Institut für Informatik der Freien Universität Berlin

Arbeitsgruppe Software Engineering

Qualitative Feldforschung zur Definition von Done in mehreren Softwareentwicklungsteams

Masterarbeit

Do, Hoang Viet

4126803

do@draphony.de

Berlin, 05.Juni 2014

Betreuer: Prof. Dr. Lutz Prechelt; Sascha Deutschkron

Gutachter: Prof. Dr. Lutz Prechelt

Zweitgutachter: Prof. Dr. Elfriede Fehr

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

05. Juni 2014

Do, Hoang Viet

Zusammenfassung

Das Ergebnis der vorliegenden Arbeit ist die gegenstandsbezogene Theorie „autonomen agile Teams“. Die Theorie stützt sich auf Interviews und Beobachtung von 19 Mitarbeitern einer Unternehmensgruppe.

Die Theorie erklärt wie ein Prozess das Fehlen einer formalen niedergelegten Definition von Done funktioniert und das Team somit zu einem autonom agilen Team wird. Das Prozessmodell strebt nach Entwickler, die nach eigenem Ermessen abwägen, wann eine Änderung als abgeschlossen gilt und wann nicht.

Es handelt sich hierbei um einen Kanban Prozessmodell, dass sich Praktiken aus Extreme Programming hält umso Kommunikation und Feedbackzyklen zu optimieren.

Table of Contents

1 Einleitung 4

1.2 Vergleichbare Arbeiten / Stand der Kunst 4

2 Software-Prozessmodell 6

2.1 Extreme Programming (XP) 6

2.2.1 Die fünf Werte 7

2.2.2 Die 14 Prinzipien 7

2.2.3 Die Praktiken 8

2.2 Scrum 9

2.2.4 Scrum Rollen 10

2.2.5 Scrum-Workflow (im Details) 11

2.2.6 Scrum-Artefakten 12

2.3 Kanban 13

3 Forschungsmethoden 15

3.1 Quantitative Forschung 15

3.2 Qualitative Forschung 15

3.3 Untersuchungstechniken 16

3.3.2 Erhebungsverfahren 16

3.3.3 Aufbereitungsverfahren 19

3.3.4 Auswertungsverfahren 21

4 Angewandten Forschungsmethoden 24

4.1 Datenerhebungsmethoden 24

4.2 Gegenstandsbezogene Theoriebildung 24

4.2.2 Forschungsgebiet 25

4.2.3 Literaturrecherche 25

4.2.4 Datenerhebung 25

4.2.5 Datenauserwertung 26

5 Prozessmodell der untersuchten Teams 31

5.1 Kanban ... 31
5.2 Rollen ... 31
5.2.2 Team Lead ... 32
5.2.3 Tester ... 32
5.2.4 Scrum Master ... 33
5.3 Meetings ... 33
5.4 Agile Praktiken ... 34
5.4.2 Paar Review ... 34
5.4.3 Sit Together ... 34
5.4.4 Test-Driven Development / Akzeptanztest ... 35
5.4.5 Small Releases ... 35
5.4.6 Continuous integration ... 35
5.4.7 Collective code ownership ... 35
5.4.8 Stories ... 35
6 Auswertung ... 36
6.1 Auswertung GT-Theorie mit Agile Manifest ... 39
6.2 Mischform von agilen Prozessen ... 40
6.3 Ausblick ... 41
7 Literaturverzeichnis ... 42

1 Einleitung

Die vorliegende Arbeit „Qualitative Feldforschung zur Definition von Done in mehreren Softwareentwicklungsteams“ untersucht die Existenz und Auswirkung der Definition von Done (ff. DoD) am Beispiel von 3 Tochterunternehmen einer international operierenden Unternehmensgruppe. Zu ihrem Kerngeschäft gehören die Entwicklung und der Bertrieb von Online Verkaufsplattformen.

Die 3 Unternehmen wenden seit Jahren agile Softwareprozesse bei der Entwicklung an. Sie präferieren kurze Entwicklungszyklen gegenüber bücherlange Spezifikationsdokumente.

Die Motivationsfrage für die Forschung der vorliegenden Arbeit war

„Gibt es eine Definition von Done und welche Auswirkungen auf den Prozess hat dieser?“.

Wie und wer entscheidet in agil arbeitenden Unternehmen ab wann eine Aufgabe als erledigt gilt? Noch zu Beginn der Datenerhebung stellte sich heraus, dass es keine einzelne Person für diese Aufgabe gibt oder eine formale niedergelegte Definition von Done existiert. So wandelte sich die Forschungsfrage von der Existenz der Definition von Done zur Frage

„Welche Auswirkungen haben das Fehlen der Definition von Done und wie wird der Done-Zustand entschieden?“.

Für die Forschungsmethodik zur Beantwortung der Frage wurde die gegenstandsbezogene Theoriebildung (kurz GT) von den Forscher Barney Glaser gewählt, da es sich bei der Arbeit um eine offene qualitative Feldforschung handelt.

Das erste Kapitel der vorliegenden Arbeit gibt einen Überblick über Prozessmodelle. In den anschließenden beiden Kapiteln werden die verschiedenen Forschungsmethoden vorgestellt. In den letzten beiden Kapitel 5 und 6 werden die gebildete Theorie und Forschungsergebnisse präsentiert und offene Punkte besprochen.

1.2 Vergleichbare Arbeiten / Stand der Kunst

Zu Agiler Softwareentwicklung existieren zahlreiche Publikationen. Veröffentlichungen zu den elementaren Themen wie etwa die Einführung von speziellen agilen Prozessen wie etwa XP [1, 2, 3, 1, 4], Scrum [5, 6, 7] und Kanban [5, 8] liegen selbsterklärend vor. Einzelne Aspekte und deren Auswirkungen werden dabei nicht nur für „traditionelle“ Softwareentwicklung erforscht, sondern auch Echtzeitsystementwicklung sowie Spieleentwicklung erforscht [9, 10, 11].

Bemerkenswert ist dabei, dass nicht nur professionelle Teams Forschungsgegenstand sind, sondern auch studentische Softwareprojekte [12, 13]. Dies zeigt wie beliebt und vielfältig das Gebiet erforscht wird.

Die vorliegende Arbeit untersucht die Definition of Done in 3 Unternehmen und leitet hieraus eine gegenstandsbezogene Theorie „autonome agile Teams“. Dieser besagt, dass eine Definition of Done durch Anwendungen von Praktiken entfallenkann.

2 Software-Prozessmodell

„Ein Software-Prozessmodell ist ein Modell für die Entwicklung eines Software-Systems[…]. Es gibt grundsätzlich schwergewichtige und leichtgewichtige Prozessmodelle, wobei die agilen Modelle zu den leichtgewichtigen gehören.“ [7]

Über die Jahre wurden in der Informatik zahlreiche Software-Prozessmodelle entwickelt. Neben den traditionellen und schwergewichteten Prozessmodellen, wie etwa das Wasserfall Modell, V-Modell, Unified Process wurden auch leichtgewichtete Prozessmodell entwickelt wie etwa Extreme Programming, Scrum und Kanban, bei denen nicht etwa die Dokumentation im Vordergrund steht sondern der funktionierende Code.

Die leichtgewichtete Prozessmodelle oder auch agile Prozessmodelle folgen dem „Agile Manifest“:

*“**Individuals and interactions** over processes and tools*
***Working software** over comprehensive documentation*
***Customer collaboration** over contract negotiation*
***Responding** to change over following a plan” [14]*

Das Agile Manifest ist die Niederschrift des Grundprinzips agiler Softwareentwicklung. Sie wurde von 17 Entwickler verfasst, worunter sich auch der Erfinder Kent Beck von Extreme Programming und Mitgründer von Scrum Jeff Sutherland befindet.

In diesem Kapitel werden die agilen Prozessmodellen im Allgemeinen dargestellt. Auf die Vorstellung von traditionellen Prozessmodellen wie etwa Wasserfall Modell wird verzichtet, da sie für die Arbeit keine weitere Relevanz hat.

Von den 5 befragten Teams arbeiten 4 Teams mit Kanban als Prozessmodell. Ein Team, das mit Scrum arbeitet, hat es derart angepasst, dass es mehr Kanban ist als Scrum. So fehlen feste Iterationen, Abschätzungen der Aufgaben und Scrum Meetings. Aus den Interviews und Beobachtungen geht auch hervor, dass sehr viele Elemente – insbesondere Praktiken – aus dem Extreme Programming das Prozessmodell prägen. Eine Aussage aus den Interviews ist, dass Extreme Programming sogar vorher genutzt wurde.

2.1 Extreme Programming (XP)

Extreme Programming (kurz XP) wurde 2003 von Kent Beck im Rahmen des Chrysler Projects Chrysler Comprehensive Compensation (C3) entwickelt [7, 2].

XP ist ein agiles und iteratives Prozessmodell. Jeder Iteration durchläuft dabei 4 Phasen: die Planung, Design, Kodieren und Testen. Während jeder Iteration, muss sich der Entwickler an die Praktiken halten, die in XP definiert sind. Diese beruhen auf die 14 Prinzipien, welche wiederum auf die 5 Werte Kommunikation, Einfachheit, Feedback, Mut und Respekt aufbauen.

Die Werte (Abschnitt 2.1.1), Prinzipien (Abschnitt 2.1.2) und Praktiken (Abschnitt 2.1.3) werden nachfolgend erläutert.

2.1.1 Die fünf Werte

Die 5 Werte sind Kommunikation, Einfachheit, Feedback, Mut und Respekt.

Mit Kommunikation ist sowohl die interne Kommunikation innerhalb des Teams, welches die Grundlage für ein gemeinsames Vorgehen im Projekt ist, als auch die Kommunikation mit dem Kunden gemeint. Beck fordert sogar, dass der Kunde als Teammitglied vor Ort zu akzeptieren ist [3].

Bei der Einfachheit wird gefordert, dass nur die aktuellen Gegebenheiten berücksichtigt werden. Anforderungen, die in einem späteren Release umgesetzt werden müssen, sollen nicht berücksichtig werden. Hierdurch wird die Komplexität verringert und nur das geliefert, was auch (im Moment) gefordert wird.

Zu den Feedbacks gehören zum einen das Feedback des Systems in Form von Testfällen und das Feedback vom Kunden. Die Forderung ist, dass ein Kunde der sowohl fachlich qualifiziert ist als auch die notwendige Entscheidungsgewalt hat das Feedback gibt.

Mut als Wert ist notwendig, denn gefordert wird der Mut zur Veränderung, die Bereitschaft Designs anzupassen und Refactoring durchzuführen. Darüber hinaus setzen die 3 Werte Kommunikation, Einfachheit und Feedback bereits Mut voraus.

Der 5. Wert, der erst in seinem 2. Buch hinzugefügt worden ist [3, 2], ist Respekt untereinander, sowohl im Team als auch gegenüber dem Kunden. Es bildet die Basis für einen vertrauensvollen gegenseitigen Umgang.

2.1.2 Die 14 Prinzipien

Die 14 Prinzipien von XP sind Menschlichkeit, Wirtschaftlichkeit, Wechselseitiger Vorteil, Selbstähnlichkeit, Verbesserung, Vielfältigkeit, Reflexion, Fluss, Gelegenheit, Redundanz, Fehlschlag, Qualität, Kleine Schritte und Akzeptierte Verantwortung.

Beispielsweise soll die Arbeitsumgebung menschenwürdig sein so dass sich die Entwickler wohlfühlen (Menschlichkeit). Die Software andererseits ist zur Erzeugung des Gewinns für das Unternehmen. Es muss wirtschaftlich sein, denn die schönste technische Lösung nützt nix, wenn sie nicht bezahlbar ist. Das müssen Entwickler ebenfalls akzeptieren (Wirtschaftlichkeit). Wobei die Qualität nicht verhandelbar ist (Qualität).

Die Teamzusammensetzung muss auf Vielseitigkeit bauen um verschiedene Sichtweisen zu beinhalten (Vielfältigkeit). Das Team strebt die ständige Verbesserung der eigenen Lösung, Produkt und Prozess an. Hierzu ersetzt das Team alte Praktiken, die nicht funktionieren durch Neue (Verbesserung). Das Team reflektiert und beruft hierzu aktiv Sitzungen ein um darüber zu debattieren (Reflexion). Das Team arbeitet kontinuierlich in Iterationen, dessen Ausgang ein ausgeliefertes Release ist (Fluss), wobei die Iterationen in möglichst kleinen Schritten erfolgen sollen (Kleine Schritte).

Eine detaillierte Beschreibung der restlichen und ebenso kurz erläuterten Prinzipien ist in Becks Buch [2] zu finden.

2.1.3 Die Praktiken

In seiner Ausgabe von 2004, stellte der XP-Erfinder Beck 2 Arten von Praktiken vor. Erstere sind primäre Praktiken, die ein Team adoptiert. Hat es eine gewisse Reife und beherrscht die primären Praktiken, so kommen die Folgepraktiken zum Einsatz.

Anstelle von (langen) Pflichtenheften, werden User Stories genutzt. Es handelt sich dabei häufig um kurze Sätze der Form

„Als <Rolle> möchte ich <Aktion>, um <Grund>.“

Diese Praxis wird als „Stories“ bezeichnet. XP definiert 2 Zyklen, den wöchentlichen Zyklus und vierteljährigen Zyklus. Ersteres definiert die Iterationslänge („Weekly Cycle“). Während der vierteljährlichen Zyklus die Dauer Releases entspricht, welche geplant werden („Quarterly Cycle“). Die Umsetzung der Änderungen erfolgt über ein inkrementelles Design. D.h. Es wird nur für den aktuellen Zweck entworfen, nicht für evtl. weitere Anforderungen in der Zukunft („Incremental Design“).

Das Team sitzt räumlich in der Nähe von einander, unter anderem im selben Raum („Sit Together“). Es verfügt über alle notwendigen Kompetenzen um die ihm gestellten Aufgaben zu erfüllen („Whole Team“). Die Arbeitsumgebung des Teams soll den aktuellen Stand des Projekts wiederspiegeln worunter beispielsweise die aktuellen Tasks fallen („Informative Workspace“). Dem Team soll jedoch auch der Freiraum geboten werden sich mit Dingen zu beschäftigen, die kein

Task sind aber in alle Regel mit dem Projekt zu tun haben. Die hierdurch gewonnen kreativen Ideen wirken sich dann auf das Team („Energized Work“, „Slack“).

Das Team arbeitet immer paarweise an seinen Aufgaben. Nach Beck wird hierdurch die Code Qualität gesteigert [2] („Pair Programming“). Zudem sollen immer zuerst die Testfälle definiert und implementiert werden bevor die eigentliche Umsetzung durchgeführt wird („Test-First Programming“). Zusätzlich hierzu wird kontinuierlich integriert, Code Änderungen werden in kurzen Abständen integriert („Continuous Integration“).

Beherrscht das Team die primären Praktiken, können weitere Praktiken eingeführt und durch das Team vorgeschlagen werden. Einige Praktiken, die sich in der Praxis durchgesetzt haben [5, 2, 7, 4, 1]:

- Planungsspiel,
- Kleine Releases,
- Messung der Projektgeschwindigkeit,
- Iterationen und Iterationsplanung,
- Move People Around,
- Stand-Up Meeting,
- Fix XP When It Breaks,
- Einfachheit des Designs,
- Systemmetapher,
- CRC-Karten,
- Risikominimierung durch “Spike Solutions”,
- Refactoring,
- Der Kunde ist immer verfügbar,
- Kodierungsstandards,
- Collective Code Ownership,
- Optimiere erst zum Schluss,
- Keine Überstunden,
- Usw.

2.2 Scrum

Scrum ist ein Prozessmodell, das von Ken Schwaber und Jeff Sutherland Anfang der 90er-Jaren entwickelt um die Organisationen bei Umsetzung komplexer Projekte zu unterstützen [6].

Scrum ist ein inkrementelles, iteratives Prozessmodell, bei dem aus jeder Iteration ein Inkrement des Produkts hervorgeht. Eine Iteration wird **Sprint** genannt. Während eines Sprints halten die Mitglieder des Teams täglich ein Standup Meeting um untereinander die Aktivitäten abzustimmen und ggf. entsprechende Anpassungen vorzunehmen. Dieses Meeting wird **Daily Scrum Meeting** genannt. Die Ziele des Sprints und somit auch das Inkrement stellt eine Liste von Anforderungen dar, dem **Sprint Backlog**.

Zu Beginn eines jeden Sprints, kommt das Team, welche die Umsetzung übernimmt, zusammen und stimmt mit einem sogenannten Product Owner ab, welche Arbeitspakete es glaubt anhand seiner Erfahrungen, Kenntnisse u. ä. in diesem Sprint schaffen zu können. Anschließend arbeitet das Team selbstständig an der bestmöglichen Umsetzung gemäß der Sprintdauer und präsentiert am Ende des Sprints das Inkrement mit den abgestimmten Anforderungen. Dieser wird dann von den Stakeholder verifiziert. Der selbstständige und kreative Prozess der Lösungsfindung vom Team „bildet das Kernstück der Produktivität von Scrum“ [6].

2.1.4 Scrum Rollen

Scrum definiert 3 Rollen:

- Der Product Owner,
- Den Scrum Master,
- Und das Team.

Der Produkt Owner ist verantwortlich die Interessen von allem zu vertreten, die einen Anteil an das Projekt und dem Ergebnis haben. Er übernimmt die Budgetierung, durch die Definition der Gesamtanforderungen zum Projektstart und die Ziele für die Kapitalrendite (ROI[1]) sowie die Release-Pläne.

Die Anforderungsliste wird als Product Backlog bezeichnet. Sie ist priorisiert und liegt in der Verantwortung des Product Owner. Die Prioritäten werden von ihm in kurzen Abständen aktualisiert.

Das Team ist für die Entwicklung der Funktionalitäten verantwortlich. Es verwaltet und organisiert sich selbst und ist für den Erfolg jeder Iteration und des Projekts als Ganzes verantwortlich.

Der Scrum Master ist der Prozessverantwortlicher. Er ist für den Scrum Inhalt verantwortlich. Daher schult und fördert er alle, die ausführend an dem Projekt beteiligt sind. Er ist für die Sicherstellung, dass jeder sich an die Scrum-Regeln und –Verfahren hält.

[1] Return on investment

Alle die nicht in einer der 3 Rollen fallen und in eine anderen Form das Projekt beeinträchtigen, haben keine Befugnis die Dinge zu unternehmen, die für den Erfolg des Projekts notwendig sind.

2.1.5 Scrum-Workflow (im Details)

Zu Beginn des Projekts erstellt der Product-Owner entsprechend der Projektvision das Product-Backlog. Die darin enthaltenen funktionalen und nicht-funktionalen Anforderungen werden priorisiert und in voraussichtliche Release-Termine unterteilt.

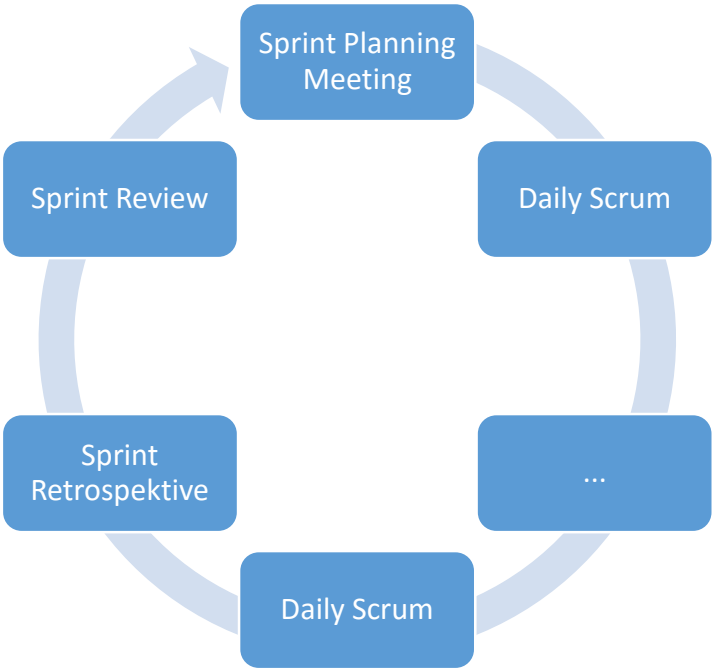

Abbildung 1: Sprintverlauf

Abbildung 1 illustriert den Verlauf eines Sprints.

2.1.5.2 *Sprint Planning Meeting*

Die Umsetzungsarbeiten sind rein Sprintbasiert. Ein Sprint dauert in der Regel 30 Tage [6]. Der Anfang eines Sprints beginnt mit dem 2 teiligen Meeting namens „**Sprint Planning Meeting“** [6], die jeweils 4 Stunden in Anspruch nehmen und somit in der Summe **maximal** 8 Stunden dauern. Das Zeitlimit ist einzuhalten, denn es geht hierbei darum die Arbeit einzunehmen und nicht darüber nachzudenken.

Im ersten Teil stellt der Product Owner Anforderungen mit den höchsten Prioritäten aus dem Product Backlog dem Team vor und bespricht das Sprintziel mit ihm. Das Team kann bei Bedarf Fragen zu Inhalt, Zweck, Bedeutung und zu den damit verbundenen Absichten stellen. Anschließend gibt es sein Versprechen - „***Commitment“*** [6] **-** ab, dass es sein Bestmögliches tuen wird, um das Inkrement vollständig zu liefern.

Im zweiten Teil plant das Team den Sprint voraus. Die anfallenden Aufgaben für den Sprint werden in einem Sprint Backlog festgehalten. Es handelt sich hierbei nur um eine Startapproximation, da das Team autark arbeitet. Änderungen können jederzeit vom Team durchgeführt werden.

2.1.5.3 *Daily Scrum Meeting*

Jeden Tag kommt das Team zu einem 15 minütigen Treffen zusammen, damit jedes Mitglied die nachfolgenden 3 Fragen beantwortet:

1. Was haben Sie in diesem Projekt seit dem letzten Daily Scrum Meeting durchgeführt?
2. Was planen Sie bis zum nächsten Daily Scrum-Meeting?
3. Welche Hindernisse sind bei der Umsetzung für diesen Sprint und das gesamte Projekt aufgetreten?

Der Zweck des Meetings ist es, die Arbeit der Teammitglieder abzustimmen, evtl. weitere Meetings zu planen, die das Team für weiteren Fortschritt und die Erfüllung seines Versprechens braucht.

2.1.5.4 *Sprint Review Meeting*

Am Ende des Sprints präsentiert das Team dem Product Owner und allen Stakeholder, die daran teilnehmen möchten, in einem informellen Meeting das Inkrement. Unter anderem können hier Überlegungen diskutiert werden, was als Nächstes vom Team umgesetzt werden soll.

2.1.5.5 *Sprint Retrospektive Meeting*

Darauffolgend treffen sich Scrum Master und das Team zu einem Sprint Retrospektive Meeting, bei dem der Scrum Master dazu animiert über den Scrum Prozess und Arbeitsweise reflektiert und optimiert.

2.1.6 Scrum-Artefakten

In diesem Abschnitt werden in Detail die Artefakte von Scrum besprochen:

- das Product Backlog,
- das Burndown-Diagramm,
- und Sprint Backlog.

2.1.6.2 *Product Backlog*

Das Product Backlog ist eine priorisierte Auflistung der funktionalen und nicht-funktionalen Anforderungen für das im Rahmen des Projekts zu entwickelndes System / Produkt. Die Verantwortung und Pflege liegt beim Product Owner.

Zu Beginn des Projekts wird das Product Backlog als eine Ausgangsschätzung erstellt. Es ist eine dynamische und niemals vollständige Auflistung. „Solange ein Produkt entwickelt wird, gibt es auch ein Product Backlog“ [6].

2.1.6.3 *Burndown-Diagramm*

Das Burndown-Diagram zeigt den Betrag des verbleibenden Arbeitsaufwandes über den ganzen Zeitraum.

Die nachfolgende Abbildung 2 zeigt ein Beispiel eines Burndown Diagramm. Es illustriert den noch offenen Arbeitsbedarf.

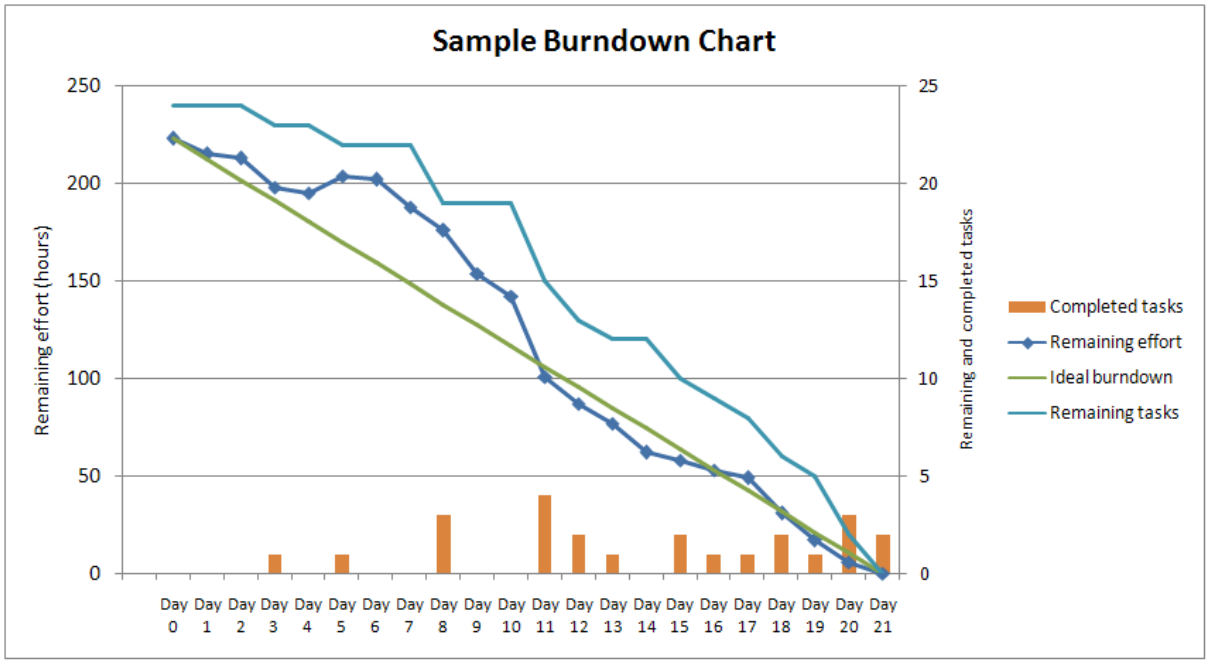

Abbildung 2: Beispiel eines Burndown-Diagramm[2]

2.1.6.4 *Sprint Backlog*

Das Sprint Backlog ist ein aussagekräftiges Echtzeitabbildung der durchzuführenden Arbeiten, die das Team für den Sprint plant. Das Team und nur das Team kann das Sprint Backlog ändern. Zu Beginn des Sprints - in dem Sprint Planning Teil 2 – erstellt es eine Ausgangsversion.

2.3 Kanban

Kanban wurde 2007 von David J. Anderson vorgestellt [15, 5]. Das Wort Kanban stammt aus den beiden Japanischen Wörter Kan („Signal“) und Ban („Karte“). In den beiden Worten lassen sich bereits 2 Kernelemente von Kanban herleiten. Im Gegensatz zu XP und Scrum definiert Kanban an sich keine Iterationen oder (strikte) Regeln. In seiner einfachsten Form ist Kanban lediglich das Kanban Board – ein Board auf dem die Stories gelistet ist. In Abbildung 3 ist ein beispielsweise ein solches Board skizziert.

[2] Lizensfreies Bild aus Wikipedia, Zugriff 10.5.2014

Das Board ist dabei eine Taskqueue auf denen Aufgaben sind, die das Team nach dem Pull Prinzip abarbeiten, d.h. Keine Taskzuweisung sondern Taskauswahl.

Die Einführung von Kanban lässt sich in wenigen Schritten durchführen:

1. Erfasse den aktuellen Entwicklungsphasen
2. Definiere den Start- und Endpunkt des Kanban-Flusses.
   Die beiden Punkte sollten bevorzugt etwas sein, auf das das Team Einfluss hat.
3. Abstimmung
4. Erstelle das Kanban Board und nutze Kanban.
   Zum Beispiel auf einem Whiteboard.

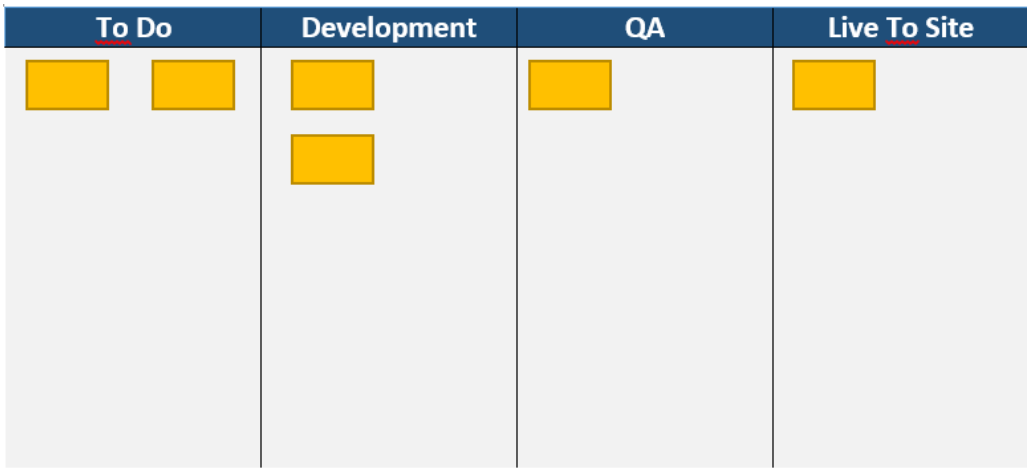

Abbildung 3: Kanban Board

Im Schritt 1 muss geklärt werden, woher die neuen Aufgaben kommen und welche Phasen sie durchlaufen bis sie den End-User erreichen? Im Schritt 2 wird eine Phase als Startpunkt des Kanban Board genommen und eine Phase als Endpunkt des Kanban Boards definiert.

Für das Beispiel in Abbildung 1 kommen die neuen Aufgaben als ToDo auf das Board. Die nächsten Phasen sind Development, QA und zum Schluss erfolgt der Upload (Live To Site). Das Live To Site ist hier der Endpunkt des Kanban Boards.

Im Schritt 3 wird die maximale Anzahl an parallele Taskabarbeitungen bestimmt. Diese Anzahl wird als WIP (Work In Progress) bezeichnet. Anschließend wird das Board erstellt.

Für Kanban gilt ebenfalls ein regelmäßige Reflektieren und Optimieren. Die Aufgaben sollten etwas in etwa den gleichen Aufwand haben.

3 Forschungsmethoden

Die empirische Forschung unterteilt sich in qualitative und quantitative Methoden. Während die quantitative Datenerhebung mit standardisierten Daten arbeitet, wird die qualitative Forschung individualisiert und interpretativ behandelt.

3.1 Quantitative Forschung

Unter der quantitativen Forschung wird ein Projekt außenperspektivisch beobachtet und anhand statischer, bereits feststehender Daten, auf vorab festgelegte Hypothesen überprüft. Bekanntermaßen greift man hierbei auf messbare Zahlen, oder auch stichprobenartige Prüfungen zurück, um kausale Zusammenhänge zu erklären und zu belegen.

Die Methoden werden anhand des formulierten Forschungsproblems ausgewählt, wobei diese auf zuvor per Stichproben ausgewählte Personen angewandt werden.

Hierzu zählen:

- die Beobachtung des Experiments,
- ein Interview mit stets gleichbleibenden Fragen (verbal, wie non-verbal),
- Inhaltsanalyse bereits bestehender Hypothesen (hermeneutische Beobachtung)
- Erhebung der Daten aus deskriptiven Statistiken (Beschreibung eines Problems)
- Erhebung der Daten aus Inferenzstatistiken (Wahrscheinlichkeiten)
- Erhebung der Daten aus multivariaten Statistiken (Datenreduktion, z.B. Ähnlichkeiten)

Resultierend aus der linearen Erhebungsstruktur folgen die Datenauswertung, die Prüfung der formulierten Hypothesen und der abschließende Bericht, beruhend auf Objektivität, Validität und Reliabilität.

3.2 Qualitative Forschung

Der Fokus der qualitativen Forschung liegt darin das Problem der Betroffenen anhand subjektiv erfasster Daten zu interpretieren. Mithilfe von dynamischen und auf das Forschungsproblem individuell angepassten Maßnahmen erfolgt ein nicht standardisierter Forschungsansatz, welcher realitätsnahe Daten zur Folge hat.

Wesentlicher Bestandteil des Forschungsansatzes ist der direkte Kontakt mit den Probanden/Befragten, sodass möglichst alltagsbezogene Sachverhalte aus einem neuen Blickwinkel betrachtet werden können

Bezogen auf die „Grounded Theory“ [16] lassen sich dank der qualitativen Forschungsverfahren gegenstandsbezogene Ergebnisse ermitteln, welche sich während der Analysen dynamisch an die jeweilige Ergebnissituation anpassen.

Die aus der induktiven Perspektive ermöglichte Offenheit lässt es zu lebenswirkliche Ergebnisse zu erzielen, wobei Sinn und Bedeutung von Handlungen, sowie Äußerungen, der Teilnehmer stets im Kontext zum Ur-Thema betrachtet werden.

Während der verschiedenen Erhebungs-, Aufbereitungs-, und Auswertungsverfahren gilt somit jeder Teilnehmer als Experte in seiner Befragung und das insbesondere aufgrund eines im Vorhinein erarbeiteten Vertrauensverhältnisses zwischen Forscher und Proband.

Die Schwierigkeit liegt hierbei gleichzeitig den wissenschaftlichen Prozess zu fokussieren, aber gleichzeitig Teil eines dynamischen und variablen Arbeitsgefüges zu sein, sodass eine Störung der subjektiven Forschungsergebnisse vermieden wird.

3.3 Untersuchungstechniken

Im Zuge der qualitativen Forschung werden drei wesentliche Analyseverfahren durchlaufen. Diese werden bezogen auf die Arbeit angerissen, auch wenn eine ausführliche empirische Arbeit stets erweitert und vertieft werden kann.

3.3.2 Erhebungsverfahren

Unter den Erhebungsverfahren fallen 3 verbale (vgl. folgend, Problemzentriertes Interview, Narratives Interview, Gruppendiskussion) und eine non-verbale (Teilnehmende Beobachtung) Herangehensweise.

Bereits hier erfährt die qualitative Forschungsweise einen erheblichen Unterschied zum quantitativen Verfahren, da der Fokus auf subjektive Sichtweisen gerichtet ist und somit jeden Beitrag als Experteninhalt bewerten lässt.

3.3.2.2 *Problemzentriertes Interview*

Das “fokussierte” Interview ist bereits seit 1945 durch Merton und Kendall geschaffen worden, um die Wirkung von Medien der Massenkommunikation zu analysieren.

Dabei wurden bestimmte Inhalte hypothetisch betrachtet, um diese dann anhand von subjektiven Erfahrungen von Anwendern und Konsumenten (z.B. Rundfunk und Zeitung) bestätigen oder ergänzen zu lassen. Seit jeher gilt der Leitfaden, dass Erfahrungen, Wahrnehmung und insbesondere die Reflexion der Befragten zu einem offenen Gespräch führt, allerdings bezogen auf ein bestimmtes Problem.

Die dabei entstehende Vertrauenssituation ermöglicht den Befragenden bewusst schrittweise Daten im Rahmen seines Problemfeldes zu gewinnen und zu prüfen, während der Befragte stets ohne Vorgabe oder monoton auf Fragen antworten muss, sodass seine Erfahrung als Mehrwert gilt.

Das Problem wird dabei bereits am Anfang formuliert und als zentraler Gesprächspunkt eines Kreislaufschemas bewertet.

Hieraus lässt sich eine Abhängigkeit, sowie Flexibilität der Bedingungen ableiten, sodass die Auswahl der Personen einen direkten Einfluss auf die Daten und auch somit auf das gesellschaftlich nahe Verständnis des Urproblems hat (zur Erinnerung: die stichprobenartige Ermittlung der quantitativen Forschung lässt nur eine Prüfung der Hypothese zu).

Mithilfe 3 bestimmter Fragetypen lässt sich das Gespräch strukturieren und passiv moderieren. Diese wären die:

- Sondierungsfragen (Einstieg in das Thema, Ermittlung der Wichtigkeit)
- Leitfadenfragen (vorgezeichnete Fragen, welche beantwortet werden müssen)
- „Ad-hoc-Fragen“ (spontane Fragen zur Konkretisierung nicht vorhergesehener Gesprächsentwicklungen)

Insofern greift das offene Gespräch die Möglichkeit ein bereits formuliertes Problem kontrovers zu interpretieren und eine eigenständige Hypothese zu erstellen.

Auch wenn eine Standardisierung nicht das Ziel einer qualitativen Analyse ist, so vereinfacht die vorhergehende Situations- und Hypothesenanalyse die Ermittlung der Ziele. Die Aufzeichnung kann somit stets auf eine bestehende Struktur zurückgreifen, aber verliert nie den Charakter eines offenen Gesprächs und der Möglichkeit der Themenexploration.

3.3.2.3 *Halbstrukturiertes Interview.*

Das Halbstrukturierte Interview ist eine Verfahrensart des fokussierten Interviews. Es basiert auf einen variablen Umfang und lässt dem Befragenden die Option die Handlungs- und Frageziele festzulegen, oder diese möglichst offen zu lassen.

Anhand eines Gespräch Leitfadens, basierend auf Schlüssel- und Eventualfragen, wird ein vorgegebener Stichpunktkatalog abgearbeitet, sodass das Ergebnis der Interviews zu einer bislang unerforschten Hypothese führt. Nachteilig zu betrachten ist, dass diese Methode aufgrund des offenen Ausganges und der nicht fest definierbaren Lösungsansätze keine Vergleichbarkeit innerhalb eines Forschungsauftrages zu.

Die Methodik des halbstrukturierten Interviews ist das in dieser Arbeit gewählte Erhebungsverfahren, sodass die Arbeitsmethodik im weiteren Verlauf detaillierter vorgestellt wird.

3.3.2.4 *Narratives Interview*

Mithilfe dieser Erhebungsmethodik wird das Interesse durch das bloße Zuhören auf Informationen, also subtile bis zu detaillierten Hinweisen, zu dem zuvor herausgearbeiteten Forschungsproblem zu ermitteln. Es wird bewusst ein freies und offenes Gespräch antizipiert, sodass die Inhalte des Befragten explorativ betrachtet werden. Unter diesem Aspekt sind subjektive Annahmen möglich.

Der Einstieg in das Themengebiet erfolgt abstrakt und stellt lediglich eine Definition zur Orientierung zur Verfügung. Im Gesprächsverfahren werden 3 Teilpunkte betrachtet. Mithilfe der Eingangsfrage wird das Thema vorgestellt, die Präsentation der Erzählung konzentriert sich auf das vorangehende Thema und zum Schluss werden erst Nachfragen gestellt und die Evaluation abgeschlossen.

3.3.2.5 *Gruppendiskussion*

Der grundsätzliche Unterschied zwischen den Einzelbefragungen ist hierbei die Gruppendynamik, welche in einem Kontext mögliche Hinweise auf Geleitgesinnungen (Ideologien) zulässt. Nachteilig zu betrachten ist, dass unter Umständen aufgrund sozialer Zusammenhänge subjektive und egoistische Meinungen nicht vertreten sind, aber gleichzeitig durch ein „gemeinsames Ziel“ psychische Sperren durchbrochen werden und kollektive Einstellungen zu einem speziellen Thema deutlich werden.

Während der Forscher stets als Moderator agiert, stellt er aber während der Darbietung eines Grundreizes gleichzeitig ein Gruppenmitglied dar, welcher eine freie Diskussion willkommen heißt. Gezielte Reizargumente lassen somit das Vertrauen leichter entstehen, als in einem Einzelinterview (Gedankenansatz: Er ist einer von uns!). Allerdings besteht die Schwierigkeit darin auch im Alltag agierende Gruppen zu ermitteln.

3.3.2.6 *Teilnehmende Beobachtung*

Die Schwierigkeit der teilnehmenden Beobachtung ist es als homogenes Mitglied eines Experiments aufzutreten, obwohl der Forschende selber lediglich eine passive Rolle bezüglich des Experiments einnimmt, aber eine aktive im Sozialgeschehen. Somit ist diese Methodik eine der sensibelsten Verfahren, da sowohl die Akzeptanz der Gruppe (Vertrauensverhältnis) erlangt werden muss und gleichzeitig nicht als Störfaktor ermittelt werden darf.

Das grundsätzliche Ziel ist es einen Forschungsgegenstand möglichst nah zu kommen und dieses aus einer Innenperspektive der Umzusetzenden zu erschließen, sodass ein alltäglicher Zugang zu der Thematik entsteht.

Beobachtungen werden konkret und vergleichsweise zu weiteren qualitativen Verfahren sehr konkret festgehalten (Beobachtungsprotokoll), aber es darf nie der Anschein entstehen, sich gezielt auf die vorher formulierten Fragen zu konzentrieren, da somit die Feldforschung ihren freien Charakter verliert.

3.3.3 Aufbereitungsverfahren

Während die Erhebung sich rein auf die Datenermittlung konzentriert ist der zentrale Punkt der qualitativen Analyse die Aufbereitung der Daten. Unter diese Fallen verschiedene Techniken, welche hier grob aufgefasst werden und nach Priorität erläutert werden.

Grundsätzlich erfolgen bei der Materialaufbereitung die Schritte der:

- Darstellungsmittelwahl
- Protokollierung
- Konstruktion

Bekanntermaßen wird gerne auf grafische Darstellungen zurückgegriffen, welche durch konkrete und leicht verständliche Sinnzusammenhänge überzeugen, aber den wissenschaftlichen Aspekt einer qualitativen Analyse nur selten gerecht werden.

Auch das implizieren der Forschungsteilnehmer in die Aufbereitung ist möglich, allerdings selten umgesetzt, auch wenn es im Beiderseitigen Interesse der Entwicklungspsychologie stünde. Selbsteinschätzungen und Deutungen von Sachzusammenhängen müssten somit nicht in einem Interview ermittelt werden.

3.3.3.2 *Wörtliche Transkription*

Die Transkription ist die Verschriftung von ausgesprochenen Wörtern, also Gesprächen und Interviews, also die vollständige Textfassung von verbal erhobenen Material.

Als effektive Methode gilt hierbei der zuvor (vgl. Erhebungsverfahren) erarbeitete Leitfaden, welcher stichpunktartig oder phonetisch protokolliert wird.

Für ein effizientes Interview muss der Befrager somit sowohl stringent bleiben, als auch „weiche“ Fragen formulieren, um das Vertrauensverhältnis aufrecht zu erhalten, aber gleichzeitig wesentliche Inhalte zu beziehen.

Dank moderner Techniken erfolgt die Protokollierung mittlerweile zielorientiert und hilft dem Befragenden unnötige Informationen zu filtern und die konsistenten Materialien zu selektieren.

Die inhaltlich-thematische Ebene steht hierbei stets im Vordergrund und liefert die Basis für ausführliche Interpretationen im Auswertungsverfahren. Ansonsten lässt die wörtliche Transkription keine weiteren Freiheiten zu.

3.3.3.3 *Kommentierte Transkription*

Ergänzend zu der wörtlichen Transkription existiert die Methode der kommentierten Transkription, welche zu der Verschriftung ebenso Hinweise bzw. Details bzgl. des Verhaltens des Befragten, oder Besonderheiten im Gesprächsverlauf (Auffälligkeiten) zulässt. Hierbei wird ein besonderes Augenmerk auf:

- Pausen
- Betonungen
- Zusatzinformationen (Abwegig vom Basisthema) gelegt.

Weiterhin ist es zulässig und hilfreich bereits hier interpretierend zu agieren, sodass die Auswertung gleich auf vorhandene Daten zurückgreifen kann.

3.3.3.4 *Zusammenfassendes Protokoll*

Das zusammenfassende Protokoll filtert bereits in der Erstellung häufige, ähnliche und gleichbleibende Daten, sodass durch die systematische Zusammenfassung ein höheres Allgemeinheitsniveau erzielt wird.

Das Ablaufmodell ist wie folgend festgehalten:

- **Aulassen:**
  An mehreren Stellen bedeutungsgleiche Aussagen werden weggelassen (insofern sie sich ableiten lassen).
- **Generalisation:**
  Abstrakte Propositionen welche eine begriffliche Überordnung ermöglichen, werden angewandt, statt allgemein verlautbaren Begriffen (Mehrfachanwendung).
- **Konstruktion:**
  Eine globale Proposition, welche den Sachverhalt zusammenfasst.
- **Integration:**
  Aussagen welche sich thematisch unterordnen lassen werden innerhalb einer Konstruktion eingegliedert.
- **Selektion:**

Bestimmte Aussagen bleiben aufgrund ihres Alleinstellungmerkmales gesondert gehalten.

- **Bündelung**
  Inhaltlich zusammenhängende Aussagen, welche aufgrund ihrer thematischen Differenzierung getrennt wurden, werden in eine einheitliche Textform gebracht

Wie bereits ersichtlich ist die Reduktion durch Bündelung, Selektion und Konstruktion gleichzeitig die Ermittlung neuer Aussagen als Kategorie System, welche primär die inhaltlich-thematischen Aspekte des Forschungszieles ermittelt.

Im Gegensatz zu den hier nicht erörterten deskriptiven und selektiven Methoden wird weitestgehend global gearbeitet, sodass trotz der Zusammenfassung selektive Bestandteile erhalten bleiben, ohne dass sich das Protokoll aufgrund einer Vorabfilterung verliert und der wissenschaftliche Aspekt der Befragung nicht erfüllt werden kann. Das Ziel ist ein Abstraktionsniveau zu bestimmen und dieses zu erhöhen, indem ein eigenständiger und innovativer Kontext konstruiert wird.

Das eruieren der inhaltlich-thematischen Faktoren der Probanden verläuft somit bereits abhängig von der ausgewählten Aufbereitungsmethode, ohne dass diese den Erhebungsprozess beeinflusst.

Weitere Methoden die als Aufbereitungsverfahren ausgewählt werden können sind das selektive Protokoll und die Konstruktion deskriptiver Systeme, welche allerdings aufgrund ihrer Ausführlichkeit nicht näher betrachtet werden, um den Rahmen nicht zu sprengen.

3.3.4 Auswertungsverfahren

Die Verfahren zur Auswertung der erhobenen Daten unterteilen sich in der Frage welches Material zur Verfügung steht. Im Bereich der qualitativen Forschung hat sich eine große Bandbreite an Auswertungsmethoden bewährt, sodass hier lediglich 3, welche für die vorliegende Arbeit in Betracht gezogen wurden, erläutert werden.

3.3.4.2 *Gegenstandsbezogene Theoriebildung*

Der Grundgedanke der qualitativen Forschung, also die Entwicklung, Verfeinerung und Verknüpfung von Datensammlungen, Hypothesen und Konstrukten, während der Datensammlung, spiegelt sich in der “Grounded Theory” von Glaser/Strauss wieder. Bereits während des Erhebungsverfahrens wird vorwiegend die induktive Konzept- und Theoriebildung zugelassen, sodass die Aufbereitungsverfahren konkretisiert werden und gleich in der Auswertung münden.

Die gegenstandsbezogene Arbeit an einem Forschungsfeld ermöglicht die Transparenz der Konzeption, sodass gleichzeitig die Beschränkung auf das Ideal des kritischen Rationalismus gebrochen wird und eine reine Hypothesenüberprüfung vermieden wird, was eine Entwicklung und dynamische Anpassung der vorangestellten inhaltlich-thematischen Ziele ermöglicht.

Im Ablauf der Theoriebildung ist ein wesentlicher Bestandteil die „Memo-Funktion“, welche mit Hinblick auf das zusammenfassende Protokoll den passenden Baustein darstellt, sodass eine aufeinander aufbauende und ergänzende Konzeption entsteht, statt wie im quantitativen linear zu arbeiten. So können Forschungsergebnisse durch Selbst- und Partnerevaluation reifen und neue Aspekte klären, zusätzlich zu der Konkretisierung und Differenzierung bereits genannter oder ermittelter.

Wie in der vorliegende Arbeit folgend erkenntlich ist dieses Auswertungsverfahren ideal für die Feldforschung mit einem längerfristigen Kontakt mit dem Gegenstandsbereich, da die Datensammlung weitestgehend von den Memos und Kodes ermittelt wird. Gleichzeitig ist der Ansatz einer kreislaufschematischen Arbeit das Sinnbild einer explorativen Untersuchung in insbesondere neuen und unerforschten Themengebieten.

3.3.4.3 *Qualitative Inhaltsanalyse*

Die qualitative Inhaltsanalyse orientiert sich schematisch an der Aufbereitung des zusammenfassenden Protokolls.

Als systematische Analyseverfahren gelten die:

a) Zusammenfassung:
Reduktion des Materials auf die Erhaltung wesentlicher Inhalte zur Abstraktion eines überschaubaren Forschungskorpus.

b) Explikation:
Erweiterung des inhaltlich-thematischen Verständnisses anhand von Erläuterungen und Deutungen.

c) Strukturierung:
Erstellung eines Querschnitts bestimmter Aspekte indem festgelegte Ordnungskriterien erfüllt werden.

Ziel ist es Kategorien zu definieren und Anker zu setzen, ohne in vorschnelle Quantifizierungen abzurutschen und das theoriegeleitete Kategoriensystem im Zentrum der Auswertung steht. Die Nähe zur quantitativen Analyse ermöglicht eine strukturierte Ableitung und greift somit auch vernachlässigte Aspekte der eben genannten Analysemethode auf.

3.3.4.3.1 Typologische Analyse

Das typologische Analyseverfahren konzentriert sich darauf aus einer großen Materialmenge typische Bestandteile herauszufiltern und diese näher zu identifizieren. Es werden Teilkategorien bestimmt, welche das Material in besonderer Weise repräsentieren.

Die Typisierung erfolgt in:

- Idealtypen,
- Besonders häufige Fälle (Typen),
- Besonders seltene Fälle (Typen),
- Extremtypen,
- Fälle von besonderem Interesse

Grundsätzlich erfolgen zwei voneinander unabhängige Durchsichtungsabläufe des Materials, wobei sich der erste Durchgang auf die Typisierungsdimensionen und – Kriterien konzentriert, während im zweiten die Veranschaulichung anhand von Filterung und Differenzierung erfolgt.

Allerdings setzt dieses Verfahren eine Rücküberprüfung der Fragestellung voraus, sodass das im Vorhinein bestimmte Material möglicherweise ergänzt wird.
Die Zuordnung in Typenkategorien ist für eine Fülle an vorhandenem Datenmaterial von Vorteil, insbesondere da auf detaillierte Fallbeschreibungen nicht verzichtet werden muss, aber baut auf deskriptive Systeme auf, sodass dieses Verfahren im Protokoll keine Anwendung findet.

Weitere Methoden der Auswertung sind unter anderem die:

- sozialwissenschaftlich-hermeneutische Paraphrase
- die objektive Hermeneutik
- die psychoanalytische Textinterpretation
- die phänomenologische Analyse

Diese werden hier allerdings aufgrund ihrer Ausführlichkeit und Abweichung des Zieles dieser Arbeit nicht näher erläutert.

4 Angewandten Forschungsmethoden

In diesem Kapitel werden die gewählte Forschungsmethode und Datenerhebungsmethoden im Detail beschrieben.

GT ist besonders für offene qualitative Feldforschung geeignet. Es fordert keine klare Forschungsfragestellung zu Beginn. Darüber hinaus sind keine speziellen Anforderungen wie beispielsweise repräsentative Teilnehmerzahl notwendig, dessen Erfüllung nicht garantiert werden konnte.

4.1 Datenerhebungsmethoden

Aufgrund der vielen unbekannten Faktoren wurde das Halbstrukturierte Interview gewählt. Es ermöglicht flexibles und sofortiges Agieren bei der Durchführung, was nötig war, da die Forschungsfrage und das Forschungsumfeld zu Beginn nicht klar definiert wurden.

Des Weiteren ist das Eingreifen möglich, wenn der Befragte die Frage evtl. falsch verstanden hat oder unterschiedliche Bezeichner genutzt werden. Insbesondere Letzteres würde angenommen, da einer der Motivationen der Arbeit „Unterschied zwischen Praxis und Theorie“ ist. So wussten einige der Befragten nicht, was ein „Ticket“ ist, da es in Jira als „Issue“[3] bezeichnet wird. Bei eine Online Umfrage wäre diese Information verloren oder die erhobenen Daten fehlerhaft. Für die Befragten ist es ein geringer Aufwand, schnell ein paar Minuten für ein Interview zu nehmen im vgl. zu eine Umfrage.

Das Interview fungierte als Einstiegsmethode um einen 1. Einblick in die Unternehmen zu gewinnen, insbesondere relevante Strukturen, Rollen, agile Prozesse sowie agile Praktiken zu extrahieren.

Es wurden insgesamt 19 Mitarbeiter aus 8 Teams, die jeweils einer der 3 Tochterfirmen FA, FB FC befragt. Die Teilnehmer gehörten im Allgemeinen zu einer der Rollen „Backendentwickler“, „Frontendentwickler“, „Teamleiter“, „Quality Assurence“ und „Product Owner“. Die Interviews gingen von ungefähr 30 – 90 Minuten.

4.2 Gegenstandsbezogene Theoriebildung

Die Gegenstandsbezogene Theoriebildung (fortfolgend auch als GT bezeichnet) wurde von den beiden Forscher Barner Glaser und Anselm Strauss entwickelt. Es ist definiert als

[3] What is an Issue, https://confluence.atlassian.com/display/JIRA/What+is+an+Issue, Zugriff 2013-12-02

„a general Methodology of analysis linked with data collection that uses a systematically applied set of methods to generate and inductive theory about a substantive area“ [17]

Übersetzt lautet die Definition: es ist eine mit Datensammlung verbundene allgemeine Analysemethode, die systematisch durch die Anwendung eines Methodensets zur induktiven Theoriebildung, über eine substantiellen Bereich ermittelt.

Die GT baut auf den Methoden des stetigen Vergleichs der erhobenen Daten und Reflektion über die erhobenen Daten um **Memoing** ausüben zu können. Das Level der Abstraktion wird dabei mit jeder Wiederholung gesteigert bis zur Bildung der Theorie.

4.2.2 Forschungsgebiet

Zu Beginn, wird ein Forschungsgebiet gewählt. Zu beachten ist, dass die Methodik von einer Forschungsfragestellung oder ähnliches absieht. Durch dessen Existenz kann die Sichtweise der Forscher beschränkt oder sogar gelenkt werden. Im Mittelpunkt der Forschung steht die Induktion von Theorien aus der Datensammlung.

Die vorliegende Arbeit untersucht „Agile Prozesse und Agile Praktiken“.

4.2.3 Literaturrecherche

Glaser warnt strikt davor bereits in der Anfangsphase der Forschung umfassende Literaturrecherchen durchzuführen, denn es würde sich ähnlich, wie bei der Festlegung einer Fragestellung, verhalten. Die Induktion der Theorien wird dann in eine Richtung gelenkt.

Die vorliegende Arbeit hat sich weitestgehend an Glasers Rat gehalten. Literaturrecherche wurde nur bis zur Aneignung von Grundlagen zu den Themen durchgeführt. Kenntnisse zu dem Thema agile Prozesse waren bereits vorhanden.

4.2.4 Datenerhebung

Bei der Datenerhebung hatte die Unternehmensgruppe X anfänglich Unterstützung angeboten. Die Ergebnisse der vorliegenden Arbeit bauen auf Interviews und Beobachtungen von 14 Mitarbeitern aus 3 Firmen einer gemeinsamen Unternehmensgruppe, die sich einen gemeinsamen Standort teilen. Befragt wurden Backendentwickler, Frontend, Teamleads, Mitglieder des QA Teams und Product Owner.

Die Interviews wurden vor Ort in den Räumen der Firmen durchgeführt. Die 3 Firmen sind zum Zeitpunkt der Untersuchung in dem Bereich E-Commerce tätig.

4.2.4.2 *Interviews und Beobachtung*

Daten wurden über Interviews erhoben und durch Beobachtungen, wie etwa passive Teilnahmen an Meetings gestützt. Es wurden 30 – 90 minütige Leitfadeninterviews (engl. „semi structural interviewes“) vor Ort mit den Teilnehmern durchgeführt. Den Teilnehmern wurden offene Fragen zum Forschungsbereich „Agile Prozesse und Agile Praktiken“ gestellt.

Neben den Eröffnungsfragen, wie etwa Rolle und Team, wurden die Teilnehmer

1. nach Erfahrungen mit agilen Prozessen und agilen Praktiken befragt,
2. welche agilen Prozesse und agilen Praktiken sie kennen und bei der Arbeit verwenden,
3. welche Vor- und Nachteile sofern vorhanden sehen sie in dessen Anwendung,
4. wie sie dessen Auswirkungen bewerten.

Zu den Interviews wurden zudem noch Beobachtungen durchgeführt:

- Teilnahme an der wöchentlichen“Open Space“ Session,
- Teilnahme an den Daily Stand Ups,
- Ausstattung und Gestaltung der Arbeitsräume

Die Interviews wurden sofern erlaubt aufgenommen und anschließend kommentiert zusammengefasst.

4.2.5 Datenauserwertung

Die Datenauswertung wird im GT Kontext als **Coding** bezeichnet. Sobald Daten vorliegen, kann die Codierung durchgeführt werden. Hierbei werden 2 Arten von Codes unterschieden: substantielle Codes (engl. „S*ubstantive codes*“) und theoretische Codes (engl. „*Theoretical codes*“). Substantielle Codes werden wie folgt definiert:

> *Substantive codes “are the categories and properties of the theory which emerges from and conceptually images the substantive area being researched” [18]*

Übersetzt lautet die Definition: substantielle Codes sind Kategorien und Eigenschaften der Theorie, welche aus dem zu untersuchenden Forschungsgebiet hervorgehen und konzeptuell verbildlichen.

Der theoretische Code ist hingegen wie folgt definiert:

Theoretical codes "implicitly conceptualize how the substantive codes will relate to each other as a modeled, interrelated, multivariate set of hypothesis in accounting for resolving the main concern" [18]

Übersetzt lautet die Definition: theoretische Codes konzipieren implizit wie die substantiellen Codes miteinander in Beziehung stehen werden als ein modelliertes, zusammenhängendes und ein multivariates Set von Hypothesen, die zur Lösungsfindung des Hauptanliegen betrachtet werden.

Die nachfolgende Kapitel beschreiben die Codierungsmechanismen:

- **Offene Codierung** (engl. „Open Coding") und **Selektive Codierung** (engl. „Selective Coding"), die zu substantielle Codes führen,
- Und **theoretische Codierung**, die zu den theoretischen Codes führen.

4.2.5.2 *Offene Codierung*

Die offene Codierung ist der erste Schritt der Datenanalyse und dient zur detaillierten Auswertung der erfassten Daten.

Hierbei wurden aus den Interviews die Kernaussagen extrahiert. Dieser wurden im Anschluss in Codes umgewandelt, wobei ein Code ein Stichpunkt der Kernaussage ist. Zum Beispiel:

- **Kernaussage:** Von Done wird gesprochen, wenn die Änderung ausgerollt wurde.
  Code: Änderung ist ausgerollt
- **Kernaussage:** Wenn die Änderung seit einigen Tagen bereits läuft.
  Code: Pilot durchlaufen
- **Kernaussage:** Von Done wird gesprochen, wenn die Änderungen die verschiedene Tests abgeschlosse hat wie etwa Unittests, Integrationstest, u.ä.
  Code: Änderung wurde getestet

4.2.5.3 *Methode des Stetigen Vergleich*

Codes die aus der Datenerhebung hervorkommen werden kontinuierlich miteinander verglichen. Diese Methode wird als Methode des stetigen Vergleichs (engl. „Constant Comparison Method") genannt.

Bei der vorliegenden Arbeit wurden die Codes aus dem Interview mit den Codes aus dem selbigen und anderen Interviews und den Beobachtungen verglichen. Anschließend wurden die Codes noch einmal miteinander verglichen um Gruppen zu bilden, welche als **Konzepte** (engl. „concepts") in GT bezeichnet wird. Die Konzepte sind auf einen höheren Level der Abstraktion.

Die nachfolgende Abbildung 4 zeigt (ausschnittsweise) das Konzept für „DoD von Tasks“.

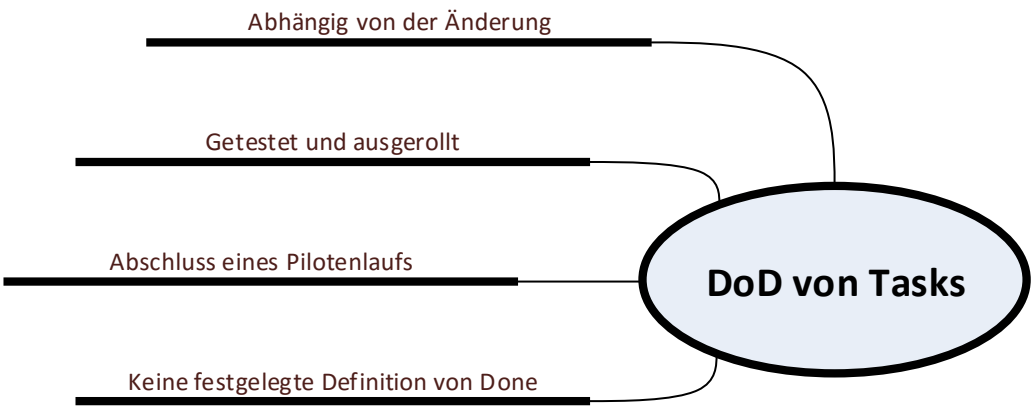

Abbildung 4: Beispiel von einem Konzept "Definition of Done"

Wird die Methode des stetigen Vergleichs nochmal auf der Ebene der Konzepte zur Gruppierung wiederholt, entstehen **Kategorien**. Diese sind in einer Abstraktionsebene über die Konzepte geordnet.

In der Abbildung 4 wäre die Kategorie „Definition of Done von Tasks“. Es gibt keine festgelegte Definition of Done für Tasks. Zudem ist dieser auch abhängig von der jeweiligen Änderung. Eine Änderung gilt unter anderem als abgeschlossen, wenn sie getestet, ausgerollt und eine Pilotenlaufdauer erfolgreich abgeschlossen hat.

4.2.5.4 *Kernkategorien*

Das Ende der Offenen Codierung ist gekennzeichnet durch das Finden der **Kernkategorie**, welche wie folgt von Glaser definiert wird

> *The core category „accounts for a large portion of the variation in a pattern of behavior“ and is considered the “main concern or problem” for the participants [17].*

Die Kernkategorie muss das Zentrum sein. Es muss in einen bedeutenden und direkten Bezug zu den anderen Kategorien und deren Codes stehen, zudem muss es wieder in den Daten zu finden sein.

Die Forscherin Rashina Hoda beschreibt in ihrer Dissertation das Finden der Kernkategorie und somit auch die Fragestellung als „time consuming“ (zeitaufwendig) und „tedious“ (mühsam)

[19]. Es ist nach ihrer Auffassung „perhaps the most demanding part of the whole GT process“[4] [19] – möglicherweise der am meisten herausfordernde Abschnitt des gesamten GT Prozess.

Die Kernkategorie der vorliegenden Arbeit lautet „autonome agile Teams“. Die Mitglieder des Teams können selbst den Done Zustand eine Änderung selbst entscheiden. Hierzu bedarf es Praktiken, die ein gemeinsames Verständnis weitestgehend sichern.

Es stellte sich heraus, dass aktiv auf Formalitäten verzichtet wurden – wie etwa eine formale Definition von Done – zu Gunsten der Ausübung von agilen Prozessen.

4.2.5.5 *Selektive Codierung*

Nach dem die Kernkategorie erfasst wurde, tritt die Phase der Selektiven Codierung ein. Hierbei wird ähnlich wie bei der Offenen Codierung weitere Daten erhoben. Die Analyse zur Extraktion der Codes beschränkt sich dann jedoch nur noch auf die Kernkategorie der relevanten Codes.

Eine Kategorie erreicht die „*theoretical saturation*“ [20], wenn weitere Datenerhebungen und Analysen keine neuen oder nur geringfügige Erkenntnisse hervorbringen.

Bei der vorliegende Arbeit trat dieser Punkt in den letzten Interviews ein, aus welchen bei den neuen Interviews nur noch bereits erfasste Informationen hervorgehen.

Die Erhebung von Daten aus weiteren Datenquellen, wie etwa die Logs der Versionsverwaltung oder die Projektverwaltungssoftware, blieb aufgrund von mangelnden Zugriffsrechten aus.

4.2.5.6 *Memoing*

Zur Unterstützung der Theoriebildung wird die Technik Memoing angewendet, bei dem der Forscher seine Gedanken als Memonotiz zu Grunde legt. Das ist keine Phase, sondern eine begleitende Technik.

4.2.5.7 *Sortierung*

Nachdem die Datenerhebung abgeschlossen ist, beginnt die Sortierung der gewonnenen Informationen und Memoiren. Es handelt sich hierbei nicht um die chronologische Sortierung, sondern um die konzeptuelle, d.h. welche Elemente gehören zueinander und in welcher Ordnung stehen sie zu einander.

[4] Deutsch: womöglich der herausfordernste Part des gesamten GT Prozess.

4.2.5.8 *Hauptliteraturrecherche*

Nachdem die Theorie entstanden ist, beginnt die Hauptliteraturrecherche. Wie zuvor erwähnt, sollte die Literaturrecherche immer im Anschluss durchgeführt werden, da sie sonst die Sichtweise der Forscher beeinflusst.

4.2.5.9 *Theoretical Coding*

Die Theoretische Kodierung ist ein optionale Schritt des GT, welche von Glaser aber wie folgt beschrieben wird

„a GT is best when they are used“ [18]

Bei diesem Schritt handelt es sich um die Zusammensetzung der Kategorien und zugehöriger Konzepte und daraus resultierender Integration der Theorien. Hierfür hat Glaser mehrere Strukturen gelistet:

- The Six C’s (causes, contexts, contingencies, consequences, covariances, and conditions)
- Process (stages, phases, passages, etc.)
- Degree family (limit, range, intensity, etc.)
- Dimension family (dimensions, elements, divisions, etc.)
- Type family (type, form, kids, styles, classes, genre)
- Usw.

4.2.5.10 *Write-Up*

In der Write-Up Phase wird die entstandene Theorie formuliert.

Die These der vorliegenden Arbeit lautet Autonome agile Teams durch gemeinsames Verständnis. Anstelle eine ausformulierten und niedergelegten Definition of Done, entscheidet das jeweilige Teammitglied nach eigenem Ermessen den Done Zustand einer Änderung. Hierfür wird ein gemeinsames Verständnis angestrebt, das unter anderem aber auch durch diverse Praktiken unterstützt wird.

Im nachfolgenden Kapitel wird das Prozessmodell der untersuchten Teams vorgestellt. Im Anschluss die Auswertung präsentiert.

5 Prozessmodell der untersuchten Teams

In diesem Kapitel wird der allgemeine Prozess der untersuchten Teams vorgestellt. Dabei werden die Rollen, eingesetzten Tools und der Prozess näher erläutert. Im nachfolgenden Kapitel werden die Ergebnisse ausgewertet.

5.1 Kanban

Die befragten Teams arbeiten ausschließlich nach Kanban oder ein stark an Kanban orientierte Version von Scrum. Bei Beiden ist fest zu stellen, dass sie sich an eine große Anzahl von Praktiken aus dem Extreme Programming halten.

Das Kanban-Board wird über eine webbasierte Lösung digital realisiert, das dann über Wandmonitor in ständige Sichtbarkeit bleibt (vgl. „Informative Workplace“). Iterationen gibt es nicht.

Die Product Owners definieren „Epics“ in einem „Meta-Kanban-Board“. Das Team stimmt mit dem Product Owner über das nächste Epic ab, und erstellt hierzu Ticket auf seinen eigenen Kanban-Board. Das ganze Team sollte möglichst nur an eine Epic arbeiten. Das ist insbesondere mit fortschreitende Bearbeitungsfortschritt nicht immer möglich, so dass die Arbeit auch an 2 oder mehreren Epics notwendig ist.

Nach dem die von Epic gewünschten Änderungen umgesetzt und ausgerollt wurden, werden vordefinierte Metriken erfasst. Werden diese gewünschten Metriken nicht erreicht, kann das Epic sogar rückgängig gemacht machen und die Änderungen widerrufen werden. Diese Metriken sind Teil des Epics. Ein solches Kriterium kann zum Beispiel die „Steigerung der Value per Visits um 10%“ sein.

Die übliche Bearbeitungsdauer eines solchen Epics beträgt 1 – 3 bzw. 2 – 4 Wochen. Hier unterscheiden sich die Aussagen.

5.2 Rollen

Ein Teammitglied kann dabei mehrere der nachfolgende Rollen ausüben: Backend-Entwickler, Frontend-Entwickler, Teamleiter, Qualitätssicherung (ff. Auch als Tester genannt), Produkt Owner und Scrum Master. Abbildung 5 illustriert die Rollen, die zu einem Team gehört.

Ein Team besteht aus einem „Team Lead“, 1 QA-Mitarbeiter, 1 – 2 Backend- und einem Frontend-Entwickler. Der Product Owner vom Team hat seinen Arbeitsplatz ebenfalls in der Nähe vom Team (vgl. „Sit Together“ und „Whole Team“)

Gelegentlich werden auch Agenturen mit eingebunden, die das Layout für den Frontend-Entwickler designen.

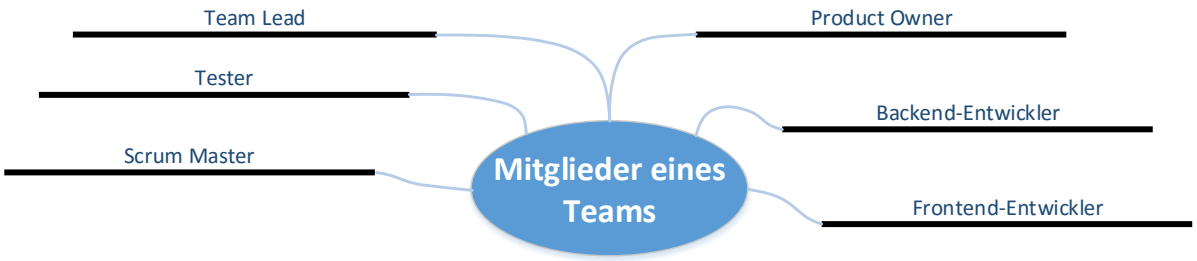

Abbildung 5: Potentielle Rollen de Teams

5.2.2 Team Lead

Die Rolle vom Team Lead wird bei allen befragten Teams von einem Backendentwickler zusätzlich übernommen. Dieser betreut die Weiterentwicklung der Teammitglieder und legt ebenfalls dessen Gehalt fest. Diese Rolle bringt sowohl Vorteile als auch Nachteile für die Ausübung nach einem agilen Prozessmodell.

Vorteilhaft ist, dass der Vorgesetzt ein Mitglied des Teams ist und in ständige Kommunikation durch die Meetings und Zusammenarbeit mit den anderen Teammitgliedern ist. Das schafft:

- Vertrauen in die Person und Verständnis,
- Und Vertrauen in die Fähigkeit, dass er Informationen hat um eine faire Bewertung abzugeben.

Auf der anderen Seite, wird bei der Untersuchung vermutet, dass evtl. Einflüsse bei fachliche Diskussionen vorliegen können. So sind Teammitglieder natürlich etwas vorsichtiger, wenn sie mit den Vorgesetzten „diskutieren“. Dieser Vermutung konnten nach Aussagen der Befragten ebenfalls bestätigt werden.

5.2.3 Tester

Die Aufgaben des Testers werden teilweise von den Entwickler mitübernommen. Nach Aussagen der Befragten soll sich dieser Arbeitsweise verstärken und die Rolle des Testers in Zukunft komplett entfallen, was zum einen die Bildung von Wissensinsel unterbindet und damit auch verbunden die Praxis der kollektiven Codeverantwortung steigert.

Zudem drängt sich die Vermutung, dass durch die Rolle des Testers die Praxis „Test-First“ negativ beeinfluss wir. Der Entwickler tendiert hier dazu, das Schreiben der Test vor der Entwicklung zu vernachlässigen.

5.2.4 Scrum Master

Ein spezielles Teammitglied, das Vollzeit Scrum Master ist, gibt es nicht. Diese Rolle wird von einem der Teammitglieder zusätzlich ausgeübt.

Von den 5 befragten Teams hat nur eines einen Scrum Masters. Dessen Aussage ist ebenfalls, dass sich das Team weiterhin in Richtung Kanban entwickeln wird. Scrum ist nur für die Einführung in agile Entwicklung gedacht.

Vorteile von dieser Praxis sind:

- eine Auslastung des Mitarbeiters ist garantiert.
- Ist das Team sicher in seinem Umgang mit Scrum, wird die Rolle abgeschafft ohne, dass der Mitarbeiter lange Umfunktionierungszeit benötigt.
- Es ist günstiger für die Firma als ein weiteres Teammitglied aufzunehmen.
-

Nachteile der Praxis sind:

- Die Rolle des Scrum Master ist es den Prozess als Unabhängige Partei zu betreuen und die Scrum Regel zu forcieren. Da er ebenfalls Teil des Teams ist, ist die Unabhängigkeit nicht immer garantiert.
- In Meetings ist der Scrum Master der Diskussionsleiter, seine Doppelrolle könnte die Funktion des Diskussionsleiters beeinträchtigen. Insbesondere, weil er als Übermittler zwischen dem Team und den Product Owner agieren soll. Bei Bedarf muss er den Product Owner an Änderungen hindern, die das Commitment gefährden.

5.3 Meetings

Die Teams führen täglich 1 – 2 “Daily Stand-Up” Meetings. Das Erste am Morgen zu Beginn des Arbeitstag und das Zweite 4 Stunden danach.

In dem Daily Meeting werden Informationen über die laufende Entwicklungsarbeiten, und somit auch den Fortschritt im Team ausgetauscht und unter einander abgeglichen. Für einige Befragten wird durch die beiden Meetings sichergestellt, dass unterschiedliche Verständnisse der einzelne Task und auch der Epic hier aufgedeckt werden.

„Der Product Owner nimmt an den Meetings teil[...] und wissen somit was ich mache. Daher kann es keine großen Differenzen geben“ zwischen der Vision und das was ich mache.

Nach Aussagen der Product Owner, werden ausgerollte Änderungen nicht häufig zurück genommen.

Zusätzlich gibt es einen wöchentlichen „Open Space“ am Mittwoch, an welchem jeder teilnehmen kann und frei und spontan sein Anliegen besprechen kann. Für das Treffen gibt es keine vordefinierte Agenda, kein Zeitlimit und die Teilnahme ist optional.

Obgleich der „Open Space“ übersetzt Freiraum bedeutet, ist hier keineswegs gemeint, dass der Entwickler hier andere projektbezogene Interessen nachgehen kann. Es geht hier insbesondere um Diskussionen bzgl. der Arbeitsatmosphäre und die Optimierung des Prozesses. Dem Zweck des Meetings nach ist es äquivalent zur Scrum Retrospektive.

5.4 Agile Praktiken

In diesem Abschnitt werden weitere Praktiken erläutert, die nicht in den vorherigen Abschnitt erwähnt wurden.

5.4.2 Paar Review

Ein Teammitglied U soll vor dem Upload des Codes in die Repository mit einem anderem Teammietglied V im Team die Änderungen reflektieren.

Hierdurch ist sichergestellt, dass mindesten ein weiteres Mitglied aus dem Team von den Änderungen weiß und U. Er bezeugt somit auch das gemeinsame Verständnis, denn sowohl U als auch V sehen die Änderung als valide.

Als ein weiterer Nebeneffekt kann U selbst nochmal über seine Änderung sprechen und reflektieren. Hierbei geht er gedanklich und verbal über den Anwendungsfall, wodurch evtl. nicht bedachte Szenarien hinzukommen.

Das Paar Review nach eigenem Ermessen von dem Teammitglied auch weggelassen werden kann, zum Beispiel bei sehr kleine Änderungen.

5.4.3 Sit Together

Dieses Prinzip ist aus der Beobachtung hervorgegangen. Mehrere Teams teilen sich einen Raum, in welchem auch mindesten ein Product Owner seinen Arbeitsplatz hat.

Dadurch, dass die Teams an dem selben Produkt arbeiten ist es ebenfalls ein fördernde Vorgehensweise. Diese Praxis entstammt dem eXtreme Programming, das seinerseits besagt, dass der Kunde zu jeder Zeit für Fragen zur Verfügung stehen sollte.

Durch die vorliegende Konstellation ist nicht nur der Product Owner, sondern auch weitere Teams für Fragen vor Ort. Es hat aber auch den Nachteil, dass sich Teammitglieder durch die vielen Diskussionen gestört fühlen, was im Open Space ebenfalls angesprochen wurde.

5.4.4 Test-Driven Development / Akzeptanztest

Tests werden zwar nicht immer vor der Implementierung durchgeführt, doch sind festes Bestandteil jeder Änderung. Dabei werden neben Unittests auch Akzeptanztests erstellt und ausgeführt.

5.4.5 Small Releases

Bei der befragten Unternehmensgruppe gibt es keine festgelegten Entwicklungszyklen. Änderungen können im Grunde am selbigen Tag ausgerollt werden. Hierfür wird eine Softwarelösung genutzt, die den Integrationsprozess automatisiert.

An einem Tag werden um die 3 Release ausgeführt nach Aussagen der Befragten.

5.4.6 Continuous integration

Eng verbunden mit „Small Releases“ ist die kontinuierliche Integration. Hierbei kommt die Software X zum Einsatz, die das Integrieren und Bauen übernimmt und somit das zeitnahe Ausrollen der Änderungen ermöglicht.

5.4.7 Collective code ownership

Durch die Festlegung von Rollen in der aktuellen Konstellation, wird dieses Prinzip für das Produkt noch nicht konsequent umgesetzt.

5.4.8 Stories

Statt Lastenhefte, arbeiten die befragten Teams mit User Stories. Das Kanban Board wird dabei über Jira realisiert und dann über Monitore (im Arbeitsraum) permanent angezeigt. Jedes Team hat sein eigenes Board auf Jira. Über ein weiteres Jira Board wird das Meta Kanban Board realisiert. Es enthält die Epics.

Zusätzlich realisieren Teams ein 3. Kanban Board zur Synchronisierung des Teamfortschritts mit anderen Teams. Auf diesen Boards werden keine User Stories abgebildet, sondern nur Referenzen auf die Jira User Stories.

6 Auswertung

Das Ergebnis der vorliegenden Arbeit ist die gegenstandsbezogene Theorie „autonomen agile Teams“. Die Theorie stützt sich auf Interviews und Beobachtung von 19 Mitarbeitern einer Unternehmensgruppe. Um die Auswirkungen tiefer untersuchen zu können und die Theorie stärker stützen zu können, sind weitere Datenerhebungen notwendig.

Die Theorie erklärt wie ein Prozess das Fehlen einer formalen niedergelegten Definition von Done funktioniert und das Team somit zu einem autonom agilen Team wird. Das Prozessmodell strebt nach Entwickler, die nach eigenem Ermessen abwägen, wann eine Änderung als abgeschlossen gilt und wann nicht. Damit er diese Entscheidung treffen kann oder bei Bedarf die Hilfe bekommen kann, konzentrieren sich 5 der 12 Praktiken direkt auf die Kommunikation wie die nachfolgende Abbildung 6 zeigt:

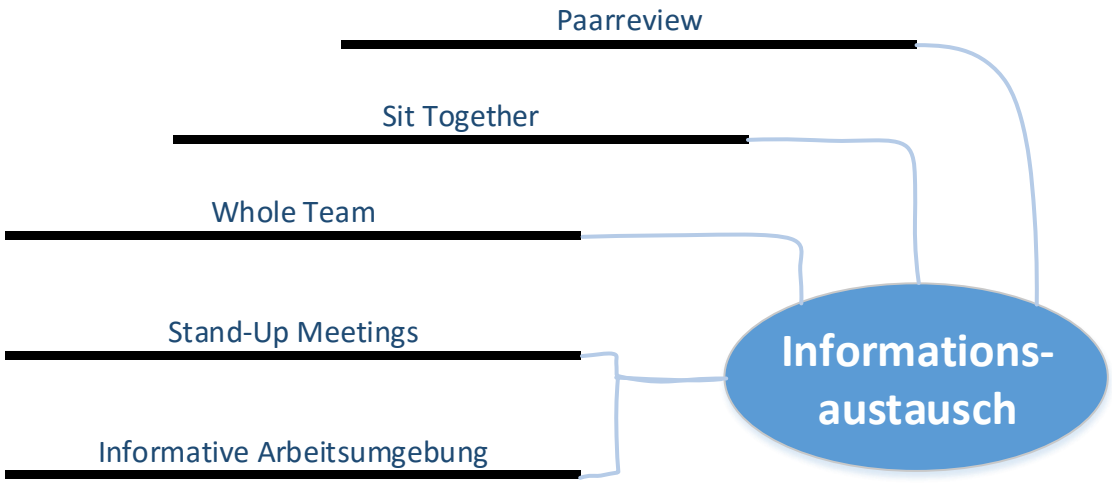

Abbildung 6: GT-Kategorie "Informationsaustausch"

Die Kommunikation hat für ein autonomes agiles Team zwei Bedeutungen. Zum einen fördert es ein gemeinsames Verständnis im Team, was besonders wichtig ist, da die Entscheidungsgewalt nun in den Händen jedes einzelne Mitglied liegt. Zusätzlich ist die Möglichkeit jederzeit zu fragen durch Sit Together und Whole Team gegeben. Die informative Arbeitsumgebung hilft den aktuellen Fortschritt zu halten und damit auch das Große Bild im Auge zu behalten. Durch zwei Stand-Up Meetings pro Tag gibt es einen Fortschritt- und Verständnisabgleich im Team (d.h. alle 4 Arbeitsstunden). Vor dem Einchecken und Uploaden erfolgt durch das Paarreview noch ein weitere Abgleich vom Verständnis.

Das Prozessmodell vertraut dem Einzelnen, der die Lage besser einschätzen kann als Prozessregeln. Daher kann der Entwickler nicht nur entscheiden, wann eine Änderungen als abgeschlossen gilt. Er kann zusätzlich das Paar-Review aussetzen und bestimmt die Testfälle.

Dieser Ansatz versucht die Entscheidungslast einer Person – dem Product Owner – umzulagern und zu verteilen auf das ganze Team. Neben eine gute Kommunikationsmöglichkeit ist schnelles Feedback ein wichtiger Aspekt, weshalb 5 der 12 Praktiken darauf fokussiert sind (vgl. Abbildung 7).

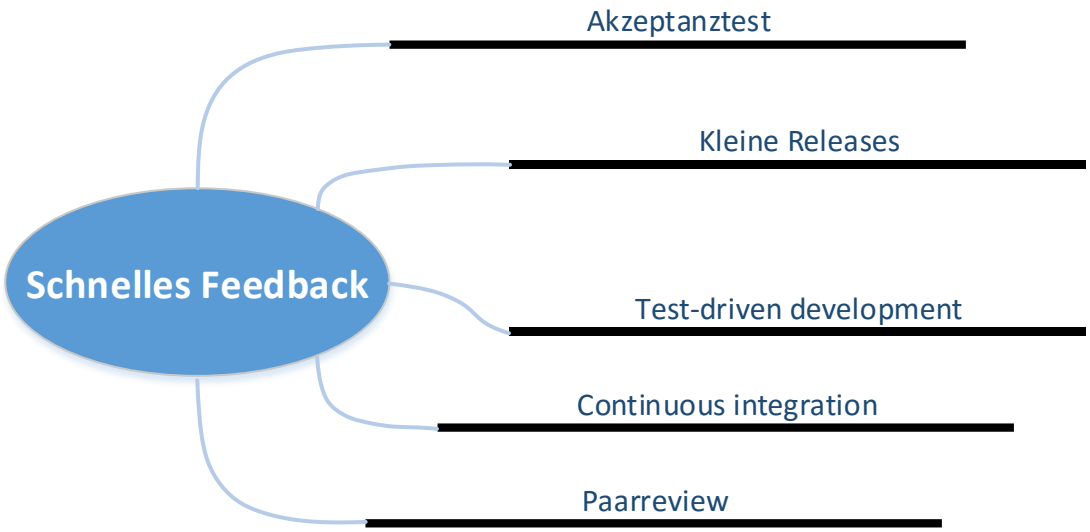

Abbildung 7: GT-Kategorie "Schnelles Feedback"

Continuous integration ermöglicht die Integration der neuen Änderung in das neue System. Zuvor werden die Tests wie Akzeptanztest und Unittest (Test-driven-development) automatisch durchgeführt. Der Entwickler kriegt somit frühzeitig vom System Feedback, ob seine Done Einschätzung richtig oder falsch war. Kleine Releases ermöglichen schnelle Feedbacks für das Team. Es bekommt Rückmeldung, ob das Epic tatsächlich Done ist oder nicht.

Von den 12 sind noch 3 ungeordnete Practices:

- Collective Code Ownership
- Stories
- Freiraum

Die 1. Praxis besagt, dass dem Team als Ganzem der Code gehört. Demnach ist es trivial, dass die Praxis die Theorie unterstützend ist.

Werden die beiden GT-Kategorien mit den beiden GT-Kategorie „Definition vons Done“ und „Autonome agile Teams entsteht die nachfolgende Abbildung 8.

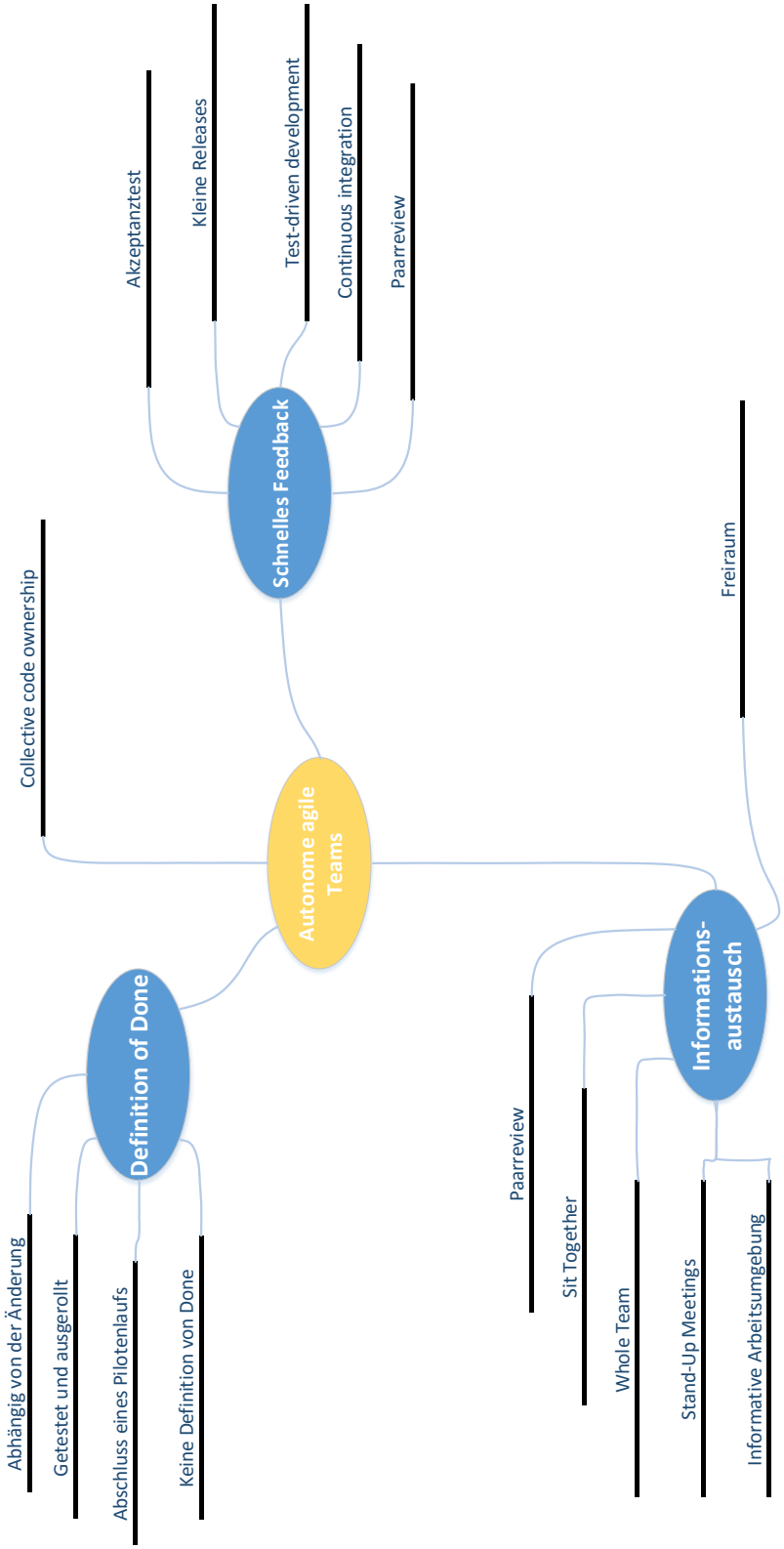

Abbildung 8: GT-Kategorie „Autonome agile Teams“

Aus den unterschiedlichen Aussagen, welche die GT-Kategorie „Definition of Done“ visualisiert, lässt sich ableiten, dass es unterschiedliche (gedankliche) Definitionen von Done gibt. Dies ist aber wenig überraschend, denn die Rollenverteilung Tester / Entwickler existiert noch. Die Tendenz ist hier aber, dass die Testerrolle in Zukunft abgeschafft wird.

Es wird danach aber dennoch weiterhin unterschiedliche Definitionen von Done geben, da die Rollenaufteilung zwischen Frontend- und Backendentwickler weiter existieren wird. Es ist auch nicht zwingend erforderlich, dass alle dieselbige Definition von Done haben, denn dann wäre das Prinzip der Diversity zerstört.

6.1 Auswertung GT-Theorie mit Agile Manifest.

Dieses Vertrauen in das Individuum entspricht dem ersten und letzten Leitsatz des Manifests der Agilen Softwareentwicklung

*„**Individuals and interactions** over processes and tools*
***Responding to change** over following a plan” [14]*

Interaction wird gefördert durch 2 Daily Stand-Ups Meetings und das Paar Review.

Die Definition von Done ist im Grunde eine Dokumentation und ein Festlegung. Würde man sie streng befolgen, müsste man sie auch für alle mögliche Situation definieren, so dass sie das beste Ergebnis liefert. Somit liegt ein Verstoß gegen die beiden Leitsätze 2 und 3 des Manifests der Agilen Softwareentwicklung vor.

*„**Working software** over comprehensive documentation*
***Customer collaboration** over contract negotiation” [14]*

Der Standardprozess sieht vor, dass der Entwickler nach der Fertigstellung der Aufgabe mit einem Kollegen darüber reflektiert und bereits Feedback bekommt. Anschließend übergibt er die Änderungen dem Testing, für welche er oder ein Tester Testfälle hinzugefügt hat. Hier kommt durch die Testfälle und ggf. den Tester ein zweites Feedback. Darüber hinaus gibt er Feedback an dem PO während der Daily Scrum Meetings.

Der Vorteil dieses Vorgehens ist:

- Der Entwickler kann besser (d.h. Situationsgerecht) agieren, als wenn er an einer formalen Definition von Done gebunden ist.

Nachteilig ist aber:

- Durch das höhere Vertrauen in den Entwickler auch ein höheres Risiko mit einhergeht. Aus Vergesslichkeit und Bequemlichkeit entstehen Fehler, dessen Prüfung er umgehen kann und keine Definition in daran erinnert.

Bemerkenswert ist auch, dass vor 4 Jahren eine Definition von Done auf der internen Wikiseite erstellt wurde. Seither hat sie niemand verändert und vermutlich ist ihre Existenz den meisten gar nicht bewusst.

6.2 Mischform von agilen Prozessen

Darüber hinaus ist auch auffällig, dass obwohl keiner der Befragten Extrem Programmierung als Prozessmodell angegeben hat, bereits 11 Agile Praktiken identifiziert werden konnten (vgl. Abbildung 9).

Der von den Befragten als Kanban benannte Prozess ist eine Mischung aus Scrum, Kanban und XP. Scrum wird zur Einleitung in Agile Projektarbeit genutzt. Anschließend von Kanban kombiniert mit XP Practices. Die Aussage würde von allen 5 befragten Teams bestätigt.

Es wird somit mit einem strikterem agilen Prozessmodell begonnen, das feste Meetings und Abläufe definiert und anschließend immer mehr aufweicht. Dabei werden alle Elemente von Scrums beibehalten, wie etwa die Sprint Retrospektive. Das Meta-Kanban-Board ist im Grunde das Product Backlog und das Sprint Backlog das Kanban-Board.

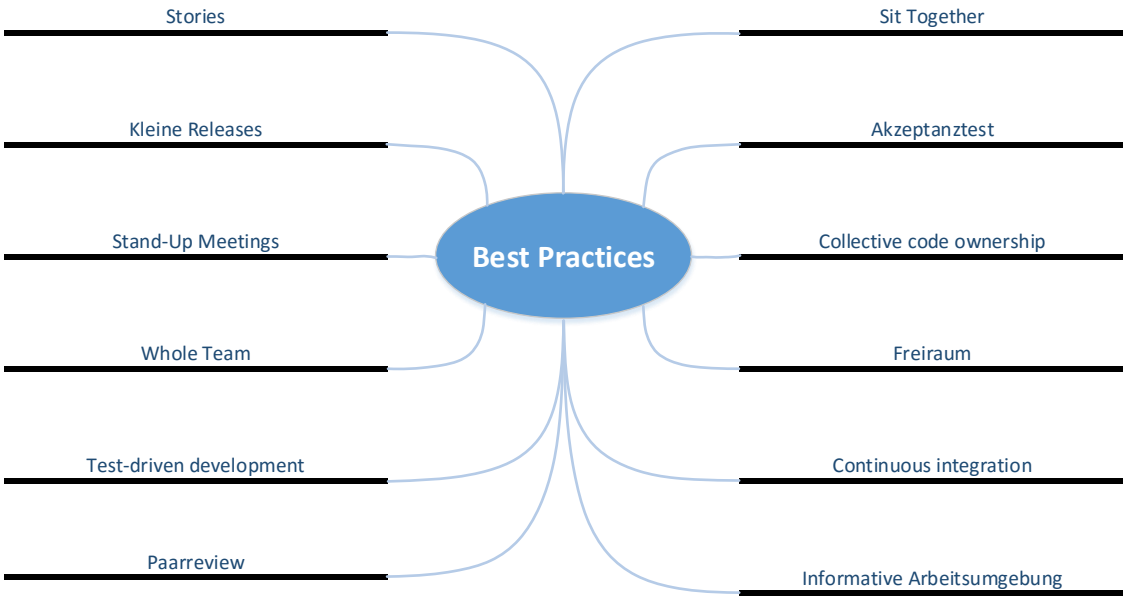

Abbildung 9: Agile Praktiken

6.3 Ausblick

Aufgrund der unvollendeten Datenerhebung wären offene Punkte die Untersuchung von weiteren Praktiken:

- Überstunden,
- Wöchentliche Zyklus und Vierteljährige Zyklus,
- Optimiere erst zum Schluss,
- Kodierungsstandards,
- Rikisominimierung durch „Spike Solutions“
- und Weitere.

Darüber hinaus ist eine vergleichende Untersuchung mit einem agil arbeitenden Unternehmen interessant, die eine Definition von Done vorlegen können. Werden da die Praktiken in der selben Intensität ausgeübt? Wie ist dort der Ist-Soll Unterschied?

Eine Analyse der Git Logs lässt Rückschlüsse über weitere Praktiken vermuten und weitere Einsichten gewinnen. Wenn beispielsweise zu einer Aufgabe über einen relativ längeren Zeitraum Code eingecheckt werden, handelt es sich vermutlich um Nachbesserung – Auswirkung des DoD.

Die Analyse der Stories auf Jira kann zu weiteren Einsichten in Auswirkungen des DoD führen. Beispielsweise können die User Stories anhand ihrer Beschreibung manuell in Kategorien eingeteilt werden:

- Rolle, Grund oder Aktion fehlen oder mangelhaft beschrieben,
- Weitere Informationen zur Abnahme enthalten
- Etc.

Nun können Daten erfasst werden wie: wie oft gab es Rückfragen zu dem Item, wie hoch sind die Git Aktivitäten bzgl. des Items und die durchschnittliche Umsetzungsdauer.

7 Literaturverzeichnis

[1] M. Ganis, D. Leip, F. Grossman und J. Bergin, „Introducing Agile Development (XP) into a Corporate Webmaster Environment - An Experience Report," *Agile Development Conference/Australasian Database Conference,* Nr. July 2005, pp. 145-152 , 2005.

[2] K. Beck und C. Andres, Extreme Programming Explained: Embrace Change, Boston: Addition Wesley, 2004.

[3] K. Beck und M. Fowler, Planning Extreme Programming (XP), Boston: Addison Wesley, 2000.

[4] D. Wells, „Extreme Programming," 8 10 2013. [Online]. Available: http://www.extremeprogramming.org/. [Zugriff am 31 5 2014].

[5] H. Wolf, Agile Projekte mit Scrum, XP und Kanban im Unternehmen durchführen: Erfahrungsberichte aus der Praxis, Heidelberg: dpunkt.verlag GmbH, 2012.

[6] K. Schwaber, Agiles Projektmanagement mit Scrum, Redmond, Washington: Microsoft Press, 2007.

[7] E. Hanser, Agile Prozesse: Von XP über Scrum bis MAP, Springer-Verlag Berlin Heidelberg, 2010.

[8] D. J. Anderson, Kanban, Washington: Blueholepress, 2010.

[9] F. Petrillo und M. Pimenta, „Is agility out there?: agile practices in game development," *Proceedings of the 28th ACM International Conference on Design of Communication,* Nr. September 2010, pp. 9-15, 2010.

[10] T. Smith, K. M. Cooper und C. S. Longstreet, „Software engineering senior design course: experiences with agile game development in a capstone project," *Proceedings of the 1st International Workshop on Games and Software Engineering,* Bd. 2011, pp. 9-12, 2011.

[11] P. Stacey und J. Nandhakumar, „Opening up to agile games development," *Communications of the ACM,* Bd. 51, Nr. 12, pp. 143-146, 2008.

[12] J.-G. Schneider und R. Vasa, „Agile Practices in Software Development - Experiences from Student Projects," *Software Engineering Conference, Australian,* Nr. April 2006 , pp. 401-410, 2006.

[13] G. P. Garvey und D. Tomczyk, „Agile game development with scrum in game lab I," *Journal of Computing Sciences in Colleges,* Bd. 28, Nr. 6, pp. 205-207, 2013.

[14] K. Beck, M. Beedle, A. v. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland und D. Thomas, „Agile Manifesto," 2001. [Online]. Available: http://agilemanifesto.org/. [Zugriff am 10 5 2014].

[15] T. Epping, Kanban für die Softwareentwicklung, Berlin Heidelberg: Sprinter-Verlag, 2011.

[16] B. G. Glaser und A. L. Strauss, Grounded Theory: Strategien qualitativer Forschung, Verlag Hans Huber, 2010.

[17] B. Glaser, Theoretical Sensitivity: Advances in the Methodology of Grounded Theory, Mill Valley, CA: Sociology Press, 1978.

[18] B. Glaser, The Grounded Theory Perspective III: Theoretical Coding, Mii Valley, CA: Social Press, 2005.

[19] R. Hoda, „Self-Organizing Agile Teams - A Grounded Theory," Wellington, 2011.

[20] B. Glaser, Basics of Grounded Theory Analysis: Emergence vs Forcing, Mill Valley, CA: Sociology Press, 1992.

[21] D. Batra, „Modified agile practices for outsourced software projects," *Communications of the ACM,* Bd. 52, Nr. 9, pp. 143-148, 2009.

[22] J. F. Abrantes und G. H. Travassos, „Common Agile Practices in Software Processes," *International Symposium on Empirical Software Engineering and Measurement,* Nr. September 2011, pp. 355-358, 2011.

[23] K. Korhonen, „Evaluating the Effect of Agile Methods on Software Defect Data and Defect Reporting Practices - A Case Study," *International Conference on the Quality of Information and Communications Technology,* Nr. October 2010, pp. 35-43, 2010.

[24] B. W. Boehm, „A spiral model of software development and enhancement.,“ *Computer,* Bd. 21, Nr. 5, pp. 61-72, 1988.

[25] P. Mayring, Einführung in die Qualitative Sozialforschung, Weinheim und Basel: 2002 Beltz Verlag, 2002.

[26] U. Flick, E. v. Kardorff und I. Steinke, Qualitative Forschung: Ein Handbuch, Rowohlt, 2005.