann, um diese Anforderungen zu erfüllen.

4.1.9 Autonomie der Teams

Die meisten agilen Vorgehensmodelle sehen vor, dass Teams selbstbestimmt und unabhängig ihre Arbeit planen und umsetzen können. Um dies zu erreichen, kann Interdisziplinarität der Teams von Vorteil sein. In der Praxis ist es jedoch des Öfteren so, dass eine gewisse Abhängigkeit von und zu anderen Teams unvermeidbar ist. Ist dies der Fall, muss evaluiert werden, ob die Vorgehensmodelle miteinander kompatibel sind. Wenn ein Team beispielsweise strikt nach Wasserfallmodell vorgeht, ein anderes jedoch nach einem agilen Modell, kann dies zu Konflikten führen. Aber auch innerhalb der agilen Modelle muss abgewogen werden: Wenn ein Team etwa in 2-monatigen Iterationen, das andere jedoch in 3-wöchigen Iterationen plant, können voneinander abhängige Aktivitäten schwerer eingetaktet werden.

4.1.10 Funktion der Teams

Wie aus Abschnitt 2.1 ersichtlich ist, haben Teams je nach Aufgabenbereich mit sehr unterschiedlichen Herausforderungen zu kämpfen. Während die Softwareentwicklung stark projektgetrieben ist, ist der IT Betrieb wiederum mit vielen ad hoc Anforderungen konfrontiert. Außerdem kann es einen großen Unterschied machen, ob an einer neuen Entwicklung gearbeitet wird oder an der Weiterentwicklung eines bestehenden Produkts. Die Frage, welche Funktion das Team im Unternehmen erfüllen soll, muss also vorab beantwortet sein, bevor ein passendes Vorgehensmodell gefunden werden kann.

4.1.11 Örtliche Verteilung der Teammitglieder

Kommunikation und Zusammenarbeit stellen in allen agilen Vorgehensmodellen wichtige Themen dar. Diese Bereiche werden jedoch erschwert, wenn Teammitglieder örtlich verteilt sind. In Zeiten der Globalisierung sind Teammitglieder des selben Teams immer öfter auf mehrere Kontinente und in mehreren Zeitzonen verstreut. Dank modernen Kommunikationstechnologien wie Telefon- und Videokonferenzen lassen sich diese Distanzen zwar teilweise überwinden, es muss dennoch beurteilt werden, welches Vorgehensmodell sich in solchen Situationen am besten umsetzen lässt.

4.1.12 Fluktuation der Teammitglieder

Ein Aspekt, der bei der Wahl des Vorgehensmodells ebenso eine Rolle spielen kann, ist die Fluktuation in den Teams. Methoden, die stark auf Teamzusammenhalt ausgerichtet sind, bedürfen dadurch auch einer gewissen Kontinuität der Teammitglieder. Wenn die Teammitglieder oft wechseln bzw. ausgetauscht werden (sei dies nun geplant oder

ungeplant), können manche agile Vorgehensmodelle weniger gut funktionieren als andere.

5. Vorstellung der Anwendungsmethode

Im folgenden Abschnitt soll anhand der in Kapitel 4 beschriebenen Aspekte evaluiert werden, inwieweit sich die vier vorgestellten Vorgehensmodelle für zwei reale Teams mit unterschiedlichen Tätigkeitsfeldern eignen. Jeder der zwölf Aspekte gilt als ein Kriterium.

Sowohl die Gewichtung als auch die Beurteilung der Eignung der jeweiligen Kriterien wurden eigenständig auf Basis von Vorwissen aus den beiden Teams durchgeführt.

Die Methode sieht folgendermaßen aus:

Gewichtung:

Im ersten Schritt wird geprüft, wie wichtig ein Kriterium für das jeweilige Team ist. Wenn ein Kriterium für ein Team irrelevant ist, muss dessen Eignung natürlich auch nicht berücksichtigt werden. Je ausschlaggebender ein Kriterium jedoch ist, desto mehr wird dessen Eignung gewichtet.

Die Gewichtung erfolgt für jedes Kriterium und wird an einer Skala von 0-3 gemessen:

0 = unwichtig, 1 = eher unwichtig, 2 = eher wichtig, 3 = wichtig.

Eignung:

Im zweiten Schritt wird pro Kriterium für jedes Vorgehensmodell bewertet, inwiefern es sich für das jeweilige Team eignet. Dies geschieht durch die Zusammenschau zwischen Vorgehensmodell und den Eigenschaften des Teams.

Die Eignung erfolgt pro Kriterium ebenfalls auf einer Skala von 0-3:

0 = ungeeignet, 1 = eher ungeeignet, 2 = eher geeignet, 3 = geeignet.

Auswertung:

Im letzten Schritt wird die Gewichtung des Kriteriums mit dem Eignungswert des jeweiligen Vorgehensmodells multipliziert. Das Ergebnis wird in einer Tabelle festgehalten. Zur Auswertung werden die Punkte jedes Vorgehensmodells zusammengezählt. Je höher die Punktezahl eines Vorgehensmodells ist, desto geeigneter kann dieses Modell für das betrachtete Team gesehen werden.

Zur besseren Verständlichkeit der Methode wird ein Beispiel anhand eines fiktiven Teams dargestellt:

1. Gewichtung:

Kriterium A: eher unwichtig (1)

Kriterium B: sehr wichtig (3)

2. Eignung:

Kriterium A:

- Scrum: 2
- Extreme Programming: 1
- Kanban: 1
- Crystal Family: 3

Kriterium B:

- Scrum: 2
- Extreme Programming: 2
- Kanban: 1
- Crystal Family: 0

3. Auswertung

Kriterium		**Scrum**	**XP**	**Kanban**	**Crystal**
Kriterium A	Gewichtung	1			
	Eignung	0	1	1	3
	Wert	**0**	**1**	**1**	**3**
Kriterium B	Gewichtung	3			
	Eignung	2	2	1	0
	Wert	**6**	**6**	**3**	**0**
Summe Wert		**6**	**7**	**4**	**3**

Tabelle 1: Beispiel-Auswertung eines fiktiven Teams

Im hier dargestellten Beispiel wäre somit das Vorgehensmodell Extreme Programming das geeignetste Modell für das fiktive Team.

6. Vorstellung des Umfelds

6.1 Das Unternehmen

Die beiden betrachteten Teams arbeiten im selben Unternehmen. Um die Rahmenbedingungen zu verdeutlichen, wird zuerst das Unternehmen näher vorgestellt.

Es handelt sich um einen global vertretenen IT-Konzern mit Hauptsitz in München und mehreren Tochtergesellschaften, die über fünf Kontinente verteilt sind. Das Kernprodukt des Unternehmens ist eine Versicherungssoftware, die die Abwicklung von Versicherungsgeschäften in sämtlichen Versicherungssparten unterstützt. Neben der Entwicklung dieser Software, wird auch dessen Auslieferung und Betrieb innerhalb des Konzerns abgewickelt.

Einer der strategisch wertvollsten Standorte ist Wien, an dem ca. 600 Mitarbeiter:innen beschäftigt sind. An diesem Standort wird die globale Auslieferung der Versicherungssoftware koordiniert und gemanagt, außerdem befindet sich hier auch das globale IT Betriebsteam, welches für Deployment und Betrieb dieser Software endverantwortlich ist.

Die Verfügbarkeit und Funktionalität von Versicherungssoftware ist unentbehrlich für das tägliche Geschäft von Versicherungsunternehmen. Ein Ausfall bzw. eine Fehlfunktion kann unter Umständen einerseits finanziellen Schaden, andererseits auch einen Image-Schaden verursachen.

Das Unternehmen hat sich vor einigen Jahren dazu entschlossen, das Betriebsteam auf mehrere Standorte auszuweiten, weshalb Mitarbeiter:innen aus den Tochtergesellschaften in Spanien und Indien hinzugezogen wurden. Grund dafür sind einerseits niedrigere Lohnkosten, sowie das „Follow-the-sun“-Prinzip, welches den 24/7 Support erleichtert.

6.2 Das Projekt VMP (Versicherungs-Master-Plattform):

Das Projekt, das im Zuge dieser Arbeit näher betrachtet wird, nennt sich Versicherungs-Master-Plattform (VMP). Es handelt sich hierbei um die Migration einer Plattform in die Cloud, in der mehrere Komponenten bzw. IT Services zur Verfügung gestellt werden, die anschließend von den Kund:innen genutzt werden können. Kund:innen sind in diesem Fall verschiedene, weltweit tätige Versicherungsunternehmen.

Bei den genannten IT Services handelt es sich um verschiedenste Anwendungen, die im Versicherungsumfeld eingesetzt werden. Neben einer zentralen Datenbank, in der sämtliche Kunden- und Vertragsdaten gespeichert sind, gehören zu den Services auch diverse Kunden- oder Maklerportale.

Zum aktuellen Zeitpunkt sieht der Plan des Unternehmens eine Bereitstellung von ca. 30 verschiedenen IT Services vor. Die langfristige Strategie des Unternehmens sieht jedoch in den nächsten Jahren Kundenzuwachs vor, der in weiterer Folge auch eine Vergrößerung des Angebots an IT Services bewirken kann. Aus diesem Grund muss eine Skalierung des Projektumfangs jederzeit möglich sein.

Aufgrund der Größe des Projektes erschließt sich auch die Anzahl der Beteiligten: Insgesamt arbeiten rund 2000 Mitarbeiter an dem Projekt. Die einzelnen Teams sind auf mehrere Standorte verteilt und entwickeln dort die jeweiligen Applikationen.

Da das Unternehmen bis vor Kurzem seine IT Services ausschließlich im einem auswärtig betriebenen Rechenzentrum gehostet hatte, liegt eine Besonderheit dieses Projektes in der Verlagerung in die Cloud.

Der Strategiewechsel in Richtung Cloud Computing stellt für das Unternehmen nun die Möglichkeit dar, bisher manuell getätigte Prozesse wie Server-Bereitstellung oder Deployments zu automatisieren. Auf diese Weise können Kunden-Instanzen und auch Software Updates schneller zur Verfügung gestellt werden. Zusätzlich verringert es auf lange Sicht den Personalbedarf.

7. „Team Entwicklung“

7.1 Vorstellung von „Team Entwicklung“

7.1.1 Aufgabenbereiche

Das „Team Entwicklung“ hat die Aufgabe, das Setup der IT Services in der Cloud zu automatisieren.

Um ein IT Service in der Cloud zur Verfügung zu stellen, müssen einige Schritte erfüllt werden. Es müssen zum Beispiel zuerst die virtuellen Maschinen bestellt und konfiguriert, gegebenenfalls Datenbanken aufgesetzt, sämtliche zusätzlich benötigte Software installiert und schließlich die Anwendungssoftware entsprechend installiert und konfiguriert werden, welche dem Endanwender:innen letztendlich zur Verfügung steht.

Dieser Prozess besteht aus sehr vielen einzelnen Schritten, was zum Nachteil hat, dass er einerseits sehr zeitaufwendig und andererseits auch fehleranfällig sein kann. Um diesem Problem vorzubeugen, hat das „Team Entwicklung“ die Aufgabe, diese Prozesse durch Skripte zu automatisieren. Da der Hauptfokus des Teams auf der Deployment-Automatisierung der einzelnen IT Services liegt, wird auch nur dieser Bereich in dieser Arbeit beleuchtet.

Die bereits oben beschriebene Plattform soll am Ende des Projektes aus rund 30 verschiedenen IT Services bestehen, für die das „Team Entwicklung“ die Skripte zur Deployment-Automatisierung schreibt.

Neben der initialen Entwicklung dieser Skripte müssen diese auch regelmäßig aktualisiert bzw. Fehler behoben werden.

7.1.2 Teamzusammensetzung

Das Team besteht aktuell aus 9 Mitarbeiter:innen und setzt sich folgendermaßen zusammen:

- 8 Entwickler:innen mit Schwerpunkt Infrastruktur-Automatisierung
- 1 Testerin

Die Aufteilung der Entwickler:innen auf die unterschiedlichen Standorte sieht folgendermaßen aus: 2 in Österreich, 3 in Rumänien, 2 in Indien, 2 in Deutschland. Die Testerin ist in Österreich.

Gründe für die Standorterweiterung liegen einerseits in den niedrigeren Lohnkosten in Rumänien und Indien, andererseits aber auch darin, dass der Arbeitsmarkt in der IT Branche in Mitteleuropa schon gut ausgeschöpft ist. Diese örtliche Verteilung der Teammitglieder führt allerdings dazu, dass der Teamzusammenhalt nicht so stark ausgeprägt ist, wie es bei Teams der Fall ist, die sich gesamtheitlich am selben Standort

aufhalten. Dies wiederum hat zur Konsequenz, dass eine hohe Fluktuation innerhalb des Teams besteht. Da der Arbeitsmarkt für gut ausgebildete Entwickler:innen sehr vielversprechend ist, sind viele Mitarbeiter:innen nur 1-2 Jahre im Team, bevor sie in ein anderes Unternehmen wechseln.

7.1.3 Herausforderungen und Abhängigkeiten

Eine Herausforderung, mit der das „Team Entwicklung“ konfrontiert ist, ist die Abhängigkeit zum IT Architektur Team. Vor jeder Neuimplementierung eines IT Services muss von den jeweiligen IT Architekten ein Konzept erstellt werden, das die Grundlage für die Automatisierung der Deployments bildet. Das IT Architektur Team arbeitet aktuell nach Scrum in 2-wöchigen Sprints.

Selbige Abhängigkeit gilt natürlich für die Entwicklerteams, die die Anwendungen selbst entwickeln. Voraussetzung für die Automatisierung eines Deployments ist ein gewisser Fertigstellungsgrad der Entwicklung, sowie die Verfügbarkeit aller notwendigen Informationen zur Applikation. Sobald diese Voraussetzungen erfüllt sind, kann das Team die Umsetzung der Anforderungen selbstständig planen, ohne weitere Abhängigkeiten berücksichtigen zu müssen.

Eine weitere Herausforderung stellen kurzfristige ad hoc Anforderungen wie die Behebung von Fehlern in den Skripts dar. Diese sollten zwar zeitnah behoben werden, sind aber in den meisten Fällen nicht betriebskritisch, weil sie durch manuelle Schritte „entschärft“ werden können.

Außerdem müssen die Skripts bei Änderungen der Anwendungen auch regelmäßig adaptiert werden. Da diese Änderungen zuerst von den Entwicklungsteams der Anwendungen selbst geplant und durchgeführt werden müssen, kann auch das „Team Entwicklung“ diese Tätigkeiten rechtzeitig planen.

Eine Schwierigkeit ergibt sich außerdem bei der Definition des „Kunden“ für das „Team Entwicklung“. Da „der Endkunde“ (Versicherungsnehmer:in, Sales Mitarbeiter:in, etc.) nur peripher vom Automatisierungsgrad der Deployments betroffen ist, kann als direkter „Kunde“ des „Team Entwicklung“ das „Team Betrieb“ angesehen werden. Sobald die Deployment-Skripts fertiggestellt sind, werden sie an das „Team Betrieb“ übergeben, welche die Deployments dann ausführt. Die Entwicklung der Skripts kann allerdings sehr selbstständig durchgeführt werden, ohne dass eine Anwesenheit des „Kunden“ erforderlich wäre.

Um die richtige Ausführung der Deployment-Skripts zu gewährleisten, muss das „Team Entwicklung“ seine Arbeit entsprechend dokumentieren, um die Skripts für das „Team Betrieb“ nachvollziehbar zu machen.

7.2 Anwendung der Kriterien auf „Team Entwicklung“

Die in Abschnitt 5 beschriebene Methode wird nun auf das „Team Entwicklung“ angewandt. Im ersten Schritt werden die Kriterium aus Kapitel 4.1 einer Gewichtung unterzogen, um zu überprüfen, ob sie für das „Team Entwicklung“ relevant sind.

7.2.1 Gewichtung

Teamgröße bzw. Projektgröße: 1

Das „Team Entwicklung“ besteht aktuell aus 9 Mitarbeiter:innen. Die Größe des Teams ließe sich jedoch bei Bedarf auch erweitern bzw. gegebenenfalls auch reduzieren, da keine feste Mitarbeiter:innen-Bindung im Team besteht. Da es sich bei dem Projekt um ein sehr großes, langfristiges Projekt handelt, ist auch die Bildung von mehreren, parallel arbeitenden Teams eine Möglichkeit, die relativ leicht umsetzbar wäre. Dem Kriterium der Teamgröße wird daher keine hohe Bedeutung beigemessen und wird mit eher unwichtig (1) bewertet.

Skalierbarkeit: 3

Das Projekt dient dem Aufbau einer IT Plattform, die von verschiedensten Versicherungsunternehmen weltweit verwendet werden soll. Auch wenn der Umfang sich aktuell auf 30 verschiedene IT Services beschränkt, liegt die Vermutung nahe, dass sich der Umfang des Projektes noch deutlich erweitern könnte. Aus diesem Grund wird das Kriterium der Skalierbarkeit mit wichtig (3) eingestuft.

Verfügbarkeit der Kund:innen: 1

Um dieses Kriterium beurteilen zu können, muss zuerst definiert werden, wer die Kund:innen sind. Die Automatisierungen, die von diesem Team entwickelt werden, werden hauptsächlich unternehmensintern eingesetzt und sind für den Endkund:innen weitgehend irrelevant. Ziel ist es, eine Lösung zu implementieren, die sowohl technisch als auch preislich am besten ist. Eine Interaktion mit (externen) Kund:innen ist kaum notwendig, da als Kund:in am ehesten andere interne Teams angesehen werden könnten (z.B. das „Team Betrieb“). Daher wird die Wichtigkeit der Verfügbarkeit der Kund:innen mit eher unwichtig (1) bewertet.

Änderungen der Anforderungen / Grad der Flexibilität: 1

Aufgrund der Größe des Projektes und der vielen unterschiedlichen Stakeholder muss eine gewisse Flexibilität der Anforderungen grundsätzlich möglich sein. Da das Team jedoch die Aktivitäten zur Automatisierung der IT Services großteils selbst planen kann, stehen diese auch meist schon zu Beginn einer Iteration fest. Dieses Kriterium wird daher mit eher unwichtig (1) bewertet.

Interdisziplinarität der Teams: 3

Eine Interdisziplinarität innerhalb des Teams ist aufgrund der Rahmenbedingungen des Unternehmens nur bedingt möglich, was wiederum bei der Wahl des Vorgehensmodells berücksichtigt werden muss. Sowohl die IT Architekt:innen als auch die Entwickler:innen der Anwendungen stellen organisatorisch eigenständige Teams dar. Eine Änderung dieser Struktur ist von der Unternehmensleitung nicht vorgesehen. Da es für manche Vorgehensmodelle jedoch notwendig ist, interdisziplinär aufgestellte Teams zu haben, wird dieses Kriterium mit wichtig (3) bewertet.

Qualifikation der Teammitglieder: 0

Da in dem international vertretenen Unternehmen weltweit auf einen sehr großen Pool an Mitarbeiter:innen zurückgegriffen werden kann und Budgets für Schulungen im Bedarfsfall vorhanden sind, muss diesem Kriterium keine allzu große Bedeutung zugestanden werden, und wird daher mit unwichtig (0) bewertet.

Wichtigkeit der Dokumentation: 2

Da die Dokumentation für die Endkund:innen zwar nicht relevant, jedoch für das „Team Betrieb“ essentiell ist, wird die Wichtigkeit der Dokumentation mit eher wichtig (2) bewertet.

Natur der Anforderung: 0

Bei den Aufgaben des Teams handelt es sich vorrangig um Automatisierungen, deren zeitnahe Umsetzung nicht existenzkritisch für das Unternehmen sind, weil sich nachgelagert noch das „Team Betrieb“ um Incidents und dergleichen kümmert. Daher wird dieses Kriterium mit einer Gewichtung von 0 bewertet.

Autonomie der Teams: 3

Das Team ist teilweise von anderen Teams abhängig, vor allem vom Team der IT Architektur, aber auch von anderen Teams, die die verschiedenen IT Services entwickeln. Aus diesem Grund wird die Gewichtung dieses Kriteriums mit wichtig (3) bewertet.

Funktion der Teams: 1

Die Funktion des Teams besteht in der Entwicklung von Automatisierungen, welche gut planbar sind. Da das Team mit wenigen ad hoc Anforderungen konfrontiert ist (mit Ausnahme von Fehlerbehebungen in den Automatisierungen) fällt diesem Kriterium eine niedrigere Gewichtung zu und wird mit eher unwichtig (1) bewertet.

Örtliche Verteilung der Teammitglieder: 3

Da die Mitglieder des Teams auf mehrere Länder und Kontinente verteilt sind, muss ein Vorgehensmodell gefunden werden, das dies berücksichtigt. Die Gewichtung wird daher mit wichtig (3) bewertet.

Fluktuation der Teammitglieder: 3

Die Fluktuation innerhalb des Teams ist relativ hoch, daher wird dieses Kriterium ebenfalls mit einer Gewichtung von 3 herangezogen. Das gewählte Vorgehensmodell muss schließlich trotz der hohen Fluktuation funktionieren.

7.2.2 Eignung der Vorgehensmodelle

Nun wird jedes Vorgehensmodell anhand jeden Kriteriums betrachtet, um zu prüfen ob es für das „Team Entwicklung“ geeignet ist.

Teamgröße bzw. Projektgröße

- Scrum: 3
  Scrum eignet sich für Teams aus ca. 5 bis 10 Personen und ist daher in diesem Fall sehr gut geeignet. Die Größe des Projektes wird in Scrum nicht beschränkt.
- Extreme Programming: 2
  XP eignet sich zwar für ein Team aus 9 Personen, allerdings ist das Modell nicht auf große Projektstrukturen mit vielen Teams ausgelegt, daher ist es nur „eher geeignet“.
- Kanban: 3
  Kanban kann unabhängig von der Team- bzw. Projektgröße angewandt werden, und ist daher gut geeignet.
- Crystal Family: 1
  Da die “größeren” Modelle der Crystal Family noch nicht näher definiert wurden, eignet sich Crystal vorrangig für Projekte mit bis zu 40 Personen. Aufgrund der Größe des gesamten Projektes wird die Methode für das „Team Entwicklung“ mit eher ungeeignet bewertet.

Skalierbarkeit:

- Scrum: 3
  Scrum lässt sich beispielsweise auf das LeSS oder das SAFe Framework ausweiten und ist daher gut skalierbar.
- Extreme Programming: 0
  XP eignet sich nur für kleine bis mittelgroße Teams. (Klinger, 2010)
- Kanban: 3
  Kanban kann bei Anwachsen des Projektes noch gleichermaßen angewandt werden und ist daher sehr gut geeignet.
- Crystal Family: 1
  Wenn das Projekt skaliert, müsste auch die Crystal Methode (innerhalb der Crystal Family) gewechselt werden, was eine gewisse Komplexität mit sich bringt

und (wie oben beschrieben) nur beschränkt möglich ist. Aus diesem Grund ist die Methode eher ungeeignet.

Verfügbarkeit der Kund:innen:

- Scrum: 3
  Da Scrum keinen unmittelbaren Kund:innenkontakt erfordert, ist die Methode trotz mangelhafter Kund:innenverfügbarkeit gut geeignet.
- Extreme Programming: 0
  Da keine Kund:innen zur Verfügung stehen, ist XP nicht geeignet, weil die Kund:innen Teil des Teams sein sollten.
- Kanban: 3
  Kanban erfordert keine Verfügbarkeit der Kund:innen und ist daher gut geeignet.
- Crystal Family: 1
  Die Rolle der Kund:innen wird bei Crystal Family durch die Anwender:innen vertreten. Dies wären im Falle des „Team Entwicklung“ das „Team Betrieb“. Da dieses auch nur bedingt verfügbar ist, ist die Methode eher ungeeignet.

Änderungen der Anforderungen / Grad der Flexibilität:

- Scrum: 2
  Bei Scrum können Anforderungen am Beginn jedes Sprints geändert werden, jedoch nicht während eines Sprints. Da auch hin und wieder ad hoc Anforderungen (Fehlerbehebungen) zu erledigen sind, ist die Methode nur eher geeignet.
- Extreme Programming: 3
  Bei XP können die Anforderungen auch während eines Entwicklungszyklus noch geändert werden. Die Methode ist daher sehr flexibel und in dieser Hinsicht sehr gut geeignet.
- Kanban: 3
  Im Kanban Modell kann auf geänderte Anforderungen sofort reagiert werden, es ist daher sehr gut geeignet.
- Crystal Family: 2
  Ein Entwicklungszyklus dauert hier zwischen einer Woche und zwei Monaten, Änderungen in diesem Zeitraum sind nicht vorgesehen. Bei der Wahl eines kurzen Entwicklungszyklus ist die Methode (genau wie Scrum) eher geeignet.

Interdisziplinarität der Teams:

- Scrum: 2
  Bei Scrum ist Interdisziplinarität nicht zwingend erforderlich aber wünschenswert. Idealerweise sollten also auch die IT Architekt:innen Teil des Teams sein. Die Methode ist daher nur eher geeignet.

- Extreme Programming: 2
  Die erforderlichen (technischen) Rollen, die bei XP erforderlich sind, finden sich grundsätzlich auch im „Team Entwicklung“. Idealerweise sollten jedoch auch in diesem Modell die IT Architekt:innen Teil des Teams sein.
- Kanban: 3
  Über die Interdisziplinarität der Teams wird in Kanban keine Aussage getroffen, die Teamstruktur des „Team Entwicklung“ ist für Kanban gut geeignet.
- Crystal Family: 2
  Auch hier ist eine Interdisziplinarität des Teams von Vorteil, weshalb die Crystal Methode ebenfalls nur eher geeignet ist.

Qualifikation der Teammitglieder:

Da dieses Kriterium mit einer Gewichtung von 0 bewertet wurde, müssen die einzelnen Vorgehensmodelle in dieser Hinsicht nicht bewertet werden.

Wichtigkeit von Dokumentation:

- Scrum: 3
  Scrum macht keine Vorgaben hinsichtlich Dokumentation, die Dokumentation richtet sich nach dem Bedarf des Projektes. Die Methode ist in dieser Hinsicht daher sehr gut geeignet.
- Extreme Programming: 3
  Dokumentation erfolgt erst in der Endphase und wird einfach gehalten, was sich gut für das „Team Entwicklung“ eignet.
- Kanban: 3
  Bei Kanban werden keine Vorgaben hinsichtlich Dokumentation gemacht, weshalb sich die Methode sehr gut eignet.
- Crystal Family: 1
  In der Crystal Family wird Dokumentation sehr groß geschrieben, was im Anwendungsfalle des „Team Entwicklung“ jedoch nicht erforderlich ist. Eine übermäßige Dokumentation könnte das Team in seiner Produktivität hemmen.

Natur der Anforderung:

Da dieses Kriterium mit einer Gewichtung von 0 bewertet wurde, müssen die einzelnen Vorgehensmodelle in dieser Hinsicht nicht bewertet werden.

Autonomie der Teams:

- Scrum: 2
  Da das IT Architektur Team, zu dem eine Abhängigkeit besteht, ebenso nach Scrum arbeitet, lässt sich dieses Modell trotz Abhängigkeit gut umsetzen. Voraussetzung ist jedoch, dass die beiden Teams eine kompatible Sprintlänge wählen.

- Extreme Programming: 1
  Die (relativ langen) Phasen in Extreme Programming lassen sich schwer mit anderen Vorgehensmodellen in anderen Teams in Einklang bringen.
- Kanban: 1
  Da es bei Kanban keine fest geplanten Zyklen gibt, lassen sich Abhängigkeiten zu und von anderen Teams schwerer einplanen (vor allem aus Sicht des Teams, das die Abhängigkeiten hat). Dies bietet zwar eine hohe Flexibilität für das Team selbst, erschwert aber den davon abhängigen Teams die Planung, weshalb die Methode im Kontext des gesamten Projektes eher ungeeignet ist.
- Crystal Family: 2
  Voraussetzung für eine Zusammenarbeit mit anderen Teams ist, dass die Entwicklungszyklen beidseits abgestimmt werden.

Funktion der Teams:

- Scrum: 3
  Die Automatisierungsaufgaben des Teams lassen sich gut in Sprints einplanen, weshalb die Methode sehr gut geeignet ist.
- Extreme Programming: 3
  Die Phasen in XP lassen sich gut auf die Automatisierungsaufgaben des Teams anwenden, daher ist auch diese Methode sehr gut geeignet.
- Kanban: 3
  Kanban eignet sich sehr gut zur Abarbeitung von Automatisierungsaufgaben, was sich bereits anhand positiver Beispiele aus der Automobilindustrie gezeigt hat.
- Crystal Family: 3
  Automatisierungsaufgaben lassen sich auch in den Zyklen der Crystal Modelle sehr gut abarbeiten.

Örtliche Verteilung der Teammitglieder:

- Scrum: 3
  Die klar strukturieren Zeremonien in Scrum, sowie modere Kollaborationsanwendungen, erleichtern die regelmäßige, wechselseitige Kommunikation. Daher ist das Vorgehensmodell ist für verteilte Teams sehr gut geeignet.
- Extreme Programming: 1
  XP empfiehlt, dass alle Teammitglieder an einem Ort sitzen, was beim „Team Entwicklung“ nicht möglich ist. Da sich die Möglichkeiten für verteiltes Zusammenarbeiten durch Kollaborationsanwendungen jedoch in den letzten Jahren massiv verbessert haben, wird die Methode trotzdem mit einem Punkt (1) bewertet.

- Kanban: 2
  Die örtliche Verteilung der Teammitglieder erschwert die Anwendung der Kanban Methode zwar, verhindert sie jedoch nicht. Eine häufige Abstimmung und Kommunikation der Teammitglieder ist notwendig, damit die Abarbeitung „im Fluss“ bleibt, daher ist die Methode im Falle einer örtlichen Verteilung nur eher geeignet (2).
- Crystal Family: 1
  Da der Fokus der Crystal Family auf Team-Zusammenarbeit und ausgeprägter Kommunikation liegt, eignet sich die Methode besser für Teams, die an einem Ort arbeiten, und wird daher mit eher ungeeignet (1) beurteilt.

Fluktuation der Teammitglieder:

- Scrum: 2
  Die klaren Strukturen und Zeremonien in Scrum stellen sicher, dass die Teammitglieder in gutem Austausch stehen, was vor allem dann wichtig ist, wenn viele neue Mitarbeiter:innen im Team sind. Nichtsdestotrotz erschwert eine hohe Fluktuation die Planbarkeit pro Sprint, daher wird die Methode nur mit 2 Punkten bewertet.
- Extreme Programming: 1
  Der phasenbasierte Aufbau von XP und die kurzen intensiven Entwicklungsphasen funktionieren schlechter bei hoher Fluktuation der Teammitglieder, weshalb das Modell mit eher ungeeignet (1) bewertet wird.
- Kanban: 2
  Da die Methode relativ simpel ist, kann sie auch ohne lange Einschulung einfach von neuen Teammitgliedern angewandt werden. Allerdings erschwert die Fluktuation die Zusammenarbeit und somit den „Fluss“ der Aufgabenabarbeitung.
- Crystal Family: 0
  Da der Fokus der Crystal Family auf Team-Zusammenhalt liegt, sollte Fluktuation innerhalb des Teams möglichst vermieden werden, da dies ein richtiges Ausleben der Crystal Methode verhindert.

7.2.3 Auswertung „Team Entwicklung“

Kriterium		**Scrum**	**XP**	**Kanban**	**Crystal**
Teamgröße bzw. Projektgröße	Gewichtung	1			
	Eignung	3	2	3	1
	Wert	**3**	**2**	**3**	**1**
Skalierbarkeit	Gewichtung	3			
	Eignung	3	0	3	1
	Wert	**9**	**0**	**9**	**3**

Kriterium		Scrum	XP	Kanban	Crystal
Verfügbarkeit der Kund:innen	Gewichtung	1			
	Eignung	3	0	3	1
	Wert	**3**	**0**	**3**	**1**
Änderungen der Anforderungen / Grad der Flexibilität	Gewichtung	1			
	Eignung	2	3	3	2
	Wert	**2**	**3**	**3**	**2**
Interdisziplinarität der Teams	Gewichtung	3			
	Eignung	2	2	3	2
	Wert	**6**	**6**	**9**	**6**
Qualifikation der Teammitglieder	Gewichtung	0			
	Eignung				
	Wert	**0**	**0**	**0**	**0**
Wichtigkeit der Dokumentation	Gewichtung	2			
	Eignung	3	3	3	1
	Wert	**6**	**6**	**6**	**2**
Natur der Anforderung	Gewichtung	0			
	Eignung				
	Wert	**0**	**0**	**0**	**0**
Autonomie der Teams	Gewichtung	3			
	Eignung	2	1	1	2
	Wert	**6**	**3**	**3**	**6**
Funktion der Teams	Gewichtung	1			
	Eignung	3	3	3	3
	Wert	**3**	**3**	**3**	**3**
Örtliche Verteilung der Teammitglieder	Gewichtung	3			
	Eignung	3	1	2	1
	Wert	**9**	**3**	**6**	**3**
Fluktuation der Teammitglieder	Gewichtung	3			
	Eignung	2	1	2	0
	Wert	**6**	**3**	**6**	**0**
Summe Wert		**53**	**29**	**51**	**27**

Tabelle 2: Auswertung für „Team Entwicklung“

Die Auswertung zeigt, dass nach dieser Bewertung Scrum mit der höchsten Punkteanzahl von 53 Punkten am besten geeignet ist, dicht gefolgt von Kanban mit 51 Punkten.

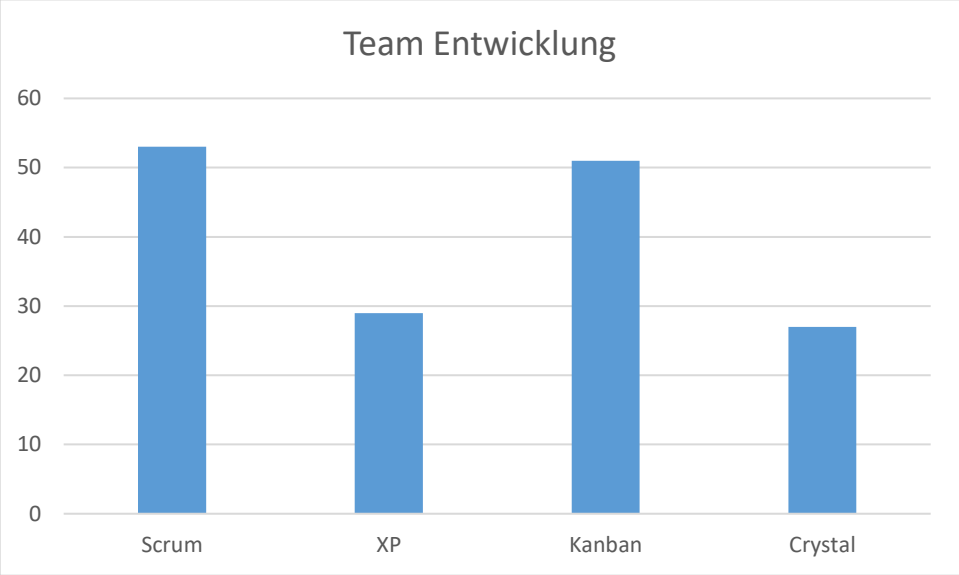


Abbildung 5: Ergebnis für „Team Entwicklung“

8. „Team Betrieb“

8.1 Vorstellung von „Team Betrieb“

8.1.1 Aufgabenbereiche

Wie bereits in Abschnitt 7 beschrieben, beschäftigt sich das „Team Entwicklung“ mit der Erstellung der automatisierten Deployment-Scripts für die einzelnen IT Services. Das „Team Betrieb“ ist für den Betrieb dieser vom „Team Entwicklung“ automatisierten IT Services zuständig. Konkret bedeutet das die Ausführung der Deployments bei den Kunden (auf Grundlage der zuvor entwickelten Skripts), Monitoring der Services, Bearbeitung von Change Requests, sowie Problembehebung bei im Falle von Incidents. Da die Services bei den Kund:innen entsprechend der SLAs rund um die Uhr zur Verfügung stehen müssen, muss auch das „Team Betrieb“ jederzeit erreichbar sein. Aus diesem Grund wurde ein 24/7 Bereitschaftsmodell entwickelt, bei dem die Mitarbeiter:innen des Teams abwechselnd je 24 Stunden auf Abruf sind.

8.1.2 Teamzusammensetzung

Aktuell besteht das Team aus 18 Mitarbeiter:innen, die alle ähnliche technische Fähigkeiten besitzen und dieselben Arbeiten ausführen. Somit kann jedes Teammitglied alle Aufgaben erfüllen, die im Team anfallen, was einen großen Vorteil hinsichtlich Flexibilität mit sich bringt.

8 Teammitglieder sind in Wien, die anderen 10 befinden sich in Indien. Die Aufteilung auf diese beiden Lokationen und die damit verbundenen 4,5 Stunden Zeitverschiebung bieten einen Vorteil bei der Erfüllung des Bereitschaftsmodells („Follow-the-sun“-Prinzip). Im Gegensatz zum „Team Entwicklung“ hat sich beim „Team Betrieb“ in den letzten Jahren eine sehr geringe Fluktuation gezeigt und der Teamspirit ist sehr gut.

8.1.3 Herausforderungen

Eine der größten Herausforderungen des Teams besteht darin, dass es viele ad hoc Anforderungen bewältigen muss. Sowohl die Ausführung eines Deployments, die Durchführung eines Changes und vor allem die Behebung eines Incidents müssen in den meisten Fällen zeitnah erledigt werden, und können daher nicht im Voraus geplant werden. Die Vorlaufzeit für solche Anforderungen beträgt oft nur wenige Tage, bzw. bei betriebsunterbrechenden Incidents auch nur wenige Stunden.

Ein Vorteil, den das „Team Betrieb“ hat ist, dass es relativ autonom und unabhängig von anderen Teams arbeiten kann. Denn erst wenn ein Automatisierungsskript fertiggestellt und alle vorherigen Abhängigkeiten abgelöst wurden, wird das IT Service in Betrieb genommen.

Als Kund:in im Falle des „Team Betrieb“ können einerseits die Endkund:innen angesehen werden, die die Software anwenden. Sie sind typischerweise auch diejenigen, die Incidents melden. Aus einer anderen Perspektive betrachtet kann das Versicherungsunternehmen selbst als Kund:in gesehen werden, mit dem ein Vertrag eingegangen wurde und welches für die Serviceleistungen bezahlt. Kund:innen können daher aus praktischen Gründen nicht Teil des Teams sein und nicht permanent in alle Tätigkeiten eingebunden werden.

8.2 Anwendung der Kriterien auf „Team Betrieb“

Im diesem Abschnitt wird die Bewertungsmethode nun auch auf das „Team Betrieb“ angewandt. Auch hier wird zuerst eine Gewichtung der Kriterien auf das Team vorgenommen.

8.2.1 Gewichtung

Teamgröße bzw. Projektgröße: 2

Das Team ist aktuell mit 18 Mitarbeiter:innen relativ groß. Dies ist notwendig, um den Betrieb für alle Kund:innen sicherstellen zu können. Eine Aufteilung auf mehrere Teams ist zwar grundsätzlich möglich, jedoch muss das dann auch in den jeweiligen Prozessen reflektiert werden und führt zu höherer Komplexität. Aus diesem Grund wird die Gewichtung mit 2 (eher wichtig) bewertet.

Skalierbarkeit: 3

Sowohl die Anzahl der Kund:innen (Versicherungsunternehmen) als auch die Anzahl der angebotenen IT Services kann sich erhöhen, was aus aktueller Sicht auch in der Strategie des Unternehmens vorgesehen ist. Daher muss das Kriterium der Skalierbarkeit in der Bewertung auf jeden Fall berücksichtigt werden und wird als sehr wichtig (3) geachtet.

Verfügbarkeit der Kund:innen: 0

Als Kund:innen aus Sicht des „Team Betrieb“ können einerseits die Endanwender:innen oder auch das Management der jeweiligen Versicherungsunternehmen angesehen werden. In beiden Fällen ist es jedoch nicht erforderlich, dass die Kund:innen für das Team verfügbar sind, da die Tätigkeiten des Teams klar definiert sind, und es im Gegensatz zu Entwicklungsprojekten wenig Interpretationsspielraum bezüglich der Anforderungen gibt. Aus diesem Grund wird die Verfügbarkeit der Kund:innen als unwichtig (0) angesehen.

Änderungen der Anforderungen / Grad der Flexibilität: 3

Der Grad der Flexibilität muss sehr hoch sein, da das Team mit ständig wechselnden Anforderungen konfrontiert ist, die meist nur wenig Vorlaufzeit bzw. Planungsspielraum zulassen. Aus diesem Grund wird dieses Kriterium mit 3 (wichtig) bewertet.

Interdisziplinarität der Teams: 0

Das Team setzt sich aktuell aus Mitarbeiter:innen ähnlicher Fähigkeiten zusammen, die sämtliche Aufgaben des Teams erfüllen können. Anhand der Erfahrungen der letzten Jahre hat sich gezeigt, dass das Team mit dem aktuellen Skillset gut funktioniert und eine weitere Interdisziplinarität daher nicht notwendig ist, weshalb dieses Kriterium mit 0 (unwichtig) bewertet wird.

Qualifikation der Teammitglieder: 0

Die nötige Qualifikation der Teammitglieder ist im „Team Betrieb“ gegeben, weshalb dieses Kriterium keine Herausforderung darstellt und bei der Wahl des Vorgehensmodells vernachlässigt werden kann.

Wichtigkeit der Dokumentation: 1

Die Dokumentation, die im Rahmen der Aufgaben des Teams durchgeführt wird, dient vorrangig dem Team selbst, beispielsweise wenn es um Dokumentation von Abläufen oder Ähnlichem geht. Die Dokumentation über Changes oder Incidents wird durch die korrekte Ausführung der IT Service Management Prozesse sichergestellt, nicht durch das Vorgehensmodell. Die funktionale und technische Dokumentation über die Anwendungen selbst, die für die Kund:innen relevant sein könnte, wird von den jeweiligen Entwicklungsteams zur Verfügung gestellt, nicht vom „Team Betrieb“. Aus diesem Grund wird dieses Kriterium der Dokumentation im Zusammenhang mit dem Vorgehensmodell als eher unwichtig (1) eingestuft.

Natur der Anforderung: 3

Die Aufgaben, die vom „Team Betrieb“ erledigt werden, haben direkten (hohen) Einfluss auf den Erfolg des Versicherungsunternehmens, da der Ausfall eines IT Services hohe finanzielle Schäden inklusive Imageverlust bei Datenleaks oder Ähnlichem mit sich bringen kann. Das gewählte Vorgehensmodell muss daher hinsichtlich dieses Kriteriums geeignet sein, weshalb das Kriterium mit wichtig (3) bewertet wird.

Autonomie der Teams: 0

Das Team kann seine Aufgaben autonom und unabhängig von anderen Teams erledigen, weshalb dieses Kriterium vernachlässigt werden kann und mit unwichtig (0) bewertet wird.

Funktion der Teams: 3

Die Funktion des Teams besteht im Betrieb von IT Services, welches mit vielen ad hoc Anforderungen verbunden ist. Dieser Fakt muss bei der Wahl des Vorgehensmodell auf jeden Fall berücksichtigt werden, weshalb dieses Kriterium mit wichtig (3) bewertet wird.

Örtliche Verteilung der Teammitglieder: 3

Da das Team bewusst auf mehrere Standorte und Zeitzonen verteilt ist, ist dieses Kriterium wesentlich und wird mit wichtig (3) bewertet.

Fluktuation der Teammitglieder: 1

Die Fluktuation im „Team Betrieb“ ist relativ gering und Ausfälle können von den Anderen schnell abgefangen werden, da sie über ähnliche technische Fähigkeiten verfügen. Daher wird die Betrachtung dieses Kriteriums als eher unwichtig (1) angesehen.

8.2.2 Eignung der Vorgehensmodelle

Nach Festlegung der Gewichtung wird nun auch für das „Team Betrieb“ die Eignung jedes Vorgehensmodells pro Kriterium betrachtet.

Teamgröße bzw. Projektgröße

- Scrum: 1
  Die Teamgröße von 18 Teammitgliedern ist für ein Scrum Team definitiv zu groß. Um es anzuwenden, wäre eine Aufteilung der Teammitglieder auf zwei Teams notwendig, weshalb die Methode eher ungeeignet (1) ist.
- Extreme Programming: 0
  XP eignet sich vor allem für kleine bis mittelgroße Teams, und ist vor allem für große Projektstrukturen nicht ausgelegt, weshalb die Methode ungeeignet (0) ist.
- Kanban: 3
  Kanban kann im Gegensatz zu anderen agilen Vorgehensmodellen auch für größere Teams angewandt werden, weshalb die Methode sehr gut geeignet (3) ist.
- Crystal Family: 1
  Da die „größeren“ Modelle der Crystal Family noch nicht näher definiert wurden, eignet sich Crystal vorrangig für Projekte mit bis zu 40 Personen, weshalb die Methode auch für das „Team Betrieb“ mit eher ungeeignet (1) bewertet wird.

Skalierbarkeit:

- Scrum: 3
  Scrum lässt sich mittels LeSS oder SAFe Frameworks gut skalieren und wird daher mit sehr gut bewertet.
- Extreme Programming: 0
  XP ist für kleine bis mittelgroße Teams ausgelegt, und weshalb es sich nicht für eine Skalierung eignet.
- Kanban: 3
  Kanban kann gut in größerem Maßstab skaliert werden, was sich anhand der Anwendungsbeispiele in der Automobilindustrie zeigt, wo Kanban in großen Settings angewandt wird. Daher ist die Methode auch hinsichtlich Skalierbarkeit für das „Team Betrieb“ sehr gut geeignet.
- Crystal Family: 1
  Wenn das Projekt skaliert, müsste auch die Crystal Methode (innerhalb der Crystal Family) gewechselt werden, was eine gewisse Komplexität mit sich bringt und (wie bereits beschrieben) nur beschränkt möglich ist. Aus diesem Grund ist die Methode eher ungeeignet.

Verfügbarkeit der Kund:innen:

Da dieses Kriterium mit einer Gewichtung von 0 bewertet wurde, müssen die einzelnen Vorgehensmodelle in dieser Hinsicht nicht bewertet werden.

Änderungen der Anforderungen / Grad der Flexibilität:

- Scrum: 0
  Bei Scrum sollen die Anforderungen am Beginn eines Sprints bekannt sein, um entsprechend im Sprint Planning geplant werden zu können. Da das „Team Betrieb“ jedoch mehrheitlich ad hoc Anforderungen erfüllen muss, ist die Methode dafür ungeeignet.

- Extreme Programming: 3
  XP lässt Änderungen der Anforderungen auch während des Entwicklungszyklus zu, daher können auch kurzfristig aufgetretene Änderungen umgesetzt werden. Die Methode ist hinsichtlich der Flexibilität also für das „Team Betrieb“ sehr gut geeignet.
- Kanban: 3
  Da Kanban auf Änderungen der Anforderungen sowie ad hoc Anforderungen sofort reagieren kann, eignet sich die Methode ebenso sehr gut für das Team.
- Crystal Family: 0
  Während eines Entwicklungszyklus sind Änderungen der Anforderungen, wie bei Scrum, nicht vorgesehen. Bei den laufend eintreffenden ad hoc Anforderungen des Team ist die Methode daher ungeeignet.

Interdisziplinarität der Teams:

Da dieses Kriterium mit einer Gewichtung von 0 bewertet wurde, müssen die einzelnen Vorgehensmodelle in dieser Hinsicht nicht bewertet werden.

Qualifikation der Teammitglieder:

Da dieses Kriterium mit einer Gewichtung von 0 bewertet wurde, müssen die einzelnen Vorgehensmodelle in dieser Hinsicht nicht bewertet werden.

Wichtigkeit von Dokumentation:

- Scrum: 3
  Da Scrum keine Vorgaben hinsichtlich Dokumentation macht, und der Bedarf an Dokumentation für das Team relativ gering ist (und durch bestehende IT Service Management Prozesse abgedeckt wird), ist die Methode in dieser Hinsicht sehr gut geeignet.
- Extreme Programming: 3
  Der Dokumentationsbedarf des „Team Betrieb“ lässt sich auch in XP gut abbilden.
- Kanban: 3
  Mangels Vorgaben von Kanban bezüglich Dokumentation, ist auch diese Methode gut geeignet.
- Crystal Family: 2
  Der in der Crystal Family hohe Fokus auf Dokumentation wird durch IT Service Management Prozesse abgebildet, weshalb diese Methode „nur“ eher geeignet ist.

Natur der Anforderung:

Bei keinem der Modelle spricht etwas gegen die Anwendung im Bereich der Versicherungs-IT, weshalb sich alle vier Vorgehensmodelle sehr gut eignen.

- Scrum: 3
- Extreme Programming: 3
- Kanban: 3
- Crystal Family: 3

Autonomie der Teams:

Da dieses Kriterium mit einer Gewichtung von 0 bewertet wurde, müssen die einzelnen Vorgehensmodelle in dieser Hinsicht nicht bewertet werden.

Funktion der Teams:

- Scrum: 0
  Scrum eignet sich vor allem für Teams, die neue Funktionalitäten entwickeln, was zumindest eine Planbarkeit entsprechend der Sprintlänge zulassen. Für kurzfristige Betriebstätigkeiten ist die Methode ungeeignet.
- Extreme Programming: 0
  Obwohl Extreme Programming zwar hinsichtlich der Änderungen der Anforderungen grundsätzlich geeignet ist, ist die Methode dennoch nicht für Betriebstätigkeiten konzipiert. Die einzelnen Phasen des Modells, beispielsweise die Explorationsphase und auch die Planungsphase, werden beim „Team Betrieb“ nicht benötigt. Stattdessen arbeitet das Team ausschließlich in der Wartungsphase. Die Methode wird daher als ungeeignet bewertet.
- Kanban: 3
  Kanban eignet sich gut für Betriebstätigkeiten, da das Modell eine schnelle Reaktion auf schlecht planbare aber gleichzeitig hoch priore Tätigkeiten zulässt.
- Crystal Family: 0
  Wie auch Scrum ist die Crystal Family für Entwicklungsprojekte konzipiert und aufgrund der schlechten Reaktionsfähigkeit auf kurzfristige Änderungen ungeeignet.

Örtliche Verteilung der Teammitglieder:

Die Eignung der Vorgehensmodelle hinsichtlich örtlich verteilter Teammitglieder wurde bereits anhand des „Team Entwicklung“ beschrieben und wird daher hier übernommen:

- Scrum: 3
  Die klar strukturieren Zeremonien in Scrum, sowie modere Kollaborationsanwendungen, erleichtern die regelmäßige, wechselseitige Kommunikation. Daher ist das Vorgehensmodell ist für verteilte Teams sehr gut geeignet.
- Extreme Programming: 1
  XP empfiehlt, dass alle Teammitglieder an einem Ort sitzen, was beim „Team Entwicklung“ nicht möglich ist. Da sich die Möglichkeiten für verteiltes Zusammenarbeiten durch Kollaborationsanwendungen jedoch in den letzten Jahren massiv verbessert haben, wird die Methode trotzdem mit einem Punkt (1) bewertet.
- Kanban: 2
  Die örtliche Verteilung der Teammitglieder erschwert die Anwendung der Kanban Methode zwar, verhindert sie jedoch nicht. Eine häufige Abstimmung und Kommunikation der Teammitglieder ist notwendig, damit die Abarbeitung „im Fluss“ bleibt, daher ist die Methode im Falle einer örtlichen Verteilung nur eher geeignet (2).
- Crystal Family: 1
  Da der Fokus der Crystal Family auf Team-Zusammenarbeit und ausgeprägter Kommunikation liegt, eignet sich die Methode besser für Teams, die an einem Ort arbeiten, und wird daher mit eher ungeeignet (1) beurteilt.

Fluktuation der Teammitglieder:

Eine geringe Fluktuation innerhalb des Teams kann grundsätzlich als positiv angesehen werden. Wie bereits im agilen Manifest beschrieben stehen die Menschen in allen agilen Vorgehensmodellen im Vordergrund, deren persönliche Entwicklung sowie die Zusammenarbeit im Team sollen gefördert werden. Alle vier beschriebenen Modelle lassen sich bei konstant bestehenden Teams daher gut anwenden, weshalb alle mit geeignet (3) bewertet wurden.

- Scrum: 3
- Extreme Programming: 3
- Kanban: 3
- Crystal Family: 3

8.2.3 Auswertung „Team Betrieb“

Kriterium		**Scrum**	**XP**	**Kanban**	**Crystal**
Teamgröße bzw. Projektgröße	Gewichtung	2			
	Eignung	1	0	3	1
	Wert	**2**	**0**	**6**	**2**
Skalierbarkeit	Gewichtung	**3**			
	Eignung	**3**	**0**	**3**	**1**
	Wert	**9**	**0**	**9**	**3**
Verfügbarkeit der Kund:innen	Gewichtung	0			
	Eignung				
	Wert	**0**	**0**	**0**	**0**
Änderungen der Anforderungen / Grad der Flexibilität	Gewichtung	3			
	Eignung	0	3	3	0
	Wert	**0**	**9**	**9**	**0**
Interdisziplinarität der Teams	Gewichtung	0			
	Eignung				
	Wert	**0**	**0**	**0**	**0**
Qualifikation der Teammitglieder	Gewichtung	0			
	Eignung				
	Wert	**0**	**0**	**0**	**0**
Wichtigkeit der Dokumentation	Gewichtung	1			
	Eignung	3	3	3	2
	Wert	**3**	**3**	**3**	**2**
Natur der Anforderung	Gewichtung	3			
	Eignung	3	3	3	3
	Wert	**9**	**9**	**9**	**9**

Kriterium		Scrum	XP	Kanban	Crystal
Autonomie der Teams	Gewichtung	0			
	Eignung				
	Wert	**0**	**0**	**0**	**0**
Funktion der Teams	Gewichtung	3			
	Eignung	0	0	3	0
	Wert	**0**	**0**	**9**	**0**
Örtliche Verteilung der Teammitglieder	Gewichtung	3			
	Eignung	3	1	2	1
	Wert	**9**	**3**	**6**	**3**
Fluktuation der Teammitglieder	Gewichtung	1			
	Eignung	3	3	3	3
	Wert	**3**	**3**	**3**	**3**
Summe Wert		**35**	**27**	**54**	**22**

Tabelle 3: Auswertung für „Team Betrieb“

Das Vorgehensmodell mit der höchsten Punktezahl ist Kanban mit 54 Punkten, welches daher als die für das Team geeignetste Methode gewählt wird.

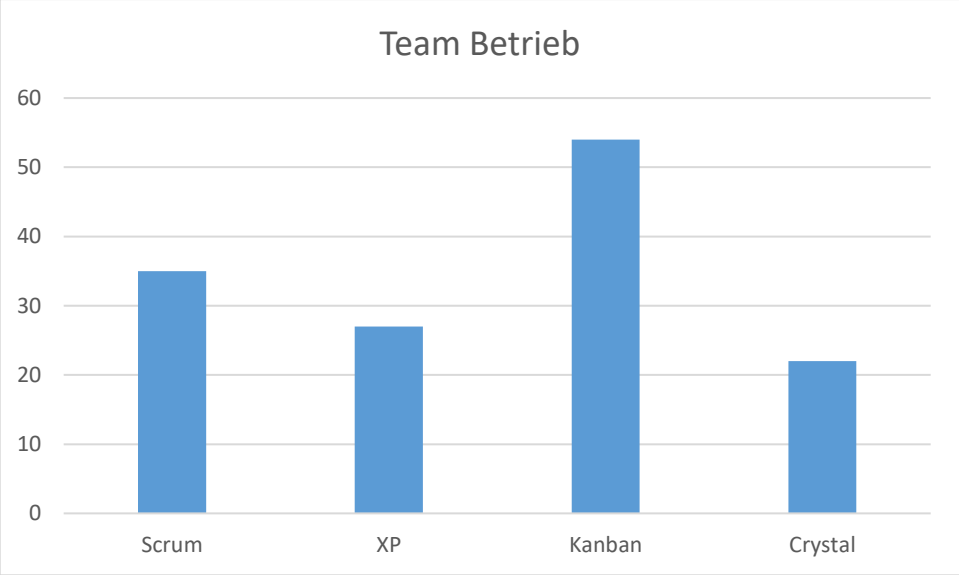


Abbildung 6: Ergebnis für „Team Betrieb“

9. Interpretation der Ergebnisse

Die Auswertung der Methode zeigt in erster Linie, dass es die sprichwörtliche „eierlegende Wollmilchsau“ unter den Vorgehensmodellen nicht gibt. Doch dank genauer Abwägung der 12 Kriterien ist es gelungen, für beide Teams das jeweils geeignetste Vorgehensmodell herauszufinden. Dies kann nun die Entscheidungsträger:innen bei der tatsächlichen Wahl des Vorgehensmodells im praktischen Anwendungsfall unterstützen.

Bei „Team Entwicklung“ wurde Scrum mit 56 Punkten mit der höchsten Punktezahl bewertet, dicht gefolgt von Kanban mit 51 Punkten. Die anderen beiden Vorgehensmodelle sind mit 30 Punkten (Crystal Family) bzw. 29 Punkten (Extreme Programming) als deutlich weniger geeignet bewertet.

Das gewonnene Ergebnis stimmt mit der eingangs formulierten Hypothese überein, dass Scrum sich gut für das „Team Entwicklung“ eignet. Dass Kanban jedoch ähnlich gut geeignet ist, hat mich persönlich ein wenig überrascht.

Die Methode ist jedoch eine qualitative und keine quantitative Bewertungsmethode. Sie ist nicht präzise genug, um eine genaue Festlegung auf „das beste“ Vorgehensmodell zu erlauben. Dennoch kann aufgrund des Ergebnisses die Aussage getroffen werden, dass Scrum und Kanban deutlich besser für das „Team Entwicklung“ geeignet sind als die beiden anderen Modelle, und daher genauer evaluiert werden sollten, damit im konkreten Fall eine Entscheidung getroffen werden kann.

Aus Management-Sicht betrachtet bietet Scrum den Vorteil, dass auch längerfristige (grobe) Planung möglich ist. Aus Erfahrungswerten kann abgeschätzt werden, wie viel ein Team innerhalb eines Sprints abarbeiten kann, wodurch ein grober Zeithorizont errechnet werden kann. Dies ist im Kontext des gesamten Unternehmens ebenso ein Argument das für Scrum spricht.

Der Grund, warum Scrum und Kanban besser geeignet sind als Extreme Programming und Crystal Family liegt meiner Ansicht nach vor allem am Unternehmensumfeld und Projekt und weniger am Team selbst. XP und Crystal Family sind besser in kleineren Projekten bzw. Unternehmen anwendbar. In Umfeld eines Großkonzerns stoßen sie teilweise an ihre Grenzen, weil hier die Konzepte fehlen, sie in größerem Maßstab anzuwenden, ohne auf organisatorische Schwierigkeiten zu stoßen. Selbiges trifft daher auch bei „Team Betrieb“ zu.

Bei „Team Betrieb“ fiel die Punktevergabe jedoch eindeutiger aus: mit 54 Punkten lag Kanban deutlich vor Scrum (35), Extreme Programming (27) und Crystal Family (22). Der Hauptgrund dafür liegt klar an der Art der Tätigkeit. Betriebstätigkeiten und ad hoc Anforderungen sind schwerer in Iterationen einzuplanen, weshalb die Kanban Methode hier gut hervorsticht, da sie eine schnelle Reaktion auf Änderungen im Plan erlaubt. Auch

in diesem Fall stimmt das Ergebnis mit meiner Hypothese überein, dass Kanban für das „Team Betrieb“ gut geeignet ist. Es ist außerdem zu beobachten, dass im Falle von „Team Betrieb“ weniger Kriterien wichtig, diese dafür jedoch umso entscheidender sind.

Da beide Teams im selben Unternehmen und Projekt arbeiten, liegen die Unterschiede zwischen ihnen hauptsächlich in ihrer Tätigkeit und der Teamzusammensetzung.

Ein Vergleich der Punkteanzahl ist nur innerhalb eines Teams zulässig. Der Schluss, dass Kanban mit 51 Punkten für das „Team Entwicklung“ „besser geeignet“ ist, als für das „Team Betrieb“ mit 45 Punkten, kann daher NICHT getroffen werden.

Die gewählte Methode zur Eignungsprüfung der unterschiedlichen Vorgehensmodelle hat grundsätzlich gut funktioniert. Ein großer Vorteil der Methode ist, dass sie zu einer genauen Auseinandersetzung mit den Gegebenheiten bzw. Eigenschaften der Teams zwingt. Ein Nachteil, der bei der Ausführung jedoch beobachtet wurde ist, dass sie relativ zeitaufwändig und die Bewertung in vielen Fällen nicht einfach ist. Es bedarf oft sehr viel Information, um eine gute Beurteilung bzw. Punktevergabe treffen zu können, idealerweise von jemanden, der/die die jeweiligen Teams und ihre Eigenheiten aus der Praxis gut kennt.

Allerdings hilft die Anwendung der Methode auch dabei, über die einzelnen Kriterien gezielt nachzudenken. Die Antwort auf die Frage des passenden Vorgehensmodells wird dadurch von Schritt zu Schritt klarer. Wichtig ist auf jeden Fall, dass die Gewichtung berücksichtigt wird, da diese pro Team stark variieren kann.

Die Auftrennung auf die 12 Kriterien zeigt außerdem, dass es nicht „das eine“ passende Vorgehensmodell gibt. Die Bewertung der Vorgehensmodelle kann von Kriterium zu Kriterium stark variieren, aus diesem Grund ist auch die Gewichtung der Kriterien unerlässlich.

Bei der Bewertung der einzelnen Kriterien konnte festgestellt werden, dass sich manche von ihnen auf das Projektumfeld, andere wiederum auf das Team selbst beziehen. Erstere wurden daher bei beiden Teams gleich beurteilt. Beispielsweise hängt der Bedarf nach Skalierbarkeit mehr vom Projekt ab als vom Team selbst. Im Gegensatz dazu kann die Fluktuation der Teammitglieder von Team zu Team innerhalb eines Projektes variieren, weshalb hier die Bewertung pro Team unterschiedlich ausfallen wird.

Bei der Ausführung der Methode hat sich herausgestellt, dass die Bewertung mancher Kriterien sehr schwer fällt. Die Qualifikation der Teammitglieder erscheint zwar im ersten Moment als relevantes Kriterium, allerdings ist sehr viel Information über das Team notwendig, um diesen Kriterium als Pro oder Kontra für ein Vorgehensmodell betrachten zu können.

10. Conclusio

Die Forschungsfrage lautete: „Welches agile Vorgehensmodell eignet sich am besten für das „Team Entwicklung“ bzw. das „Team Betrieb“ unter Berücksichtigung der verschiedenen Zielsetzungen?“

Die ursprünglich aufgestellte Hypothese lautete: „Für das „Team Entwicklung“ eignet sich Scrum am besten und für das „Team Betrieb“ eignet sich Kanban am besten.“

Die Forschungsfrage kann anhand dieser Arbeit folgendermaßen beantwortet werden:

Für das „Team Entwicklung“ eignen sich sowohl Scrum als auch Kanban nahezu gleichermaßen gut. Für das „Team Betrieb“ eignet sich Kanban am besten.

Die Hypothese wurde somit grundsätzlich bestätigt, mit dem Zusatz, dass für das „Team Entwicklung“ noch eine weitere gute Variante (Kanban) zur Auswahl steht.

Das Ziel der Arbeit lag darin zu zeigen, welches agile Vorgehensmodell für ein Entwicklungsteam bzw. ein Betriebsteam einer Softwareentwicklungsabteilung jeweils am besten geeignet ist, anhand eines konkreten Beispiels aus der Versicherungs-IT. Das Ziel der Arbeit wurde somit erfüllt.

Es muss festgehalten werden, dass die getroffene Wahl des Vorgehensmodells nur für genau diese beiden Teams die richtige ist, und keine pauschalierte Aussage über Entwicklungs- bzw. Betriebsteams im Allgemeinen getroffen werden kann. Die Methode muss für jedes Team erneut angewandt werden, um eine Aussage treffen zu können.

11. Zusammenfassung und Ausblick

11.1 Zusammenfassung

Diese Arbeit stellt einen grundlegenden Überblick über die gängigsten agilen Vorgehensmodelle dar und soll eine Hilfestellung bei der Auswahl eines geeigneten agilen Vorgehensmodell für Teams in der Softwareentwicklung bieten.

Um die Arbeit auch für Außenstehende und branchenfremde Personen gut verständlich zu machen, werden zu Beginn grundlegende Begriffe wie Softwareentwicklung oder IT Betrieb erklärt.

Agile Vorgehensmodelle wurden entwickelt, um Softwareentwicklung schneller und flexibler zu machen, Kund:innen früher in die Entwicklung einzubeziehen und gleichzeitig auch die Zufriedenheit der Mitarbeiter:innen zu erhöhen. Der Grundstein dazu wurde mit dem agilen Manifest gelegt. Im Laufe der Jahre entwickelten sich verschiedene agile Vorgehensmodelle, vier davon werden im Zuge dieser Arbeit näher vorgestellt: Scrum, Extreme Programming, Kanban und Crystal Family. Jedes dieser Modelle zeichnet sich durch unterschiedliche Charakteristiken aus, welche näher behandelt werden.

Um zwischen diesen Vorgehensmodellen eine Wahl treffen zu können, werden 12 mögliche Aspekte bzw. Kriterien aus der Literatur identifiziert.

Eine selbst entwickelte Bewertungsmethode wird anschließend beispielhaft auf zwei reale Teams angewandt, welche im selben Unternehmen in der IT Brache arbeiten. Eines der Teams beschäftigt sich mit der Automatisierung von Deployments in der Cloud, das andere ist das Betriebsteam für diese IT Services.

Nach Anwendung der Methode auf beide Teams zeigt sich, dass sich für das „Team Entwicklung“ Scrum bzw. Kanban nahezu gleichermaßen gut eignen, für das „Team Betrieb“ eignet sich Kanban am besten.

11.2 Ausblick

Eine Möglichkeit, die vorgestellte Methode zu verbessern, könnte darin bestehen, sie um weitere Kriterien zu erweitern. Beispielsweise könnten die jeweiligen Charaktere und Persönlichkeiten der Teammitglieder in die Entscheidung einbezogen werden, oder aber auch die Anforderungen und Bedürfnisse des Managements gegenüber den jeweiligen Methoden. Natürlich bringt das eine weitere Komplexität der Methode mit sich, die allerdings auch möglicherweise Einfluss auf die Aussagekraft der Auswertung hätte.

Hilfreich könnte zukünftig sicher die Entwicklung einer quantitativen Methode sein, welche die Wahl des Vorgehensmodells erleichtert, möglicherweise durch einen

standardisierten Katalog. Dies wäre eine interessante Forschungsaufgabe für eine weitere Arbeit.

12. Literaturverzeichnis

AROCOM GMBH. *Arocom.de.* 2022. https://www.arocom.de/fachbegriffe/webentwicklung/deployment (Zugriff am 15. 01 2022).

Beck, Kent, et al. *Agile Manifesto.* 2001. http://agilemanifesto.org/iso/de/manifesto.html (Zugriff am 12. 12 2021).

Cockburn, Alistair. *Agile Softwareentwicklung.* Boston: Addison-Wesley, 2003.

Cron, Tobias. *Agiles Projektmanagement in der IT mit Kanban.* Mittweida: Hochschule Mittweida, 2013.

Derntl, Michael, Erich Schikuta, und Helmut Wanek. *Grundlagen des Software Engineerings.* Wiener Neustadt: Ferdinand Porsche Fernfachhochschule GmbH, 2019.

Dogs, Carsten, und Timo Klimmer. *Agile Software-Entwicklung kompakt.* Bonn: mitp-Verlag, 2005.

Engel, Sebastian. *Vorgehensmodelle agiler Softwareentwicklung - Ein Überblick.* Saarbrücken: VDM Verlag Dr. Müller GmbH & Co. KG, 2011.

Epping, Thomas. *Kanban für die Softwareentwicklung.* Berlin Heidelberg: Springer-Verlag, 2011.

Highsmith, Jim. *Agile Development Ecosystem.* Boston: Addison-Wesley, 2002.

Hofer, Christian. *Versicherungsmaganzin.* 09. 01 2022. https://www.versicherungsmagazin.de/lexikon/it-betrieb-1945618.html (Zugriff am 09. 01 2022).

Hollenstein, Silvan, und Thomas Rutz. „The Crystal Family." *Crystal Methodologies.* Zürich: Institut für Informatik der Universität Zürich, 2003.

Hruschka, P., C. Rupp, und G. Starke. *Agility kompakt.* Heidelberg-Berlin: Spektrum Akademischer Verlag GmbH, 2004.

Klinger, Sabine. *Kritische Betrachtung klassischer und agiler Vorgehensmodelle unter Berücksichtigung von Erfolgsfaktoren in Softwareentwicklungsprojekten.* Saarbrücken: VDM Verlag Dr. Müller GmbH & Co. KG, 2010.

Krach, Mario. *Prozesscontrolling und -optimierung des Softwareentwicklungsprozesses der Niederösterreichischen Gebietskrankenkasse mit Hilfe von Kanban.* Bischofstetten: Hochschule Mittweida, 2012.

Kriegisch, Alexander. *Scrum-master.de.* 2022. https://scrum-master.de/Was_ist_Scrum/Scrum_auf_einer_Seite_erklaert (Zugriff am 23. 01 2022).

Kumar, Rakesh, Timothy Matche, und Priti Maheshwary. „Inside Agile Family: Software Development Methodologies.“ *International Journal of Computer Sciences and Engineering*, 30. 06 2019.

Larman, Craig, und Bas Vodde. *less.works.* 2014. https://less.works/ (Zugriff am 08. 01 2021).

Onpulson-Lexikon. *Definition: Was bedeutet Agilität?* 11. 12 2021.

Pfeffer, Joachim. *Grundlagen der agilen Produktentwicklung.* Wangen im Allgäu: peppair GmbH, 2019.

Pfitzinger, Bernd, und Thomas Jestädt. *IT Betrieb - Management und Betrieb der IT in Unternehmen.* Berlin Heidelberg: Springer Verlag, 2016.

Scaled Agile, Inc. *scaledagileframework.com.* 01. 01 2020. https://www.scaledagileframework.com/ (Zugriff am 08. 01 2022).

Scrum.org. *Scrum.org.* 2022. https://www.scrum.org/resources/what-is-a-daily-scrum (Zugriff am 23. 01 2022).

Wells, Don. *Extreme Programming: A gentle introduction.* 8. 10 2013. http://www.extremeprogramming.org/ (Zugriff am 12. 12 2021).

13. Abbildungsverzeichnis

14. Tabellenverzeichnis