FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

SOFTWARE PROJEKT MANAGEMENT ein neuer Ansatz „BALANCING TEAM BUILDING“

MAGISTERARBEIT

zur Erlangung des akademischen Grades

Magister

im Rahmen des Studiums

INFORMATIKMANAGEMENT

eingereicht von

Arnold Belohlavek, Bakk.techn.
Matrikelnummer 9855621

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: Prof. Dr. Dipl.-Ing. Hilda Tellioğlu
Mitwirkung:

Wien, 15.10.2014 _____ _____

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

ERKLÄRUNG ZUR VERFASSUNG DER ARBEIT

Arnold Belohlavek Bakk.techn. , Am Südblick 3/2 3550 Langenlois Österreich

„Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit –einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe."

Ort, Datum, Unterschrift

KURZFASSUNG

Softwareerstellung in Projekten erfordert heutzutage ein hohes Maß an Projekt-Management. Softwareprojekte sind oft sehr komplex. Die Anforderungen an entwickelte Anwendungen sind sehr hoch in Bezug auf Liefertermine und Qualität. Gleichzeitig ergibt sich aber aus der Erfahrung öffentlicher Studien, dass Software-Projekte oft nur mit Verzögerungen, oder im schlimmsten Fall gar nicht fertiggestellt werden. Die Schäden durch Verzögerungen sind teilweise enorm. Menschen die in Software-Projekten arbeiten stehen oftmals unter hohem Druck. Um Software-Projekte termingerecht und in hoher Qualität fertigzustellen werden verschiedene Methoden des Projekt-Management angewandt. Die bestehenden Fakten zur Wertschöpfung aus Softwareprojekten spricht aber gegen eine Verbesserung der Situation. Projekte werden immer komplexer. Die Mitarbeitenden leiden unter Stress. Unternehmen bestehen oft aus komplexen Hierarchien. Neue Projekt-Management Methoden einzusetzen, fällt dadurch oft sehr schwer. Eingesetzte Projekt-Management Methoden erweisen sich als unzureichend. Manchmal werden aber gute Ansätze nicht korrekt umgesetzt. Jeder Projektmanagement-Methode ist eine Philosophie hinterlegt, die MitarbeiterInnen auch verstehen und leben müssen. Diese Arbeit beschäftigt sich mit der Frage, ob die bestehenden Methoden ausreichend sind und wie diese Methoden abgewandelt werden können, um leichter einsetzbar zu sein. Die Frage ist, ob es möglich ist mit einem einfachen Set an Tools ein Gerüst aufzubauen, das den modernen Entwicklungsmethoden gerecht wird. Dazu wurden die aktuellen Softwareprojekt-Management Methoden betrachtet und analysiert. Um die momentane Situation in Software-Projekten in Hinsicht auf das Projektmanagement und die Erfolgsquote beurteilen zu können, wurde ein Online-Fragebogen erstellt. Es wurde mit einem repräsentativen Querschnitt von MitarbeiterInnen aus der Software-Branche eine Befragung durchgeführt. Mittels dem Statistik-Programm Gnu-R wurden die Auswertungen der gewonnen Daten durchgeführt. Als Ergebnis ist ein Bild entstehend wie die MitarbeiterInnen die momentane Lage einschätzen. Die Fragestellungen sind einerseits die Abfrage von fachlichen Dingen aber andererseits auch die Abfrage von sozialen Faktoren und emotionalen Skills wie auch der Sicht auf Projekterfolge und momentane Arbeitsweise. Aus der Analyse der bestehenden Projektmanagement Methoden sowie eigenen Erfahrungen und der Informationen aus der Literatur-Recherche wurde versucht, die besten Ansätze für ein funktionierendes Projekt-Management abzuleiten. Und das immer mit dem Blick auf die aus der Recherche bekannte Erfolgsfaktoren. Es ist daraus eine Beschreibung für einen Ansatz entstanden der leicht umzusetzen ist. Um diesen Ansatz vollumfänglich zu beschreiben, war es notwendig, neue oder abgewandelte Rollen des Projektmanagements zu definieren. Es wurden auch die nötigen Hilfsmittel zur Visualisierung und zur Messung der Qualität oder Leistung beschrieben und die Gründe dafür genannt. Es wurde auch ein Einsatzplan entwickelt und die Erklärung dafür, um optimale Ergebnisse im Falle einer Umsetzung zu garantieren.

ABSTRACT

Software development in projects requires a high level of project management today. Software projects are often very complex. The requirements for advanced applications are very high in terms of delivery and quality. At the same time facts are published, that software projects are often not completed and if then with delay or in the worst case get never completed. The damage caused by delays are partially enormous. People are working in software projects often under high pressure. In order to complete software projects on time and in high quality, various methods of project management are applied. However, the existing facts to the value of software projects speaks to no improvement in this situation. Projects are becoming increasingly complex. Employees suffering from stress. Companies often consist of complex hierarchies. It is often very difficult to use new project management methods. Applied project management methods prove to be insufficient. Sometimes good ideas are not implemented correctly. Each project management methodology is a philosophy deposited, which the employees have to understand and have to live. This work deals with the question whether the existing methods are adequate and how these methods can be modified to be easier to use. The question is whether it is possible with a simple set of tools to build a framework, that the modern development methods meet. Current software project management methods were considered and analyzed. To view the current situation in software projects in terms of project management and to assess the success rate, an online questionnaire was created. A survey with a representative sample of employees from the software industry was carried out. The statistical software GNU R was used to perform the analysis. As a result, a figure was created, how the employees assess the current situation. The questions are, on the one hand, the query of technical things but, on the other hand, the query of social factors and emotional skills as well as the view of project achievements and current work. From the analysis of the existing project management methods as well as from own experience and the information from the literature research the best approaches for an efficient project management has been tried to derive. This always with the view on success factors from the research. There was a description of a developed approach which is easy to implement. To describe this approach fully, it was necessary to define new or modified roles of project management. The result was also a description for the necessary tools for visualization and for measuring the quality or performance and the resulting motivation therefore. An operational plan was developed to ensure the explanation for optimal results while implementation.

Inhalt

1 EINLEITUNG....1

2 RECHERCHE UND THEORIE....2

2.1 Grundlagen....2

2.2 Überblick über bestehende Projektmanagement-Methoden....5

2.2.1 Klassische Vorgehensmodelle....6

2.2.2 Agile Methoden....8

2.2.3 Visualisierung und Metriken....20

2.3 Menschen in Projekten....26

2.4 Recruiting....27

2.5 Das Team....29

2.6 Geschäftsführung und Management....30

3 EIN NEUER ANSATZ AUF BASIS „BALANCING TEAM BUILDING“....32

3.1 Grundlagen und Motivation....32

3.2 Eigene Erhebungen....34

3.2.1 Online Fragebogen....34

3.2.2 Statistische Auswertungen....34

3.3 Das Gerüst....47

3.3.1 Kontinuierliche Lieferung an den Kunden....47

3.3.2 Anforderungsänderungen begegnen....47

3.3.3 Iterationen und regelmäßige Lieferungen....47

3.3.4 Schnittstellen zum Kunden....48

3.3.5 Selbstorganisierte Teams....48

3.3.6 Kommunikation....48

3.3.7 Funktion und Qualität....49

3.3.8 Work Life Balance....49

3.3.9 Fachliche Qualifikation und Weiterbildung....49

3.3.10 „Keep it simple“, oder die richtige Wahl des Teams....50

3.3.11 Gleichberechtigung in Teams....50

3.3.12 Selbstreflexion....50

3.4 Die Rollen....51

3.4.1 Request Manager....51

3.4.2 Team Coach....52

3.4.3 Global Architect....52

3.4.4 Toolsmith....53

3.4.5 Team....53

3.4.6 Kunde....54

3.5 Die Hilfsmittel....55

3.5.1 Backlog....55

3.5.2 Visualisierung....55

3.5.3 Cumulative Flow Diagram und mittlere Durchlaufzeit....56

3.5.4 Eine Beschreibung für den Einsatz....57

4 ANALYSE....61

4.1 Evaluierung....67

5 ANHANG....I

5.1 Anhang I: Fragebogen....I

6 LITERATURVERZEICHNIS....VII

Verzeichnis der Abbildungen

Abb. 1: Quelle: http://de.wikipedia.org/wiki/Wasserfallmodell.....6
Abb. 2: Quelle: http://de.wikipedia.org/wiki/V-Modell.....8
Abb. 3: Quelle http://www.agilemodeling.com/essays/fdd.htm.....16
Abb. 4: Quelle:http://leankit.com/blog/2010/12/10-kanban-boards-leankit-kanban-style/.....20
Abb. 5: Quelle: http://www.mountaingoatsoftware.com/agile/scrum/task-boards.....21
Abb. 6: Cumulative Flow Diagram, eigene Darstellung.....23
Abb. 7: Average Cycle Times, eigene Darstellung.....24
Abb. 8: Efficiency Diagram, eigene Darstellung.....24
Abb. 9: Kategorie Befragter.....36
Abb. 10: Altersgruppe Befragter.....36
Abb. 11: Größenordnungen im Projekt.....37
Abb. 12: Auswertung Projekterfolg.....38
Abb. 13: Auswertung Skills.....39
Abb. 14: Aussagen über Projekt-Management.....40
Abb. 15: Grundwissen über agile Methoden.....41
Abb. 16: Empfindung in Bezug auf agile Methoden.....42
Abb. 17: Erfahrung in Projekten.....43
Abb. 18: Erfahrung im Team.....45
Abb. 19: Kanban-Board Beispiel,eigene Entwicklung.....55

Verzeichnis der Abkürzungen

WIP	Work in progress
XP	Extreme Programming
PM	Projektmanagement
PO	Product Owner
SM	Scrum Master
TDD	Test driven development
FDD	Feature driven Development
ASD	Adaptive Software Development
RM	Request Manager
TC	Team Coach
GA	Global Architekt
TS	Toolsmith

1 Einleitung

Diese Arbeit beschäftigt sich mit den Methoden des Software-Projektmanagement und den Vor/Nachteilen bestehender Methoden oder Vorgehensweisen. Es wird versucht einen Lösungsansatz für die genannten Problemfelder zu finden und die bestehenden Softwareprojektmanagement-Methoden zu kombinieren und zu erweitern um die fehlenden Elemente zu ergänzen. Die fehlenden Elemente sind nach Ansicht dieser Arbeit jedoch nicht unbedingt in neuen Methoden oder Mechanismen zur Kontrolle (Zeit, Quantität) zu finden, sondern in der Gestaltung der Projekthierarchie und der Projektteams auf Basis ausbalancierter Elemente. Es ist der momentane Ansatz gegeben, dass ProjektMitarbeiterInnen auf Basis von Selektionen gefunden werden, welche in erster Linie auf die benötigten Skills abzielen. Die soziale Kompetenz scheint im Bereich Recruiting bestenfalls minimal behandelt zu werden. Dies hat ergeben, dass Recruiting in der heutigen Form mehr Profillesen ist als die gesunde Selektion der einzelnen MitarbeiterInnen auf Basis Ihrer gesamten Qualitäten. Da Softwareprojekte ein gut zu verkaufendes Gut geworden sind, haben sich mehrere Hierarchien von Lieferanten von Human-Resources entwickelt. Dies geht soweit, dass Projekte mehrere Ebenen von Lieferanten haben, um das Risiko möglichst klein zu halten. Ein Ansatz ist der des Core-Supplier, welcher das Recruiting nach seinen Möglichkeiten und über Sublieferanten vornimmt. Bis MitarbeiterInnen in einem Projekt arbeitet, gibt es zwei oder mehr Ebenen, die zwischen ihm und dem eigentlichen Projektauftraggeber stehen. Dies führt dazu, dass in den einzelnen Ebenen unterschiedliche Interessen der einzelnen Parteien bestehen. Diese Interessen sind meist rein umsatzgesteuert und dienen nicht der Nachhaltigkeit der Entwicklung von Human-Resources, sondern eher dem Prinzip des quantitativen Recruiting, um möglichst schnell Lücken füllen zu können. Alleine diese Darstellung zeigt schon auf, dass es die Ware ProjektMitarbeiterInnen gibt und nicht den Menschen mit all seinen Fähigkeiten, aber auch Bedürfnissen. Genau diese Betrachtungsweise scheint ein wesentlicher Faktor im Scheitern von Softwareprojekten zu sein. Es scheint viel sinnvoller zu sein sich um nachhaltige Personalentwicklung zu bemühen.

Diese Arbeit will diesem Thema Rechnung tragen und herausarbeiten, dass mit einer weiterreichenden Methodik zur Auswahl und Gestaltung von MitarbeiterInnen und deren Rollen in Projekten der Wirkungsgrad der Projektteams signifikant erhöht werden kann.

Es soll gezeigt werden, dass das Ausbalancieren der Teams unter Betrachtung deren emotionaler und psychischer Möglichkeiten sowie der gezielten didaktischen Unterstützung zusätzlich zu ihren technischen Fähigkeiten einen erheblichen Zuwachs an Leistung bringt. Als Ansatz dient die Betrachtung der Reibungsverluste auf Grund zu hoher Komplexität durch Projekt(Recruiting)-Strukturen und die damit eingehende Unsicherheit über wichtige Lebensthemen der Menschen und zukünftigen ProjektMitarbeiterInnen (Stichworte: Sicherheit, Freude, WORK-LIFE Balance, Gesundheit, BURN-OUT).

Diese Arbeit möchte zeigen, dass es einfache Möglichkeiten gibt, um die Zufriedenheit und Stärke der Teams zu steigern und daraus resultierend ein automatischer Leistungszuwachs entsteht. Dazu sollen Methoden herangezogen

werden, um ein zusätzliches Set an Tools in das Projektmanagement zu bringen. Es ist das Ziel dieser Arbeit, die agilen Softwareprojektmanagement Methoden zu hinterfragen und entweder zu ersetzen oder um die notwendigen Elemente und Methoden und auch Rollen zu erweitern.

2 Recherche und Theorie

Es wurden Literatur-Recherchen, als auch Recherchen in öffentlichen Reports durchgeführt um die Software-Produktion in Ihrer Qualität beurteilen zu können. Es wird ein Überblick über bestehende Projekt-Management Methoden gegeben. Diese Linie teilt sich in klassische und agile Methoden. Vor allem in den agilen Projekt-Management Methoden gibt es ein breites Feld. Grundlage zu den agilen Methoden ist das agile Manifest und die agilen Prinzipien. Aus den verschiedenen Methoden leiten sich Erfolgsfaktoren ab. Um Projekte erfolgreich abwickeln zu können sind auch weitere Hilfsmittel, wie Metriken oder Visualisierung von Prozessen, von Bedeutung. Weiters ist es von besonderer Bedeutung Menschen in unterschiedlichen Funktionen in Software-Projekten näher zu betrachten. Es gilt also, die handelnden Personen und Ihre Interaktion in den Teams, im Management und die Geschäftsführung näher zu betrachten.

2.1 Grundlagen

Unzulänglichkeiten bei der Projektabwicklung scheinen das Problem Nummer 1 zu sein, warum Softwareprojekte scheitern oder ein massiver Wert-Vernichter sind. In [12] sind Daten der erfolgreichen oder nicht erfolgreichen Projekte erhoben worden.
Zitat [12, S.6] *„Verteilung des Projekterfolges: 45,2% der Projekte wurden als erfolgreich bewertet. Bei 19,4% konnten die Ziele nur mit Zeit- und Kostenüberschreitung erreicht werden. Bei den übrigen 35,4% wurden trotz Zeit- und Kostenüberschreitung die Ziele nur partiell oder überhaupt nicht erreicht. „*
Zitat[12, S.10]:

> *„Fazit*
> *Die genauere Betrachtung der Umfrageergebnisse legt folgende Aussagen nahe:*
> *Projekte scheitern nicht an der Technik.*
> *Projekte scheitern an fehlender Information und Kommunikation.*
> *Projekte scheitern an fehlendem Vertrauen im Projektteam.*
> *In Projekt-Teams mit mehr als 4 Mitgliedern sinkt die Erfolgsquote nochmals drastisch.*
> *Die gruppendynamische Prozesse stellen hier offensichtlich ein zusätzliches Risiko dar.*
> *Die Auswahl des Vorgehensmodells hat keinen Einfluss auf den Projekterfolg.“*

Es lässt sich ein Zusammenhang zwischen Technik und Vorgehen (Projektmanagement – Methode) weder positiv noch negativ ableiten. Laut [12, S.8] scheinen eher agile Projekt-Management-Methoden eine bessere Erfolgsquote zu zeigen. Ebenfalls scheint die Aussage „Arbeitszeit unter 50 Stunden“ bei erfolgreichen Projekten mehr Zustimmung zu bekommen. Es scheint also Vorteile für Projektmanagement-Methoden zu geben die einen agilen iterativen Ansatz wählen. Im Chaos-Report der Standish-Group [16] werden die Zahlen gescheiterter Projekte noch größer angesehen.
Laut dem Chaos-Report der Standish-Group [16, S.1, Daten von 2003-2012 aus der Chaos-Datenbank] werden nur 6% der Projekte als erfolgreich eingestuft. 52% der

Projekte werden mit Mehrkosten und erhöhten Aufwänden fertiggestellt. 42% der Projekte scheitern total.

Wie auch immer, scheint ein großer Teil der Softwareprojekte zu scheitern oder mit massiven Mehrkosten konfrontiert zu sein. [2, S.24] erweist auf eine Studie der Info-Tech Research Group die abschätzt, dass der monetäre Verlust durch gescheiterte Projekte in den USA um die 130 Mrd. Euro und in Europa um die 120 Mrd. Euro betragen.

Wie auch immer diese Zahlen bewertet werden, es scheint Handlungsbedarf gegeben zu sein. Befragte Institutionen sowie Manager aus diesen Bereichen meinen, dass ein großer Teil dieses Problems im fehlenden oder fehlerhaften Projektmanagement sowie dessen Methoden liegt.

Andere meinen, dass unklare Zielsetzungen das Problem darstellen, weitere Meinungen führen die Komplexitätsgrade als Problem an. Dabei werden das Projektteam und deren Leitung sowie die Team-Mitglieder gerne rein als Ressource betrachtet, welche auf Basis einer Personentag-Schätzung für die zu erwartenden Aufwände eine reine Kostenstelle darstellen. Es wurden Methoden erfunden, um Softwareprojekte erfolgreich meistern zu können, allen voran die agilen Softwareprojektmanagement-Methoden (siehe [1], [4], [6]). Diese Methoden zielen alle darauf ab, um erstens eine Basis für das Recruiting, zweitens eine Basis für Aufwandsschätzungen und drittens im Laufe des Projektes eine Kontrolle der Teilergebnisse und eine Bewertung des Fortschrittes zu haben. Probleme sollen möglichst früh erkannt werden, um dagegen steuern zu können. Trotz aller Hilfsmittel in diesen Bereichen scheint es trotzdem an Etwas zu fehlen, ansonsten wären die Anzahl der gescheiterten Projekte und deren Schaden nicht so hoch.

All diesen Methoden anheim ist das große Bestreben des Managements nach Sicherheit bzw. Risikoabschätzung (und der Versuch der Optimierung, siehe [1, S.57]. So werden an sich gut ausgedachte Methoden sehr leicht verbogen um den Kontrollen mehr Zahlen liefern zu können. Es stellt sich die Frage ob mehr Zahlen mehr Sicherheit bedeuten. Dies beginnt bei der Schätzung von Aufwänden, wo aus Verkaufsgründen oftmals sehr gerne im Sinne des Kunden optimistisch geschätzt wird [1, S.67]. Diese Schätzungen entbehren jedoch oft jeglicher vernünftiger Grundlage, bringen aber sehr schnell Teams (v.a. agile Teams) in die Situation das Teilergebnisse nicht zeitgerecht fertiggestellt werden können und dadurch Projekte sofort ins Hintertreffen geraten und der erhöhte Stressgrad der Teams bis zum Projektende nicht mehr abnimmt. Was daraus resultiert ist ein fehlerhafter Code welcher oftmals in langer Nacharbeit mit hohen Aufwänden korrigiert werden muss.

[5, S.29] beschäftigt sich mit der agilen Softwareentwicklung in Großprojekten und sieht den häufigsten Grund für negative Entwicklungen in mangelnder Kommunikation. Ebenfalls spielt die Größe eines Projektes eine entscheidende Rolle, wenn man Kommunikation betrachtet. Es wird ebenfalls der Umstand betrachtet das gerade bei agilen Entwicklungsprozessen die gewählte Methode (oftmals SCRUM) nicht zu allen Projekten (den handelnden Personen) passt [5, S.30]. Außerdem läge die Entscheidung über Verlauf und Umfang beim Kunden, und das genau dort auf Kundenseite viel gut oder schlecht gemacht werden kann.

[13, S.7] sieht das klassische Projektmanagement zu technisch orientiert, was oftmals nicht zum Erfolg führt. Außerdem sei der Faktor Mensch nicht ausreichend berücksichtigt und es kranke meistens an der Kommunikation.

Die Standish-Group [16] in Ihren Reporten sieht die Fehlentwicklung (Stressfaktoren) folgendermaßen, Zitat[16, S.3]:

> *„COMMON STRESS FACTORS*
> *There were many more both challenged and failed projects to consider, therefore there was a more diverse set of stress factors.*
> *One factor that was very common is optimal team size. Many of the organizations had large teams.*
> *Second, the project teams did not manage or optimize scope, which in this case is moot since the scope is well defined and mandatory.*
> *Third, users were not engaged and those who were did not know their subject matter. Many of the project teams did not involve the users and demonstrations were held sparingly.*
> *Fourth, many of the organizations had poor emotional maturity skills.*
> *The Standish Group has outlined 50 skills organizations need to achieve emotional maturity.*
> *These skills include managing expectations, gaining consensus, and reflective listening.*
> *These 50 skills are outlined in our Emotional Maturity Research Report. Last, many of the projects were the "Big Bang" type and did not use an iterative process. „*

Aus diesen Informationen lässt sich ableiten, dass offensichtlich im Softwareentwicklungsbereich das Recruiting falsche Ansätze hat, d.h. auf die Bedürfnisse und Fähigkeiten (Möglichkeiten) der Handelnden zuwenig eingegangen wird.

Die Teams waren zu groß. Das Ergebnis waren schlechte Projekterfolge. Die Standish-Group beziffert in [16, S.3]: Erfolgreiche Projekte hatten 61% emotional reife Persönlichkeiten an Board, wogegen Projekte die nicht oder nur durch Verzögerungen fertiggestellt wurden lediglich 19% emotional reife Persönlichkeiten an Board hatten. Dies entspricht auch obigen Aussagen von [12, S.10] und [5, S.2], welche schlechte Kommunikation und Teamfähigkeit als die Hauptfehlerquellen in Projekten einstufen.

Um erfolgreich Projekte meistern zu können, merkt der Report der Standish-Group wie folgt an, Zitat[16, S.5]:

> *„COMMON SUCCESS FACTORS*
> *There were many successful projects in our database that matched the migration and integration profile.*
> *One of the most prevalent success factors affecting this outcome is a highly competent technical team member.*
> *This is a very important factor for projects of this type and style, since they rely heavily on people who know the new system, know the technology, and know how the existing interfaces and databases work.*
> *Second, the project manager had basic PM skills and encouraged a sense of ownership for team members.*
> *Third, the project team used prototypes with lots of user feedback. Fourth, progress tracking was transparent with minimal use of project management tools, processes, and governance.*
> *Last, executive sponsors were inspiring.*

In other words, for success you need:
1) A highly skilled executive sponsor;
2) A highly competent team;
3) A good project manager;
4) Prototypes with fast user feedback;
and 5) Transparent progress tracking.“

Möchte man von diesen Aussagen ableiten ergibt sich ein Bild eines Softwareprojekt-Managements welches auf optimale Teambildung fokussiert. Es sind in diesem Bereich vor allem die Recruiting-Ansätze der Gegenwart zu hinterfragen. Es ist zu hinterfragen was ein hochkompetentes Team ist. Welche Parameter sollen erfüllt sein. Nur technische oder fachliche Qualifikation oder emotionale Intelligenz. Rasches Feedback durch Prototypen spricht eindeutig für die agilen Ansätze [3, S.25]. Die Rolle des Projekt-Manager wird eindeutig hervorgehoben. In den verschiedenen Ansätzen des Projekt-Managements wird die Rolle des Projektmanagers oft verändert oder einer anderen Rolle zugesprochen. Auf diesen Umstand ist ebenfalls besonders Rücksicht zu nehmen. Vor allem die agilen Methoden haben oftmals die Rolle eines Projektmanagers aufgeteilt in verschiedene Rollen [3, S.39]. Transparenz kann dadurch entstehen, indem die richtigen Hilfsmittel verwendet werden um dem Kunden aber auch dem gesamten Projekt-Teams Informationen leicht ersichtlich darzustellen. Es gibt in diesem Bereich viele elektronischen Hilfsmittel [21] sowie physische Boards in verschiedenen Ausprägungen [10, S.73]. Alle diese Dinge funktionieren dann am Besten, wenn Sie einfach und optimal (ohne unnötigen Overhead) eingesetzt werden. Die Rolle des Kunden wird hier nicht explizit angesprochen, aber verpackt sich in die Information „user feedback“ als Erfolgsfaktor. Kunden sind die Herausgeber der Anforderungen. Sie entscheiden über Art und Umfang der Umsetzung. Die Schnittstelle Kunde zu Entwicklung ist von besonderer Bedeutung. Kunden müssen durch Transparenz (Visualisierung und iterative Auslieferung von Prototypen) [10, S.71,99] in den gesamten Entwicklungsprozess integriert sein. Das daraus resultierende Feedback [10, S.172] ist von enormer Bedeutung für eine beschleunigte Entwicklung von Software. Vor allem die „Waste“-Time durch die Umsetzung falscher oder nicht vollständiger Anforderungen kann nur so minimiert werden.

2.2 Überblick über bestehende Projektmanagement-Methoden

In einem Softwareprojekt werden normalerweise mehr oder weniger komplexe Geschäftsanwendungen entwickelt, die das harmonische Zusammenspiel aller beteiligten Kräfte benötigt. Es sind meist mehrere Software-Entwickler, Tester, Personen aus dem Fachbereich, Vertreter des Kunden, Projektmanager, Personen aus dem Management, Administratoren und Andere in einem Projekt als Team oder Teams vereint.

Die strategische Abwicklung unter möglichst geringen Reibungsverlusten sowie Einhaltung der vereinbarten Termine sind das Ziel des Projektmanagements. Aus der Geschichte heraus haben sich verschiedene Methoden entwickelt, welche zum Ziel hatten die Projekte erfolgreich umzusetzen. Die „klassischen“ Modelle [4, S.13] sind

Vorgehensmodelle, welche sich prozessorientiert gestalten. Der Entwicklungsprozess wird in Phasen aufgeteilt, die auch öfter durchlaufen werden können, aber eher starr geplant sind. Die Entwicklungszyklen sind meistens länger als bei den neueren Methoden. Entwicklung, Tests und Benutzerabnahme wurden meist in verschiedene Phasen aufgeteilt. Kunden haben nicht die Möglichkeit Software in einem sehr frühen Stadium zu sehen und begutachten zu können.

Die zweite etwas neuere Richtung sind die „agilen" Methoden [4, S.39]. Sie basieren auf der Auslieferung von Software in iterativen Schritten. Daraus ergeben sich mehrere Vorteile. Die Anforderungen werden in kleine kontinuierliche Pakete gefasst und permanent in kleinen Fortschritten an den Kunden ausgeliefert. Kunden können in einem sehr frühen Stadium ihre (wachsenden) Produkte begutachten. Wenn Fehler in der Anforderung oder Änderungen erkannt werden, kann relativ früh darauf reagiert werden. Agile Methoden tragen bereits in der Bezeichnung die schnellere Reaktionsmöglichkeit. Um jedoch die Vorteile dieser Methoden zu nutzen ist es erforderlich das sich MitarbeiterInnen auf gewisse Rahmenbedingungen einstellen. Meist haben diese neueren Methoden eine intensive Rollenverteilung, welche sich klar von den klassischen Verfahren abhebt. Es wurden zur Darstellung der bestehenden Methoden Informationen aus eigener Erfahrung und aus der Recherche aus [1] [4] [5] [6] [7] und [10] herangezogen.

2.2.1 Klassische Vorgehensmodelle

WASSERFALLMODELL

Dieses Modell [4, S.16] gliedert sich in Phasen, welche der Reihe nach abgearbeitet werden. Ein Beispiel: Anforderungen - Systemanalyse - Systementwurf - Implementierung - Test & Integration – Abnahme.

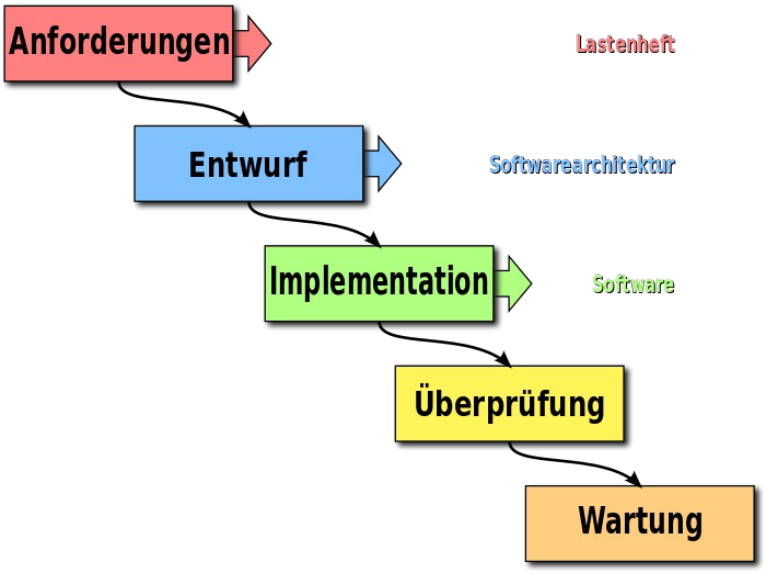


Abb. 1: Quelle: http://de.wikipedia.org/wiki/Wasserfallmodell

Grundlage dieses Modells ist es, dass erst nach erfolgter Qualitätsprüfung und Freigabe der vorhergehenden Phase die nächste Phase begonnen wird. Phasen werden in einem straffen Zeitplan eingeplant. Es leidet darunter die Flexibilität. Die Möglichkeiten um Änderungen einzufügen sind minimal. Der Umfang der einzelnen Phasen liegt im großen und ganzen fest. Diese Art des Projekt-Managements ist geeignet für Projekte die einen überschaubaren Rahmen haben und die Anforderungen klar sind und fest stehen.

Eine Erweiterung ist das Wasserfallmodell nach Royce [4, S.16]. Diese Erweiterung ermöglicht die Rückkopplung in frühere Phasen, welche dazu dienen kann um Anforderungsänderungen oder Fehler in der jeweiligen Phase behandeln zu können. Es sind auch Rücksprünge in frühere Phasen möglich. Trotz dieser Erweiterung ist dieses Modell eher für einfache Projekte geeignet inklusive der Möglichkeit bedingt auf Änderungen eingehen zu können.

Das Wasserfall-Modell ist ein etwas älteres Modell bzgl. Vorgehensweise in Projekten. Grundsätzlich versucht diese Modell in der Planung Zeiträume der Umsatzung der einzelnen Phasen vorherzusagen. Nicht ausreichend berücksichtigt ist jedoch der Umstand das Anforderungsänderungen auftreten können. Dies Änderungen lassen sich nur schwer im Stufen-Zyklus integrieren. Zeitpläne können sehr schnell nicht mehr eingehalten werden, bzw. neue Anforderungen oder Änderungen bis zum Release nicht mehr aufgenommen werden. Es ist eine sehr starre Vorgehensweise. Es gibt jedoch Bereiche in sicherheitskritischen Bereichen der Softwareentwicklung (in überschaubaren Größenordnungen) wo sich dieses Modell rechtfertigen lässt. Dort dient der Charakter der einzelnen Stufen inklusive abgeschlossener Qualitätssicherung und jeweiliger Abnahme der Sicherheit. In Projekten mit größerem Umfang ist dieses Modell auf Grund mangelnder Flexibilität weniger geeignet. Es lässt sich auch schlecht skalieren, vor allem in Bezug auf das Parallelisieren von Aufgaben.

V – MODELL

Das V-Modell [4, S.19] ist eine Erweiterung des Wasserfall-Modells mit dem Zusatz das frühere Phasen über Testphasen mit den späteren Phasen verbunden sind.

Das V-Modell wurde von Barry Boehm 1979 aus dem Wasserfallmodell entwickelt. Es ist typischer Weise so aufgebaut das die jeweiligen Anforderungs/Umsetzungs-Ebenen einer spezifizierten Qualitätssicherungsebene gegenüber steht. Interessant hierbei ist bereits die Unterteilung der verschiedenen Testebenen bis hin zu den Akzeptanz-Tests.

Es handelt sich hier ebenfalls um sequentielle Abarbeitung der einzelnen Schritte (inklusive Ihrer Tests). Auf der untersten Ebene beginnen Entwickler ihre Unit-Tests ablaufen zu lassen. Integration und System-Integration sind nachgeordnete Schritte ebenso wie die Akzeptanz-Tests. Auch hier ist deutlich die Abgrenzung zu neueren Methoden gegeben , welche im Bereich Test und Abnahme viel früher ansetzen. Klar ist im V-Modell die klare zeitliche Abgrenzung zwischen den einzelnen Bearbeitungsschritten.

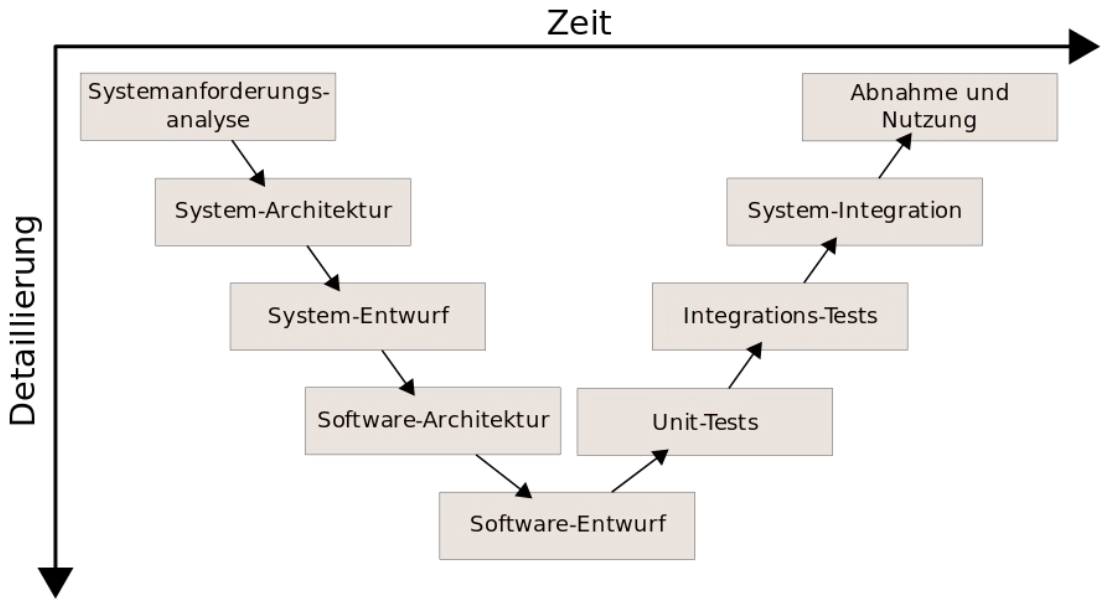


Abb. 2: Quelle: http://de.wikipedia.org/wiki/V-Modell

2.2.2 Agile Methoden

Agile Methoden [4, S.39] sind der neuere Ansatz von Software-Projektmanagement. Sie basieren auf dem Grundgedanken, das es in der Anwendungsentwicklung normalerweise immer zu Anforderungsänderungen kommt sei es aus rechtlichen oder firmenpolitischen Gründen, oder einfach deswegen das Software-Projekte oftmals sehr komplex sind und es nicht einfach ist den gesamten Umfang und seine Aufwände komplett vorherzusagen. Es gibt viele Faktoren die den Aufwand der Umsetzung von Projekten erhöhen aber auch vermindern können. Wie in der grundlegenden Recherche in 2.1 genannt ist es selten die technische Qualifikation die den Misserfolg von Projekten begründet. Es sind eher Kommunikation und zwischenmenschliche Faktoren die zum Erfolg oder auch Misserfolg beitragen.

Um diesem Umstand Rechnung zu tragen hat es sich eine Gruppe erfahrener Spezialisten aus dem Bereich der Software-Entwicklung zum Ziel gesetzt zu erheben, was positive Einflüsse auf Software-Projekte haben kann. Daraus ist eine Idee oder Gedankengut entstanden welches als Rahmen ([17] sowie [20]) gefasst wurde. Diesen Rahmen verwenden alle agilen Methoden als philosophischen Grundbaustein, um darauf aufzubauen und verschiedene Mittel zu entwickeln um das Projektmanagement und die Projekte effizient umzusetzen.

DAS AGILE MANIFEST

Zitat aus [17]:

> *„We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> Individuals and interactions *over processes and tools*
> Working software *over comprehensive documentation*
> Customer collaboration *over contract negotiation*
> Responding to change *over following a plan*
>
> *That is, while there is value in the items on the right, we value the items on the left more.***“**

Diese Gedanken bilden die Grundlage für die agilen Methoden im Software-Projektmanagement. Sie sind als Vereinbarung entstanden welche eine Gruppe von Menschen definiert hat, welche in der Softwareentwicklung tätig sind und die Problemfelder erkannt haben. Die Liste der Unterzeichner dieser Werte umfasst mittlerweile tausende Personen und wächst weiter. Sie definieren ein völlig anderes Bild als die klassischen Verfahren. Anstatt Prozessorientiert zu planen, in fixen Schritten und mit starren Terminplänen, versuchen die agilen Methoden in Iterationen Software kontinuierlich weiterzuentwickeln. Dazu werden Zeiträume definiert in welchen ein gewisser Umfang an lauffähiger Software entwickelt und zum Testen zur Verfügung gestellt wird.

Die Iterativen Ansätze verfolgen alle das Ziel, in jeder Iteration funktionierende Software auszuliefern. Diese Software soll aber bereits bestmöglich getestet sein [10, S.29,31].

Dazu ist es erforderlich einige Hilfsmittel in Anspruch zu nehmen, wie zum Beispiel Test-Driven-Development [1, S.186] und Pair-Programming [1, S.194]. Sogenannte best-practices Ansätze in der Softwareentwicklung.

Agile Methoden versuchen möglichst sofort auf Änderungen zu reagieren. Die Veränderung wird höher bewertet als das befolgen eines Plans [17]. Die Idee dahinter ist eine Reaktion auf den Umstand das Softwareprojekte meistens hoch komplex sind und schon die Planung schwierig ist. Es ist schwer möglich alle Konsequenzen eines Projektes bereits im Vorfeld zu durchdenken. Man versucht also bereits als feststehend anzunehmen, das sich Veränderungen im Projekt in Form von Abhängigkeiten und Anforderungsänderungen ergeben werden und bereits darauf zu reagieren. Ein weiterer Punkt ist die Dokumentation. Agile Methoden verfolgen die Richtung nach minimaler Dokumentation [1, S.270]. Der Grund liegt darin, dass Dokumentation sehr zeitaufwändig und immer nur zum Zeitpunkt des Schreibens gültig ist. In der iterativen Vorgehensweise stellt sich aber die Frage der Aktualität. Es ist nahezu unmöglich alle Änderungen zu dokumentieren. Verfechter dieser Methodik verwenden teilweise keine Dokumentation im Code. Der geschriebene Source-Code soll die Dokumentation sein. Dazu werden Teammitglieder angehalten sich an Style-Guides zu halten damit es einheitlich zu lesenden Code gibt. In verschiedenen Entwicklungsplattformen haben sich spezifische Style-Guides entwickelt die eine große Verbreitung gefunden haben. Anders verhält es sich mit Tests. Diese können und

sollen immer angepasst werden um aktuell zu sein. Unit-Tests, werden vom Entwickler während des Programmierens geschrieben [10, S.30]. Es haben sich mittlerweile Vorgehen etabliert die Tests vor der Umsatzung vorsehen. Also zuerst den Test entwickeln und dann den Code schreiben der diesen Test erfolgreich laufen lässt. Dieser Ansatz nennt sich Test-Driven-Development [23] und gehört mittlerweile zu den Standard-Methoden in der Softwareentwicklung.

Die agilen Methoden versuchen der Dynamik Rechnung zu tragen, die sich aus der Softwareentwicklung ergibt. Im Gegensatz dazu haben sich klassischen Verfahren eher in der Richtung fest stehender Termine und Auslieferung im großen Stil (Big-Bang) orientiert. Auf wachsende Dynamik und verändernde Anforderungen können die klassischen Verfahren nicht ausreichend schnell reagieren. Daraus ergibt sich auch die Abgrenzungen der beiden (agilen, klassischen) Richtungen [7].

DIE 12 AGILEN PRINZIPIEN

Prinzipien hinter dem Agilen Manifest

Wir folgen diesen Prinzipien:

Unsere höchste Priorität ist es,
den Kunden durch frühe und kontinuierliche Auslieferung
wertvoller Software zufrieden zu stellen.

Heisse Anforderungsänderungen selbst spät
in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen
zum Wettbewerbsvorteil des Kunden.

Liefere funktionierende Software
regelmäßig innerhalb weniger Wochen oder Monate und
bevorzuge dabei die kürzere Zeitspanne.

Fachexperten und Entwickler
müssen während des Projektes
täglich zusammenarbeiten.

Errichte Projekte rund um motivierte Individuen.
Gib ihnen das Umfeld und die Unterstützung, die sie benötigen
und vertraue darauf, dass sie die Aufgabe erledigen.

Die effizienteste und effektivste Methode, Informationen
an und innerhalb eines Entwicklungsteams zu übermitteln,
ist im Gespräch von Angesicht zu Angesicht.

Funktionierende Software ist das
wichtigste Fortschrittsmaß.

Agile Prozesse fördern nachhaltige Entwicklung.
Die Auftraggeber, Entwickler und Benutzer sollten ein
gleichmäßiges Tempo auf unbegrenzte Zeit halten können.

Ständiges Augenmerk auf technische Exzellenz und
gutes Design fördert Agilität.

Einfachheit -- die Kunst, die Menge nicht
getaner Arbeit zu maximieren -- ist essenziell.

Die besten Architekturen, Anforderungen und Entwürfe
entstehen durch selbstorganisierte Teams.

In regelmäßigen Abständen reflektiert das Team,
wie es effektiver werden kann und passt sein
Verhalten entsprechend an.

FAZIT AUS DEN AGILEN PRINZIPIEN

Die agilen Prinzipien [20] weisen einen Weg der Nachhaltigkeit. Die Teams passen sich an Anforderungen an und reflektieren gegenüber Ihren bisherigen Ergebnissen. Sie sehen motivierte MitarbeiterInnen vor. Es bilden sich Vertrauensebenen rund um motivierte Teams welche selbstorganisierend arbeiten. Es gibt dazu eine funktionierende Umgebung die dieses Verhalten fördert. Veränderungen sind willkommen. Durch die iterative Auslieferung kleinerer Teillieferungen ist es dem Kunden möglich schneller Ergebnisse zu beurteilen und auch Fehler in der Anforderung früher zu entdecken. Diese Anforderungsänderungen dürfen relativ früh in die laufende Entwicklung einfliessen. Teams organisieren sich selbständig und können ihr Arbeitstempo über unbegrenzte Zeit hin halten. Ein Resultat aus Selbstorganisation und Motivation eingebetet in eine funktionierende Umgebung. Daraus resultiert Sicherheit und Stetigkeit. Daraus resultiert mehr Kraft unter geringeren Reibungsverlusten während des Projektes. Direkte Kommunikation ist bevorzugt. Wie auch in 2.1 genannt ist direkte Kommunikation ein Erfolgsfaktor für erfolgreiche Projekte. Fördere direkte Kommunikation, erhöhe dadurch auch das soziale Kapital. Die Kunst der Einfachheit soll gelebt werden um Komplexität zu beherrschen. Einfachheit minimiert auch die Aufwände und minimiert den „Waste“, also unnötigen Aufwänden die durch Komplexität entstehen um dadurch resultierende Fehler wieder zu korrigieren. Das Team ist selbstreflektiert und lernt aus der eigenen Erfahrung. Technische Exzellenz soll gelebt werden. Das Team erweitert über längere Zeiten seine Fähigkeiten. Dies tut es in dem es über vergangene Ergebnisse reflektiert und darüber hinaus versucht best-practices Ansätze zu leben. Vor allem beim Testen. Durch interne Kommunikation wird ebenfalls Wissen optimal und rasch verteilt. Um diese Prinzipien zu leben, braucht es einige Werkzeuge und Überzeugung damit Sie in die momentane Projektlandschaft integriert werden können. Es gab über die Zeit verschiedene Ansätze um die agilen Prinzipien umzusetzen. Daraus haben sich viele Verfahren, mehr oder weniger ähnlich, entwickelt. Hier sollen einige der Verfahren vorgestellt werden welche in der breiten Öffentlichkeit wahrgenommen wurden und auch weit verbreitet sind. Diese Methoden nehmen vor Allem auch auf die Rollen-Struktur Rücksicht. Es wurden zur Darstellung der bestehenden Methoden Informationen aus eigener Erfahrung und aus der Recherche aus [1] [4] [5] [6] [7] und [10] herangezogen.

SCRUM

SCRUM [1, S.31] ist wohl die bekannteste Methode aus den agilen Verfahren und beschreibt einen iterativen Software-Entwicklungsprozess. Es werden Time-boxes eingerichtet, sogenannte Sprints [1, S.287]. Innerhalb dieses Sprints möchte man funktionierende Software fertigstellen (inklusive Tests) und am Ende dem Kunden zur Abnahme zur Verfügung stellen. Kunden sollen möglichst früh die Umsetzung Ihrer Anforderungen testen können. Die Aufgaben in einem Scrum-Projekt übernehmen verschiedene Rollen.

Dazu gehören:

- Scrum Master [1, S.145]
  Ist die Person welche Verantwortung übernimmt und das Team begleitet. Er versucht das Team von Widerständen und Reibungsverlusten frei zu halten. Er versucht das Team im Einsatz von SCRUM zu begleiten und nimmt Einfluss sowohl nach innen (Team) als auch nach außen. Er versucht dem Team eine Atmosphäre der Zusammenarbeit bereitzustellen. Der Scrum Master sollte außer den emotionalen Skills und Empathie auch technisches Know-How mitbringen um dem Team helfen zu können. Er beteiligt sich nicht an der Entwicklung oder den Tests. Dem Scrum Master kommt die Rolle eines Coach zu, der die Struktur des Teams im Auge hat und darauf achtet das Kommunikation reibungsfrei verläuft.

- Product Owner [1, S.135]
  PO sorgt dafür, dass ein Team die richtigen Ziele verfolgt. Der PO ist die Schnittstelle zum Kunden und seinen Anforderungen. Er strukturiert Sie, erstellt, pflegt und priorisiert auch das Backlog. Er kann fachliche oder technische Änderungen rasch erkennen. Er schätzt die Aufwände der Arbeitspakete und ordnet die Größen in der Reihenfolge so, das Kunden ihre Wünsche erfüllt bekommen (Art und Reihenfolge) aber die Größen nicht die Kapazitätsgrenze der einzelnen Teams überschreiten. Die Rolle des Projektmanagers wurde in SCRUM komplett gestrichen. Ehemalige Projektmanager können in die Rolle des Scrum Master oder Product Owner gehen. Allerdings erfordert dies ein hohes Maß an Schulung um die ehemaligen Gedanken des Geschäftes in die neue Philosophie umsetzen zu können.

- Scrum Team [1, S.206]
  Das Team sind alle Personen in die laufende Entwicklung eingebunden sind. Das Team organisiert sich weitestgehend selbständig und übernimmt Verantwortungsgrade, die früher der Projektmanager getragen hat. Architekten , Entwickler, Tester, Administratoren usw., sitzen alle in einem Team. Sie müssen lernen als Team zu funktionieren. Teilweise verschwinden im Bereich Entwicklung und Test die Grenzen, da Entwickler Tests schreiben (Unit-Test). Die Teamgröße sind üblicherweise 4-10 Personen. Sie bekommen im sogenannten Sprint-Planning die Arbeitspakete mitgeteilt und äußern sich über die Umsetzbarkeit im geplanten Zeitrahmen (Timebox/Sprint). Das Team reflektiert nach jedem Sprint in der sogenannten Retrospektive über die eigene Leistung. Das Team kommuniziert direkt und persönlich jeder Zeit, aber auf jeden Fall einmal am Tag zum Daily-Meeting.

- Externe Rollen
  Dazu gehören intern das Management sowie extern der Kunde. Diese Rollen erhalten Informationen über Ihre Schnittstellen (Product Owner, Scrum Master).

Scrum Teams beziehen Ihre Arbeitspakete (sogenannte User stories) aus dem Backlog welches vom Product Owner aufbereitet wird, der wiederum die Schätzungen der einzelnen Aufwände mit dem Kunden vorgenommen hat (Estimation). Das Team hält während des laufenden Sprints ein Sprint-Planning ab, um die Planung für den nächsten Sprint durchzuführen. Dabei erklärt der Product Owner die nächsten Schritte (Arbeitspakete aus dem Backlog) und das Team schätzt die Aufwände für den Sprint nochmals ab. Ebenfalls im laufenden Sprint wird eine sogenannte Retrospektive abgehalten. Sie soll dem Team ein Feedback über den letzten (vorhergehenden) Sprint ermöglichen. Feedback-Schleifen sind absolut erforderlich für die Entwicklung der Teams. Nur so können sie Fehler erkennen und ändern.

Grundsätzlich ist Scrum erdacht als Methode, welche Teams die Möglichkeit einräumt sich selbstständig zu organisieren. Sprints innerhalb eines Projektes (Unternehmen) sollten immer gleich lang sein (Bsp.: 14 Tage). Als Maßeinheit für Arbeitspakete wird die Einheit sogenannter Story-Points genommen [1, S.332]. Dies ist eine zeitlose Einheit, welche lediglich die Aufwände von Arbeitspaketen in Verhältnis setzt. Jedes Paket wurde vom PO mit Story-Points versehen um die Größe des Aufwandes darzustellen. Schafft ein Team zum Beispiel 20 Story-Points in einem Sprint umzusetzen so ist dies die sogenannte Velocity eines Teams. Teams können unterschiedliche Velocities [1, S.332] haben, das liegt an der Natur der Sache. Leider wird Scrum als Methode oft korrumpiert durch falsche Schätzungen, welche wiederum dazu führen das Sprints nicht fertiggestellt werden können. In der Praxis ist zu sehen, das Teams die einmal einen Sprint nicht schaffen, es bis an das Projektende nicht mehr ändern können. Ein Grund dafür mag in dem Umstand liegen, das Product Owner und das Management im direkten Kontakt zum Kunden oftmals zu optimistisch die Aufwände schätzen. Die Aufteilung der Story-Points und deren Beziehung zueinander wird dann verzerrt. Dies hat eine fatale Auswirkung auf die Teams. Teams die mit Ihrer Velocity ständig hinterher hinken, sind psychologisch gesehen sehr angespannt und deren Leistung sinkt weiter.

Grundsätzlich ist der Aufwand SCRUM als Projekt-Management Methode einzuführen sehr hoch. Es erfordert meistens einen hohen Schulungsgrad der MitarbeiterInnen (aller Agierender) und der Geschäftsleitung sowie dem Management. Im SCRUM erfolgreich leben zu können müssen die Grundparameter von SCRUM erfüllt werden, da sonst das System nicht gelebt werden kann und SCRUM mehr Behinderung wird als ein Vorteil. Wichtig ist die Bedeutung der Akzeptanz der Agierenden. [1, S.126] erklärt das der Einsatz agiler Methoden meist mit Widerständen der MitarbeiterInnen konfrontiert ist. SCRUM stellt die bisher gelebten Methoden grundsätzlich auf den Kopf, da so ziemlich alles anders gemacht wird als vorher. Dieser Umstand weist auf einen besonders hohen Überzeugungsaufwand der MitarbeiterInnen hin.

XP EXTREME PROGRAMMING

XP [4, S.66] ist ein relativ radikaler Ansatz in der Software-Entwicklung. Vor allem an die Software-Entwickler stellt XP hohe Anforderungen. Die Anforderungen liegen im Bereich der technischen Qualifikation und persönlicher Entwicklung. Dazu hat es sich zum Ziel gesetzt einen „best practices“ - Ansatz zu verfolgen. Der Architekt beginnt seinen ersten Entwurf zu machen. Dann werden die Releases geplant. In die Release-

Planung fließen die User-stories ein. Dann beginnen Iterationen der Entwicklung. Das aktuelle Release wird nach eingehenden Tests und Fehlerbehebungen an den Kunden übergeben. In XP werden während der Entwicklung Praktiken wie Pair-Programming [4, S.70] und Test-Driven-Development (TDD) [23] angewandt. XP erfordert ein hohes Maß an Kommunikation zwischen den Team-Mitgliedern. XP zielt auf Einfachheit ab. Das bedeutet, dass es besser ist Dinge möglichst einfach umzusetzen und evtl. später zu erweitern als zu generisch Dinge zu entwickeln, die dann nie eingesetzt werden. Darin liegt eine Stärke in XP. XP verwirft auch gerne mal einen Ansatz, wenn die Neuentwicklung erfolgsträchtiger erscheint als auf dem bestehenden fehlerhaften Ansatz zu beharren. XP ist nicht für große Projekte gedacht. Es gibt eine Rollenverteilung welche aber den generellen Rollen in Projekt-Teams sehr nahe ist. Sie ist weniger abstrakt als beispielsweise in SCRUM-Prozessen.

Folgende Rollen werden in XP gesehen [4, S.67]:

- Coach
  Versucht die Werte und Prinzipien von XP im Team aufrecht zu erhalten. Erfahrene Teams brauchen diese Funktion nicht da Sie diese Werte selbst leben.
- Tester
  Entwickeln die automatisierten Tests und sind Teil des Teams. In TDD werden zuerst die Tests entwickelt und nachher Code vervollständigt um die Tests zu erfüllen. Dies ist ein aufwändiger aber qualitätssichernder Ansatz.
- Interaction Designer
  Schnittstelle zwischen Entwickler und Kunde. Entwickeln mit dem Kunden die User-stories und kontrollieren die Umsetzung.
- Architekt
  Entwirft die technische Struktur der Applikation und ändert Sie bei bedarf. Es ist zuständig für den reibungsfreien Ablauf der Module.
- Projektmanager
  Schnittstelle zur Kommunikation zwischen Entwicklern und Kunden
- Geschäftsführung
  Trägt das Projektrisiko, darf die Ergebnisse der Projektteams jederzeit einsehen
- Anwender (Kunde)
  Erstellt die User-stories und gibt seine Anforderungen bekannt
- Entwickler
  Setzt die User-stories in lauffähigen Code um und entwickelt die Tests (gemeinsam mit den Testern , Unit-Tests)
- Tracker
  beobachtet die Leistung des Teams und warnt, falls Ziele im Zeitrahmen nicht erreicht werden können.

Wie bei SCRUM hat auch XP eine klare Rollenverteilung, lässt aber durchaus schlankere Strukturen mit Mehrfachbesetzung von Rollen auf Einzelpersonen zu. Die primäre Idee (best-practices) soll gewahrt bleiben. Die Verteilung der Rollen dient zum erhalt der agilen Sichtweise und dem Strukturaufbau.

Teams erhalten Feedback[6, 6.1.3] durch Ebenen von Feedback-Loops [24]:

- Pair Programming ergibt direktes Feedback vom Partner.

- Unit-Tests ergeben Feedback innerhalb von Minuten, wenn sie ausgeführt werden.
- Stand Up Meetings werden täglich abgehalten.
- Akzeptanz-Tests mehrmals in einer Iteration. Eine Iteration läuft eine bis mehrere Wochen und Releases werden innerhalb von Monaten ausgeliefert.

Das Team bezieht sein Feedback also über in sich geschachtelte Ebenen. Die Übersicht ist sicher nicht so einfach zu erkennen. Es erfordert eine Zusammenarbeit des Teams um die Feedback-Loops nutzen zu können. Die Zusammensetzung der Teams ist idealerweise durch Personen mit unterschiedlichen Fähigkeiten gegeben. Probleme können so aus verschiedenen Perspektiven betrachtet werden. Auch der Austausch von Wissen und die direkte Kommunikation sind dadurch gefördert. Verbesserungen sollen sofort durchgeführt werden, wenn immer möglich (Refactoring). XP geht ja vom Ansatz aus das es besser ist komplexen Code wegzuschmeissen wenn er vereinfacht werden kann als aufrecht zu erhalten, da die zukünftigen Aufwände bei Änderung und Wartung sonst ungleich höher werden. Die Team-Mitglieder sitzen räumlich enger zusammen. Dies bedingt alleine der Umstand das Pair-Programming praktiziert wird und der Kontakt zu den Testern enger ist als üblich. Wenn Team-Mitglieder Verantwortung für ein Thema übernommen haben, sind sie bis ans Ende dafür zuständig. Continuos Integration (ständiges integrieren neuer Code-Teile in die Applikation, am Besten automatisiert) wird angewandt.

XP funktioniert nur wenn die Basiselemente wie Test-Driven-Development, Pair-Programming, einfaches Design, continuos integration wie eingefordert angewandt werden. Dies erlaubt keine besonders hohe Flexibilität in der Organisation, bis auf die Rollenverteilung. Es können mehrere Rollen auf eine Person gebündelt werden.

FDD FEATURE DRIVEN DEVELOPMENT

Diese Methode [4, S.73] stellt die Features einer Anwendung in der Planung in den Vordergrund. Ein Zielgebiet für diese Methode sind Festpreisangebote mit festem Umfang. Aus dem Gesamtmodell wird eine Feature-List erstellt. Pro Feature in der Feature-List wird dann die Planung und Umsetzung durchgeführt.

Der Prozess stellt sich wie folgt dar:

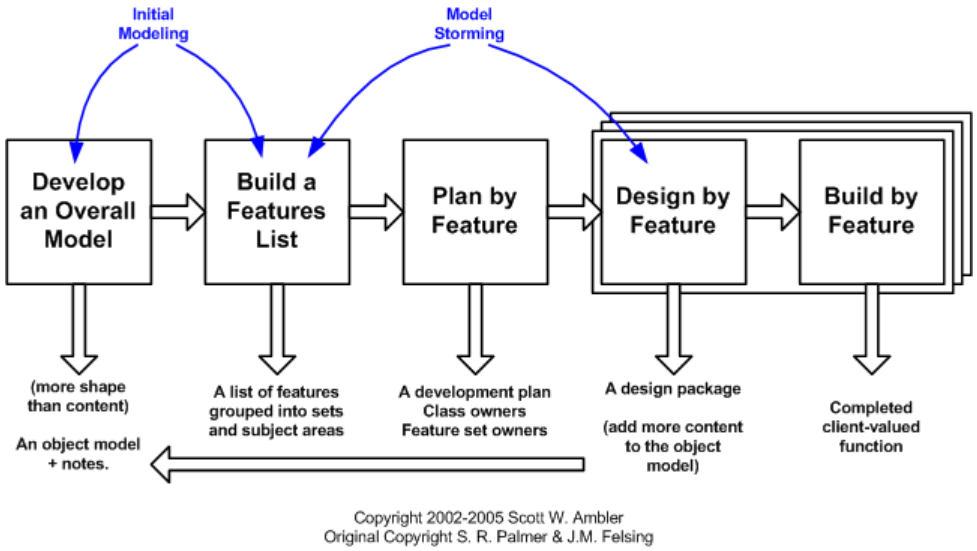


Abb. 3: Quelle http://www.agilemodeling.com/essays/fdd.htm

Es werden folgende Rollen definiert (nach [18] sowie [4, S.73]):

- Domain Manager
  Ist der Benutzer oder Auftraggeber und stellt sein Fachwissen zur Verfügung
- Release Manager
  Analysiert die Ergebnisse und beobachtet den Fortschritt des Projektes
- Language Guru
  Ist der Ansprechpartner für technische Fragen über verwendete Technologie und Programmiersprachen
- Build Engineer
  Kümmert sich um den Build des Systems und die Integration der Komponenten
- Toolsmith
  Stellt durch seine Tätigkeit Hilfsmittel (Programme) zur Verfügung die nötig sind um die Arbeit der Class Owner zu erleichtern.
- System Administrator
  Richtet das System (Hard & Software) grundsätzlich ein und wartet es.
- Tester
  Entwickelt die Test für das Gesamtsystem und führt die Tests durch
- Deployer
  Ist für das Deployment und Datenübernahmen zuständig

- Technical Writer
  Beschreibt die technische Entwicklung und entwirft die Dokumentation

FDD wird in Projektgrößen zwischen einem Team und mehreren Teams angewandt. Um alle Rollen zu besetzen, ist doch ein erhöhter Aufwand nötig. Für kleine Projekte wird der Aufwand zu hoch sein. Für einen funktionierenden FDD Prozess sollen die Rollen alle Verteilt sein. FDD geht davon aus, dass bereits zu Beginn die Anforderungen alle bekannt sind. Ändern sich die Anforderungen nicht, so stellt FDD eine sehr effektive Methode dar.

ASD ADAPTIVE SOFTWARE DEVELOPMENT

ASD [4, S.39] ist eine sehr abstrakte agile Methode. Es werden lediglich 3 Rollen vergeben

- Kunde
- Moderator
- Protokollant

ASD dient dem Prinzip ständig zu hinterfragen (Bsp. 4 Wochen) ob das laufende Ergebnis den Erwartungen des Kunden entspricht. Um dies festzustellen treffen sich die handelnden Personen und unterhalten sogenannte Jad-Sessions (Joint Application Developer Session). In diesen Treffen wird der Fortschritt festgestellt und etwaige Änderungen eingefordert. Diese laufen in der nächsten (den nächsten) Iteration(en) (nach Prioritäten) ein. Zwischen den Jad-Sessions laufen die Phasen „Spekulation – Kollaboration – Lernen“ durch. ASD ist ein Feature-Driven-Development Ansatz. Also für Projekte geeignet wo der Umfang relativ klar bekannt ist.

CRYSTAL METHODS

Crystal Methods [4, S.45] ist ebenfalls eine eher abstrakte Methode des Software-Projektmanagement.

Crystal Methods ist eine Familie an Methoden die sich an der Komplexität und Größe der Projekte anpasst. Von einfachen Strukturen und Rollenverteilungen zu komplexen Strukturen wachsen die davon abgeleiteten Methoden.

Es gibt mehrere Methoden die entwickelt wurden oder sich in Entwicklung befinden. Entwickelt wurden z.b. Crystal Clear und Crystal Orange.

Die Grundprinzipien von Crystal Methods sind:

- Je Größer das Team, desto größer die Methodik
- Je kritischer ein Projekt ist, desto intensiver ist die Methodik
- Höheres Gewicht in der Methodik erhöht die Kosten
- Die effektivste Kommunikation ist von Angesicht zu Angesicht

Alistair Cockburn beschreibt diese Methode folgendermaßen (Zitat aus [19]):

"Crystal is a family of human-powered, adaptive, ultralight, stretch-to-fit software development methodologies"

Er meint damit das Crystal Methods Menschen in den Vordergrund stellt. Der Projekterfolg soll durch das Fördern der Arbeiten der Team-Mitglieder erreicht werden. Der Overhead des Projektes soll minimiert sein, um leicht praktisches Arbeiten zu ermöglichen. Es wird zuerst etwas Kleineres entwickelt, um es dann zu vergrößern, bis es genau passt. Der Grundsatz ist, dass es leichter und sicherer ist etwas Kleineres weiter zu entwickeln als etwas Großes in seinen Möglichkeiten zu beschneiden. Dies ist ein paralleler Ansatz zu XP, wo ebenfalls ein Grundprinzip Einfachheit ist. Die Rollen sind je nach Methode aufbauend, nach dem Prinzip das kleineres größer gemacht werden soll.

In Crystal Clear gibt es die Rollen:

- Kunde
- Programmierer und Designer
- Benutzer

Alle weiteren Crystal Methoden (skaliert auf größere Projekte) fügen nach Bedarf und Methode weitere Rollen hinzu. Der Grundgedanke dieser Methoden ist skalierbar zu sein und nur einzusetzen, was auch benötigt wird. Es ist ein sehr flexibles System das je nach Projekt immer anders aussehen kann. Dieser Umstand macht Crystal Methods auch schwer fassbar.

KANBAN UND SCRUMBAN

Kanban [10, S.14] ist eine Methode aus dem Lean Management Bereich (schlankes Management). Kanban gibt im Grunde keine Rollen vor. Kanban Systeme können auf bestehende Projektmanagement-Methoden aufgesetzt werden. Es ist im Grunde ein Change-Modell für bestehende Methoden.

Die Basis von Kanban besteht im wesentlichen der „continuos-flow", der Projektflusses. Ein Kanban System hat einen Anfang und ein Ende. Arbeitspakete fliessen durch das System. Wenn jeder Task auf eine Karte geschrieben ist, dann wandert diese Karte in ihren Verarbeitungsprozessen vom Backlog durch verschiedene Bereiche von links nach rechts bis der Task vollständig erledigt ist. Kanban versucht so wenig wie möglich „waste" (auf Halde) zu produzieren. Umgesetzt in die Software-Entwicklung bedeutet dies, keine Arbeiten zu erledigen wenn nachgereichte Aufgaben nicht abgearbeitet werden können. Wenn also der Fluss gestört ist. Zur Visualisierung wird ein Kanban-Board verwendet [10, S.73]. Die einzelnen Bearbeitungsschritte sind an einem Kanban-Board ersichtlich in Form von Spalten und weitreichenderen Elementen wie Swim-Lanes , Progress-Tracking usw.(dazu später mehr). Die Reihenfolge von Arbeitsschritten in einem Kanban-System könnte sein: *"Backlog – Ready for Development – Active Development – Testing – Deployment – Done"*

Der Beginn ist also das Backlog und das Ende dieses Systems ist "Done". Das ist wichtig weil zwischen dem Anfang und dem Ende der Durchsatz gemessen wird. Die einzelnen Bereiche (Spalten) werden dann begrenzt. Es dürfen also nur eine gewisse

Anzahl an Tasks im jeweiligen Bereich (Spalte) eingehängt sein. Dies ist der Faktor WIP (Work in progress). Die Begrenzung des WIP bringt zu tage das sofort ersichtlich wird wo Engpässe entstehen, da sich in den Queues davor Staubildungen zeigen. In der Visualisierung ist dies auf einen Blick ersichtlich. Es können dann Ressourcen umgeschichtet werden um den Fluss wieder herzustellen. Dies ist die balancierende Eigenschaft von Kanban-Systemen.

Kanban kommt eigentlich aus der Automobilindustrie [10, S.8], wird aber seit geraumer Zeit in Software-Projekten eingesetzt. Es ergänzt oft bestehende Methoden wie zum Beispiel Scrumban. Als Scrumban wird die Methode oder der Prozess verstanden, wo Scrum-Teams immer mehr Lean/Kanban Elemente aufnehmen, wie zum Beispiel den WIP, weil sich die Begrenzung gleichzeitig befindlicher Artefakte, als Vorteil erweist. Gleichzeitig gibt es keine Sprints mehr da das Kanban Prinzip des Continuos-Flow angenommen wird. Die ist sehr bemerkenswert. Es gibt auch keine Estimations mehr, da alle Items gleiche (ähnliche) Größe haben. Es gibt jedoch Gruppen von Größenordnungen. Änderungen oder Neuanforderungen müssen auch keine Sprints abwarten, da sie über das Backlog sofort in den Bearbeitungsfluss eingehen können. Es ist nur eine Frage der Priorisierung [10, S.190] wie die Artefakte in den Projektfluss eintreten. Dies ist auch eine Schlüsselstelle in Kanban-Systemen. Nach welchen Kriterien werden die Artefakte in das Backlog eingepflegt, denn Teams entnehmen in Kanban-Systemen selbstorganisierend die Arbeitspakete und reihen sie in den Fluss ein. In solchen Systemen wird erwartet das es eine Firmenkultur gibt, die MitarbeiterInnen so viel Vertrauen entgegenbringt damit sich echte Teams entwickeln können.

2.2.3 Visualisierung und Metriken

Alle agilen Methoden verwenden Mittel zur Visualisierung und zur Messung der Leistung von Teams. Ein Punkt den die agilen Methoden alle Haben ist die mehr oder weniger ausgeprägte Selbstorganisation von Teams. Damit sich Teams selbst reflektieren können und das Management Kenngrößen entwickeln kann um Entscheidungen zu treffen stehen mehrere Hilfsmittel zur Verfügung.

VISUALISIERUNG

Ein wesentlicher Punkt ist die Visualisierung der Prozesse. Es soll für jeden ersichtlich sein wer woran arbeitet und wie der Status der Tätigkeiten ist. Eines der geeignetsten Mittel ist die Verwendung eines Kanban-Board [10, S.73]. Ein Kanban-Board bildet den Status des kontinuierlichen Flusses an Aufgaben durch die Bearbeitungsreihe ab:

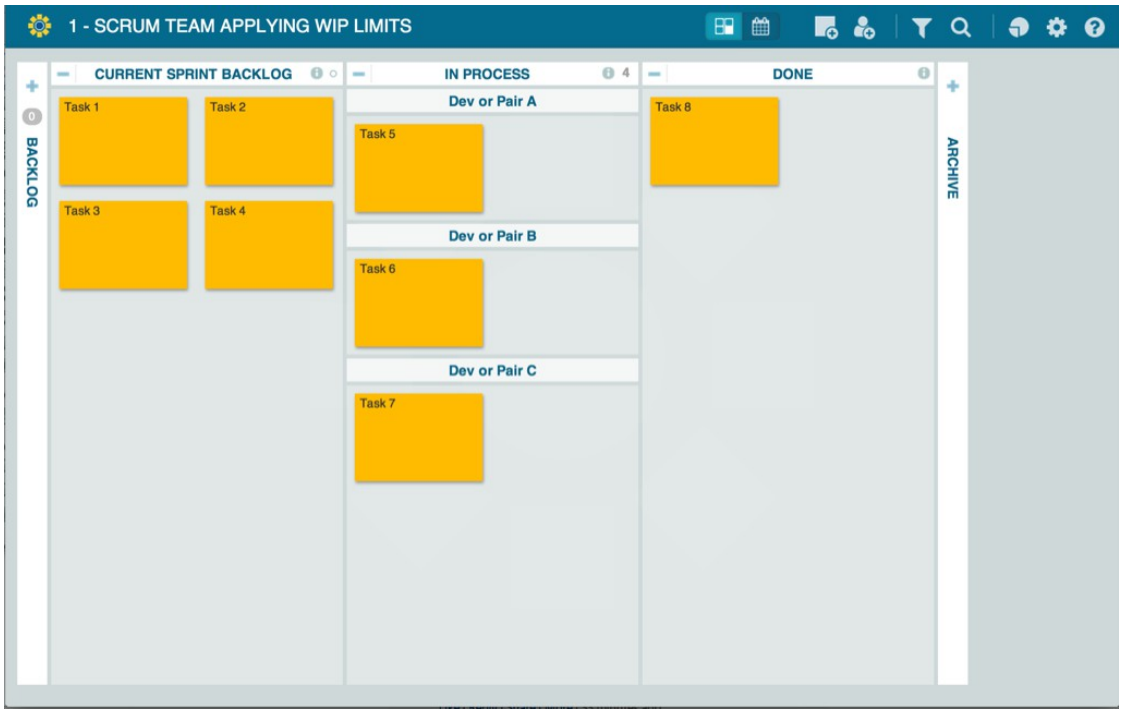


Abb. 4: Quelle:http://leankit.com/blog/2010/12/10-kanban-boards-leankit-kanban-style/

Kanban-Boards definieren den Fluss der Arbeit von links nach rechts. Links im Backlog werden Tasks zur Bearbeitung eingereicht. Ein Task durchwandert dann die verschiedenen Bearbeitungsbereiche. Manche Spalten haben in der Bezeichnung eine WIP-Begrenzung angegeben. Es sollen nicht mehr als die angegebene Anzahl des WIP-Limits von Tasks in die jeweilige Spalte aufgenommen werden. Dies ermöglicht einen klaren Überblick über Engpässe, da sich ein Bereich sofort füllen würde wenn Tasks nicht weitergereicht werden können. Es ermöglicht den Anwendern auch Ihre parallelen Arbeiten zu begrenzen und sich auf das Wesentliche zu konzentrieren. Ausserdem kann jeder Anwender selbst sehen was sein Bereich ist (mit Namen versehene Tasks) und es kann auch die Durchlaufzeit gemessen werden. Die

durchschnittliche Durchlaufzeit ist eine der Messmöglichkeiten und gibt Aufschluss darüber wie performant Teams agieren. Boards können nur aus wenig Spalten bestehen oder wesentlich komplexer Aufgebaut sein. Je nachdem wie erfahren Teams sind, gestalten Sie das Board selbst. Außer den vertikalen Linien können auch noch horizontale Linien, sogenannte Swim-lanes eingetragen werden um unterschiedliche Bereiche zu markieren (Bsp. GUI-Entwicklung, Server-Entwicklung). Es können Serviceklassen [10, S.132] entwickelt werden und ebenfalls getrennt in den Fluss gereiht werden. Es können Multi-Tasks (Tasks mit mehreren Arbeitsschritten) am Board heften und vieles mehr. Die Grenze bestimmt der Entwicklungsgrad des Teams. Je entwickelter ein Team ist desto komplexer aber feingranularer sind die Visualisierung und somit auch die Aussagen daraus. Das Board wächst mit dem Team, ist einfach erstellt, und leicht zu warten. Es bildet keine Sprints ab, sondern einen kontinuierlichen Prozess. Es bildet den Arbeitsfluss ab. Scrum-Boards hingegen werden wie der Name schon sagt in SCRUM-Methoden verwendet. Diese Boards unterscheiden sich wesentlich gegenüber Kanban-Boards. Sie bilden nämlich keinen kontinuierlichen Fluss ab, sondern einen Status im Moment.

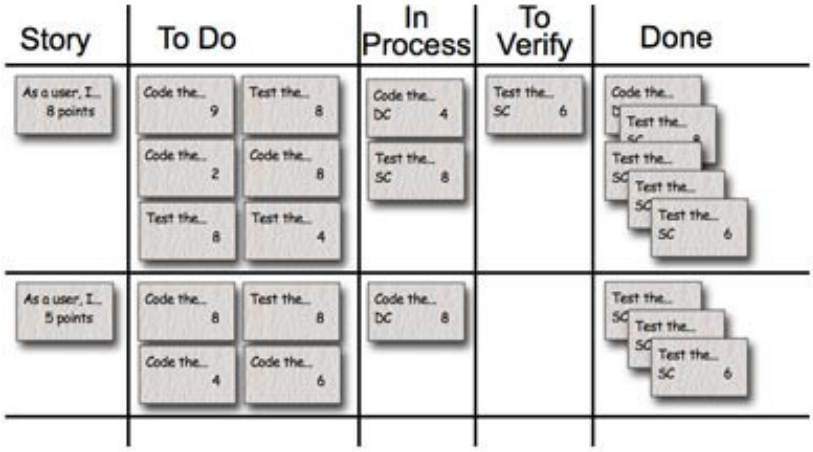


Abb. 5: Quelle: http://www.mountaingoatsoftware.com/agile/scrum/task-boards

Dieser Unterschied ist essentiell. An einem Scrum-Board lässt sich der Status des aktuellen Sprints ablesen. Weiterreichende Scrum-Boards bilden auch den nächsten Sprint ab. Die Basis-Elemente sind ebenso wie beim Kanban-Board das Backlog, die Tasks, und die Einteilung in zu bearbeitende Elemente bis hin zu „Done“. Es findet jedoch keine wie immer geartete Begrenzung einzelner Spalten statt. Der Grund liegt im Wesen von SCRUM. Es iteriert in festen Zyklen und reiht am Beginn eine Reihe von Aufgaben ein. Während des Sprints werden keine neuen Aufgaben eingereiht. Es wird jedoch während des Sprints bereits für den nächsten Sprint geplant. In Scrum werden die Tasks meist Personen zugeordnet. Im Grunde sind Scrum-Boards eher einfach aufgebaut und ändern sich im Laufe der Zeit eher wenig. Es steht dahinter auch eine gänzlich andere Philosophie als in Kanban.

METRIKEN

DER SINN VON MESSUNGEN

Mike Cohen schreibt in [1] über die Sinnhaftigkeit von Messungen folgendermaßen , Zitat [1, S.455]: *„Die meisten Menschen die Sie fragen was der Sinn und Zweck von Messungen ist, werden Ihnen antworten das der Sinn darin besteht, zu bestimmen wie groß, wie schwer oder wie lang etwas ist o der wie viel es von etwas gibt. Das ist jedoch eine übermäßig anspruchsvoll Definition von Messen. Der wirkliche Zweck von Messungen besteht darin, den Grad der Unsicherheiten zu verringern. Dazu muss die Messung auch nicht exakt sein“*

Man kann alles Mögliche messen. Die Anzahl von Fehlern in einem gewissen Zeitraum. Die Zahl ausgelieferter Features usw. Auch die Frage nach den Erträgen eines entwickelten Produktes können nicht die Frage beantworten wie ein Team arbeitet. Umgekehrt kann auf Basis von Vergleichsdaten wie sehr ein Team im Jahresvergleich seine Arbeitsweise verbessert hat und agile Prozesse deutlicher angenommen hat keine Aussage getroffen werden ob dieses Team besser und kostengünstiger Entwickeln wird. Allerdings wird die Wahrscheinlichkeit höher sein. Um die Arbeitsweise von Teams zu erkennen gibt es zum Beispiel die Shodan-Konformitätsstudie [1, S.463] oder Agile:EF [1, S.458]. Dies sind Fragebögen die auf die gelebten Praktiken der Teams eingehen und schnell zu beantworten sind. Auf dieser Grundlage lässt sich über einen längeren Zeitraum feststellen wie die Teams ihre Arbeitsweise verbessern können.

VERZERRTE MESSUNGEN

Mike Cohen bemerkt in [1, S.456] ebenfalls die Probleme bei der Einführung von Metriken. Zitat [1, S.465] *„Wenn Sie einem Team mitteilen, dass es anhand der Anzahl der Fehler im Bug-Tracking-System bewertet wird, so wird diese Anzahl sinken – vielleicht auf Grund ehrlicher Verbesserungen, vielleicht aber auch, weil die Teammitglieder Möglichkeiten finden, sich auf inoffiziellem Wege über Bugs auszutauschen und damit das Bug-Tracking-System umgehen.“*. Möglicherweise auf Kosten der Qualität oder Produktivität. Um Verzerrungen in der Messung zu vermeiden wurde die Balanced Scorecard herangezogen. Sie bewertet auf Basis einer breiten Betrachtung die Teams. Eine Auflistung der betrachteten Themen findet sich ebenfalls in [1, S.465] Zitat:

- Operative Leistung
  Die Teams streben danach, qualitativ hochwertige Produkte zu erzeugen und Termine und Kostenvorgaben zu erfüllen
- Anwenderorientierung
  Die Teams liefern Features für Anwender und Kunden
- Geschäftswert
  Die Teams liefern dem Unternehmen Werte in Form von Kosteneinsparungen oder erhöhten Erträgen

- Zukunftsorientierung
  Die Teams liefern heute Produkte und Features, erweitern aber ihre Fähigkeiten und Möglichkeiten für die Zukunft

Die Balanced Scorecard enthält diese vier Bereiche und wird pro Bereich um Strategische Ziele ergänzt. Generell wird empfohlen Metriken einfach zu halten.

METRIKEN IN KANBAN UND WIP-TRACKING

In Kanban gibt es, bedingt durch das kontinuierliche Fluss-System, etwas andere Metriken [10, S.147]. Kanban zeigt nicht ob ein Projekt einem Plan folgt. Es zeigt vielmehr ob es wie ausgedacht arbeitet. Es soll vorhersehbar sein in Bezug auf Durchsatz und kontinuierlicher Verbesserung.

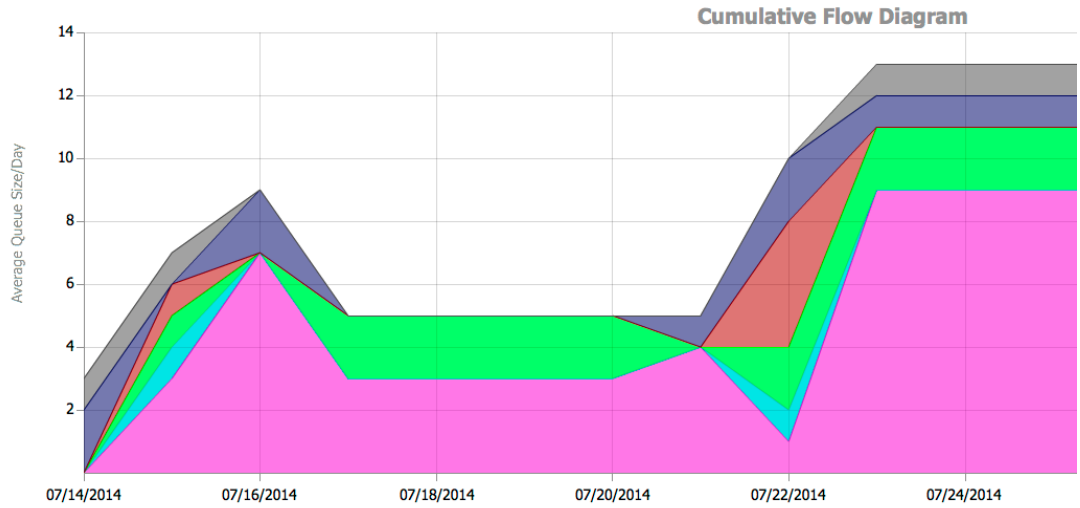


Abb. 6: Cumulative Flow Diagram, eigene Darstellung

Das Cumulative-Flow Diagramm [10, S.32] zeigt die Menge der Aufgaben pro Prozessschritt an. Es ist ein Indikator über die Funktion eines Kanban-Systems.
Wie man am Beispiel sieht hat sich der Durchsatz gegen 07/16/2014 auf einer ersten Spitze befunden. Dann lief das System bis zum 07/20/2014 gleichmäßig stabil um danach die Produktivität wieder zu steigern und gegen 07/24/2014 wieder zu stabilisieren. Die Vorteile dieses Diagramms sind das eindeutig ersichtlich ist ob das System wie gewünscht arbeitet oder ob es Schwankungen unterliegt (wie am Beispiel ersichtlich). Das Ziel eines funktionierenden Kanban-System ist es einen konstanten Fluss an bearbeiteten Tasks zu erzeugen (im Durchschnitt). Es ist auch sehr einfach zu erstellen und ein automatisiertes Produkt bei elektronischen Kanban-Boards. Die unterschiedlichen Farben zeigen die einzelnen Prozess-Schritte an. Die Höhe ist die Anzahl der bearbeitenden Tasks.

Average Cycle Times

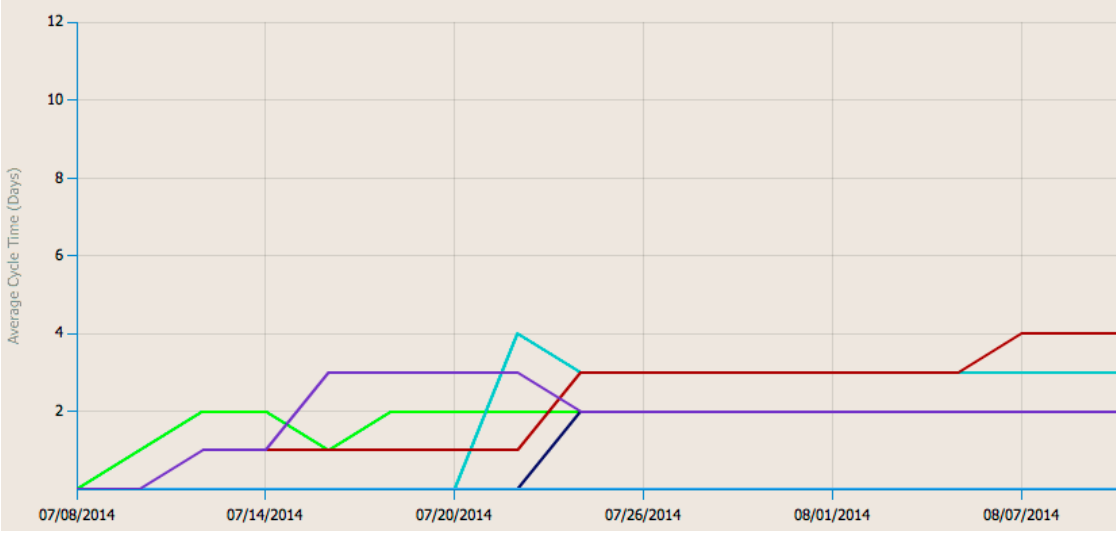


Abb. 7: Average Cycle Times, eigene Darstellung

Dieses Diagramm [10, S.149] zeigt den Durchschnitt des Durchsatzes an, wenn ein Ticket aus dem Backlog ins System kommt und letzten Endes auf „Done“ gesetzt werden kann. Also vom Anfang bis zum Ende. Es ist ein Indikator für die verwendete Zeit zu Bearbeitung der Tasks. Hier sieht man deutlich das es bezogen auf unterschiedliche Serviceklassen unterschiedliche Durchlaufzeiten gibt. Das Bild selbst ist aber stimmig. Nach dem Start des Systems braucht es eine Zeit bis sich der Durchlauf stabilisiert. Ab 07/26/2014 sind die Werte sehr stabil.

Efficiency Diagram

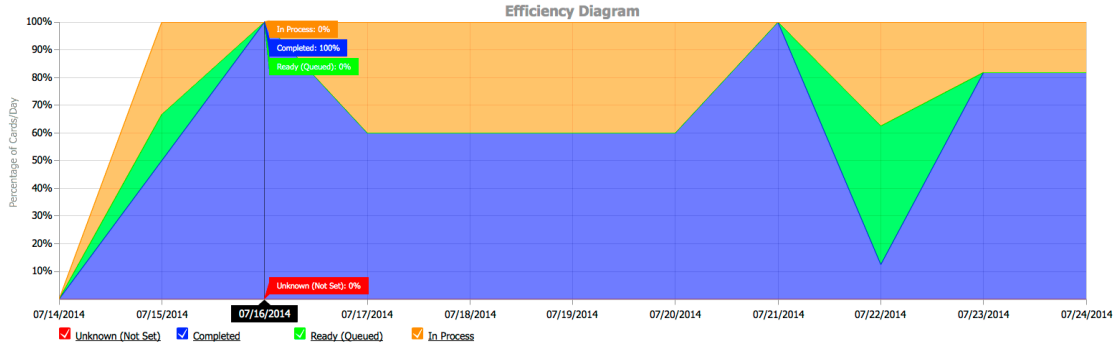


Abb. 8: Efficiency Diagram, eigene Darstellung

Dieses Diagramm zeigt wie Effizient die Tagesleistung ist. Also wie viele Karten (Tasks) verarbeitet werden konnten. Blau ist dabei „Completed“ grün ist „Queued“.
Auch dieses Diagramm ist sehr aussagekräftig und zeigt die Schwankungen im System.

Generell zeigen gleichmäßige Graphen ein funktionierendes Kanban System an. Die Betrachtung des Durchsatzes [10, S.148] inKanban ist eine Betrachtung der Produktivitätsentwicklung über die Zeit. Im Cumulative Flow Diagram ist der

Indikator das ein System richtig läuft zu sehen, wenn die einzelnen Streifen glatt sind und in der Höhe gleich bleiben. Dann läuft der Fluss sozusagen gleichmässig. Der Betrachtung des Durchsatzes in Kanban verleitet zum Ableiten von Größen wie Umsatz oder Aufwand. Man ist versucht aus dieser Betrachtung auch Zeiträume abzuleiten für die Planung. Dies kann auch alles funktionieren, jedoch sollten die Teams schon einige Erfahrung im Umgang mit Kanban haben. Dann kann es Größen an die Karten heften. Der wichtigste Aspekt bei der Betrachtung des Durchsatzes ist jedoch zu sehen ob das System funktioniert und ob es kontinuierliche Verbesserungen gibt. Das ist der wahre Wert des Kanban Systems. Es optimiert die Arbeitsweise von Teams. Sind Teams dadurch wesentlich effizienter, können kostengünstiger und mit höherer Qualität entwickeln? Die Wahrscheinlichkeit ist sehr hoch. Ein funktionierendes Kanban System sollte sich auch in der Zufriedenheit der Beteiligten äußern.

Ein wichtiger Aspekt ist auch die Betrachtung der Durchlaufzeit im Verhältnis zu gesteckten Terminen oder Mengen an ausgelieferten Features. Scrum versucht diese Aussagen zu treffen und misst die Velocity. Velocity [1, S.118] ist die Menge an Einheiten die ein Scrum-Team in einem Sprint (oder im laufe der Zeit über mehrere Sprints) erfolgreich abarbeiten kann. In Scrum werden Arbeitseinheiten als zeitlose Größen geschätzt, sogenannte Story-Points. Die Velocity wäre also die Summe der Story-Points in einem Sprint. Die Unterschiede solcher Messungen werden also zwischen den Methoden deutlich sichtbar. Während Kanban weitreichende Aussagen trifft ob ein Kanban-System funktioniert und auch ziemlich exakt zeigen kann wie die Durchfluss-Leistungen zum Zeitpunkt sind, kann die Velocity aus Scrum nur am Ende des Sprints die Anzahl der Story-Points summieren.

WIP ALS QUALITÄTSKONTROLLE

David J. Anderson beschreibt in [10, S.31] durch den Vergleich von Cumulative-Flow Diagrammen verschiedener Teams , das ein Zusammenhang zwischen WIP und der Qualität gelieferter Features besteht.

Zitat in [10, S.34]:
„Es besteht ein kausaler Zusammenhang zwischen der Menge an WIP und der durchschnittlichen Durchlaufzeit – und dieser Zusammenhang verhält sich linear. In der Fertigungsindustrie ist dieser Zusammenhang als Littles Gesetz bekannt. Es besagt, dass in einem Produktionssystem, das alle Aufträge abarbeiten kann, der mittlere Bestand im System und die mittlere Durchlaufzeit direkt proportional zueinander sind. Je größer der mittlere Bestand im System ist, desto länger bleiben Stücke im System, und umso größer ist die mittlere Durchlaufzeit.“

Weiter beschreibt [10, S.34] das tatsächlich eine etwa sechseinhalbfache Verlängerung der Durchlaufzeit eine mehr als 30-fache Erhöhung der initialen Fehler bedingt. Diese Aussagen bedeuten, das es eine Möglichkeit gibt die Qualität zu regulieren. Der einzige Weg die mittlere Durchlaufzeit zu regulieren ist es den WIP zu begrenzen. Das heisst man begrenzt die Anzahl der Artefakte die sich in einem Bearbeitungsschritt befinden so weit das sich die mittlere Durchlaufzeit vom Anfang bis zum Ende erhöht. Damit steigert sich die Qualität. Grund ist das ein Team eine Grenze hat, bis zu der es parallel Aufgaben in hoher Qualität erledigen kann. Den WIP begrenzen fokussiert das Team auf die Aufgaben. Jedes Team wird unterschiedlicher WIP-Begrenzungen

einführen. Dies gilt es herauszufinden. Die WIP-Begrenzung [10, S.121] ist ein sehr mächtiges Instrument in Kanban-Systemen. Die WIP-Begrenzungen bleiben nicht statisch. Sie verändern sich über die Zeit und werden je nach Teamgröße und Veränderungen im Team angepasst werden müssen.

2.3 Menschen in Projekten

GRUNDSÄTZLICHE BETRACHTUNG

Menschen [15, S.67] sind Individuen. Trotzdem gibt es Strömungen von ähnlichen Verhaltensmustern. Vor allem in Arbeitsumgebungen aber auch im privaten Umfeld zeigen sich ähnliche Charaktere. In unserer westlichen Welt üblich werden Menschen zu Wettstreitern erzogen. Überall gibt es die Rangordnung. Nirgendwo wird dies ersichtlicher als in hierarchisch organisierten Unternehmen.

Aus der Sicht des Software-Projektmanagement stellen sich Hierarchien oft als sehr komplex und problematisch dar. Dies beginnt beim Recruiting, wo oftmals mehrere Ebenen und Lieferanten von Dienstleistungen vorzufinden sind. Dies gilt vor allem für größere und langfristige Projekte. Interne und externe MitarbeiterInnen aus den verschiedensten Bereichen treffen aufeinander und bilden Teams. Diese Teams sollen möglichst profitabel arbeiten und das über lange Zeiträume. Das dies oftmals nicht ohne Spannungen abläuft ist natürlich auch in den Projekterfolgen zu sehen. Momentan scheint sich ein Bild zu ergeben das bei der Team-Bildung viel zu sehr auf die technischen Fähigkeiten Rücksicht genommen wird. Diese Größe fließt zu stark in die Selektion ein. Um diese Größe zu relativieren ist ein weitergehender Blick auf die Typen von Menschen erforderlich. Vor allem im Hinblick auf die Fähigkeit als Team zu funktionieren und zu kommunizieren.

MENSCHENTYPEN IN SOFTWARE-ENTWICKLUNGS-TEAMS

Zu den sich wiederholenden Typen in klassischen Entwicklungs-Teams haben sich [3] weitreichend Gedanken gemacht. Prinzipiell sollten Fähigkeiten charakterisiert werden. Zur Grundvoraussetzung von Menschen in Software-Projekten gehört, dass Sie das nötige Wissen und die Fähigkeiten technischer Natur mitbringen. Davon ist grundsätzlich auszugehen. Für den Projekterfolg entscheidend ist jedoch die Team-Bildung. Sollen Teams laut den agilen Prinzipien miteinander arbeiten, braucht es einen genauen Blick auf die verschiedenen Eigenschaften. Es gibt viele positive Eigenschaften wie Weitblick, Proaktivität, Entscheidung, Lernen, Kommunikation, Teamfähigkeit, Respekt und weitere. Immer wieder sind Konstellationen in Teams vorhanden die das Team und letztendlich das Projekt entscheidend beeinflussen. Dazu zu zählen währen zum Beispiel Personen die Perfektion um jeden Preis anstreben. Sie sind das Gegenteil von Praktikern. Meist begegnen Sie Praktikern (oder Pragmatikern) mit Unverständnis wenn keine perfekte Lösung gefunden werden konnte. Sie bevorzugen langfristige Planung und stehen dem agilen Prinzip entgegen. Praktiker wissen, das man aus Fehlern und der Erfahrung lernt, und das man in kleinen Schritten besser voran kommt als auf das Große und Ganze zu warten. Praktiker fügen sich besser in das agile Bild ein und sehen auch die Vorteile. Als einen von weiteren Grundtypen beschreiben [3, S.41] den Typus des Diktator. Er bestimmet von Außen oder Innen wie sich das Team zu verhalten hat und welche Methoden es anzuwenden

hat. Es muss nicht erwähnt werden welche negative Folgen dies auf die Teambildung haben kann. Als weiteren Typus benennen [3, S.67] den Code-Held. Er meint um jeden Preis die Code-Herrschaft an sich reißen zu müssen und den meisten Code selbst herzustellen und das möglichst wenig den anderen Teammitgliedern zu kommunizieren. Dies sind nur einige Beispiele wie unterschiedlich Menschen sein können aber der wesentliche Aspekt ist, dies widerspricht den agilen Prinzipien und schadet dem Team und in weiterer Folge dem Projekt. Weitere Personen [1, S.33] können sich nicht von alten Mustern oder Methoden und deren Vorschriften trennen. In einem erneuerten Entwicklungsprozess mit entsprechender Methode kann dies sehr schnell zum Stillstand der Neuerungen führen. Ein sehr wichtiger Punkt ist die Abschätzung von Zeiträumen bei der Umsetzung. Vor allem in der Projektleitung oder den unteren Ebenen des Managements werden Zahlen gerne mal geschönt um zu verschleiern was tatsächliche Aufwände sind. Team-Mitglieder haben oft keine Chance diese Zahlen zu berichtigen und kommen dadurch schnell ins Hintertreffen mit Ihrer Performance. Es muss also mehr darauf Rücksicht genommen werden wie die einzelnen Charaktere in bestehenden oder zu rekrutierenden Teams beschaffen sind. Es muss ein Weg gefunden werden die Stärken dieser Charaktere zu nutzen ohne die negativen Folgen erleben zu müssen. Für gesunde Teams, die selbst organisierend arbeiten können, muss Homogenität hergestellt werden. Die Team-Mitglieder müssen in der Lage sein auf Augenhöhe miteinander zu kommunizieren und sich zu respektieren. Nur so können die Prinzipien von Nachhaltigkeit gelebt werden und soziales Kapital in Unternehmen aufgebaut werden. Wie oben gesagt wurde in der Vergangenheit zu sehr auf die technischen Fähigkeiten acht genommen. Die Recruiting-Prozesse waren so abgestimmt. Es braucht also eine Reform in der Auswahl der Handelnden Personen.

2.4 Recruiting

GRUNDSÄTZLICHE BETRACHTUNG

Die Recruiting-Prozesse [1, S.435] heutzutage sind oftmals sehr technisch beeinflusst. Es scheint als ob die Profile von Bewerbenden nach möglichst vielen Punkten durchsucht werden, welche mit den technischen Voraussetzungen im Projekt zu tun haben. Nicht nur die grundlegenden technischen Skills werden als bedeutend erachtet sondern auch das Wissen um den Einsatz von Tools. Es gibt also die berühmte Speisekarte aus der man sich die Leckerbissen heraussuchen möchte. Vordergründig scheint dies eine sichere Variante der Selektion zu sein. Sie berücksichtigt jedoch nicht die emotionalen Skills. Dies ist fatal da einer der wichtigsten Punkte für erfolgreiche Projekte die Kommunikation ist. Wie schon in 2.1 benannt sind die Erkenntnisse von [12, S.10], das Projekte nicht an der Technik scheitern sondern an mangelnder Kommunikation. Es scheinen die Unsicherheiten zu sein die diese Art von Recruiting hervorgebracht haben. Man kann von Technikern in der Software-Branche erwarten das sie Ihr Handwerk verstehen. Dies ist die Grundvoraussetzung. Sobald ein MitarbeiterInnen ein Projekt erfolgreich gemeistert hat, kann man davon ausgehen. Doch um hoch komplexe Software-Projekte erfolgreich durchzuführen braucht es viel mehr. Das Verständnis von Tools und Programmiersprachen ist nur ein Teil davon. Wesentlich mehr sind Dinge wie fachliches Verständnis oder Weitblick und Teamfähigkeit gefordert. Vor allem in großen Projekten über längere Zeiträume. Bei durchwegs standardisierten Verfahren und best practices Guides in den bekannten Software-Architekturen und deren Tools ist es viel wichtiger das ein Team in der Lage

ist gegenseitig dieses Wissen auszutauschen und gemeinsam zu nutzen. Unterschiedliche Fähigkeiten technischer Natur bei gleichlaufenden Fähigkeiten im Bereich emotionaler Intelligenz und Kommunikationsfähigkeit sind der Schlüssel zu einem erfolgreichen Team. Dies ist einer der Schlüssel dazu. Es braucht noch mehr, vor allem im Hinblick auf äußere Einflüsse.

NEUES RECRUITING

Um die erstrebten Neuerungen in Bezug auf Nachhaltigkeit und Prinzipien in Projekten auch leben zu können ist die Auswahl der entsprechenden Teams elementar. Würde man nur die fachliche Qualifikation beurteilen wie dies oft bei Interviews der Fall ist wo Chefentwickler oder Software-Architekten high-level Abfragen anstellen um die fachliche Stärke der Bewerber festzustellen werden natürlich immer jene gewinnen, die dieses meist gespeicherte und angelernte Wissen direkt abrufen können. Oftmals ist auch bekannt, was bei Interviews so gefragt wird. Darüber reden zu können heißt nicht es auch es umsetzen zu können. Und meistens erwirbt man dann einen Bewerber der möglicherweise der Kategorie Perfektionist [3, S.89] oder Code-Held [3, S.67]. Die Auswirkungen solcher Charaktere auf die Teams sind meistens gegen die agilen Prinzipien. Die Frage ist, wie man Bewerber erkennt, die die Stärke haben sich rasch an veränderte Umgebungen anzupassen, gerne im Team sind und als Team agieren. Respekt gegenüber Anderen als hohes Gut einschätzen und wissen, dass ein Team mit durchschnittlichen Fähigkeiten mehr erreichen kann als ein Team mit ein oder zwei Perfektionisten. Es gibt ein Fallbeispiel eines Projektes bei Motorola von [10, S.31] wo zwei von den Positionen her identische Teams, jedoch von der fachlichen Qualifikation total unterschiedliche Teams überraschende Ergebnisse lieferten. Zitat [10, S.31]*"Beide Teams arbeiteten nach der Methode Feature Driven Development (FDD) und hatten in etwa dieselbe Größe – ungefähr acht Entwickler, einen Architekten, bis zu fünf Tester und einen Projektleiter"*. Weiter in Zitat [10,S.33] *„Dem OTA-Download-Server-Team fehlte es an der Disziplin oder vielleicht am Engagement, um FDD korrekt anzuwenden. Sie arbeiteten nicht gemeinschaftlich, so wie es FDD verlangt. Sie gaben ganz große Mengen an Features auf einmal an die Entwickler". Weiter in* Zitat [10, S.33] *„Das OTA-GM-TEAM wendete FDD nach dem Lehrbuch an. Sie arbeiteten gut zusammen. Sie erstellten Unit-Tests für jede Funktionalität. Und was ganz wichtig ist: Sie arbeiteten nur an wenigen Features gleichzeitig, üblicherweise befanden sich für das ganze Team nie mehr als fünf bis zehn Features gleichzeitig in Bearbeitung"*. Dazu kommen die Ergebnisse, Zitat [10, S 3 3] *„Das OTA-Download-Server-Team benötigte eine durchschnittliche Durchlaufzeit von circa drei Monaten..." und „Das OTA-GM-Team benötigte eine durchschnittliche Durchlaufzeit im Bereich von fünf bis zehn Tagen..."*. [10] liefert ebenfalls einen Hinweis auf die Fähigkeiten der einzelnen Teams in Zitat[10, S.35] *„Wenn man sich die Lebensläufe angesehen hätte, so hätte man festgestellt, dass das OTA-Download-Team dem OTA-GM-Team durchschnittlich drei Jahre an Erfahrung voraus war. Auf dem Papier sprach alles dafür, dass das OTA-Download-Team besser war. Und zu diesem Zeitpunkt waren einige der Teammitglieder tatsächlich davon überzeugt – trotz der zahlreichen Beweise für das Gegenteil"*.

Ein durchschnittliches Team erreichte also wesentlich mehr, weil es gelernt hat als Team zu agieren und die Vorteile zu erkennen. Recruiting-Methoden der Vergangenheit hätten möglicherweise die Qualität dieses Teams und seiner Mitglieder

absolut unterschätzt und solch ein erfolgreiches Team gar nicht zusammengestellt da es nicht sichtbar gewesen wäre. Aus diesem Grund sollte das Bewerbungsgespräch unbedingt zwei Teile aufweisen. Teil 1 ergibt den Level an fachlicher Qualifikation. Darin sollte man das grundlegende Maß setzen und auch den Bereich „Nice to have“ . Bewerber welche Teil 1 erfüllen sollen in einem zweiten Teil interviewt werden. Diesen zweiten Teil der Befragung sollten allerdings nur Personen durchführen welche die Bereiche Psychologie, Soziologie und soziale Intelligenz beurteilen können. In diesem Teil ist es wichtig herauszufinden, ob die Personen als Team agieren können und das Bewusstsein für diesen Umstand vorhanden ist. Nur die Frage ob jemand Teamfähig sei, wird zu wenig sein. Eine Möglichkeit ist es, erfahrene MitarbeiterInnen welche die nötige Qualifikation mitbringen an den Recruiting-Prozessen zu beteiligen. Zum Beispiel kann es ein Teamleiter oder Coach sein, der in den weiteren Gesprächen herausfindet welche Qualitäten in den BewerberInnen zu finden sind. Letztendlich sollte das Ziel sein, eine weite Basis pro Team herzustellen. Unterschiedliche Fähigkeiten in einem Team vereint können von Vorteil sein. In den agilen Methoden wird auf selbstorganisierende Teams gesetzt. Wenn diese Teams in Ihrer kleinsten Struktur breit aufgestellt sind, können sie effektiv agieren und Wissen direkt austauschen.

2.5 Das Team

Folgen wir den agilen Prinzipien dann sollen Teams [5, S.41] selbstorganisiert sein. Egal welche PM-Methode man anwendet, die meisten agilen Methoden streben danach. Die Selbstorganisation ist der zentrale Faktor wenn Teams auch als Team agieren wollen. Diese Selbstbestimmtheit ist es, welche das Selbstbewusstsein und damit einhergehend das Teambewusstsein stärkt und soziales Kapital aufbaut.

Es ist meistens ein Kulturwandel nötig, da in unseren Breiten immer nach Hierarchien gestrebt wird. Es ist von essentieller Bedeutung, dass die Mitglieder eines Teams erkennen, dass man am erfolgreichsten ist, wenn man an sich und die Gruppe denkt. Dann partizipieren Alle in positiver Weise. Ein sich selbst organisierendes Team schafft auch Ordnung. Es hat die Möglichkeit Strategien zu ändern oder anzupassen. Teams werden von außen beeinflusst. Man kann die Arbeitsweise eines Teams ändern mit den Mitteln der Organisation. Aber man sollte sich bewusst sein das jede Änderung, von außen verordnet, im Team Unruhe erzeugt [10, S.27]. Man erklärt dem Team mit der momentanen Arbeitsweise nicht einverstanden zu sein und damit implizit, dass die momentane Leistung nicht wertgeschätzt wird. Dies produziert Angst und Unsicherheit. Alles Faktoren die keinem Projekt dienlich sind. Es ist also erforderlich einen Weg zu finden womit man Teams darlegen kann, welche Möglichkeiten es gibt, sich anders zu organisieren. Wenn das Bewusstsein im Team so groß ist, das Änderungen möglich sind und später auch die Vorteile erkannt werden hat man den Klimawandel geschafft. Teams die diesen Zustand erreichen wirken auch befruchtend auf andere Teams. Kanban hat genau diese Zustandsänderungen zum Ziel. Dies wird erreicht durch mehrere Maßnahmen wie auf Qualität zu fokussieren und den WIP zu reduzieren. Diese beiden Maßnahmen können von Teams leicht aufgenommen werden und damit positive Ergebnisse erzielt werden. Weitere Schritte können folgen. Dazu weiter unten mehr in der Beschreibung für den Einsatz. Teams sind am Leistungsfähigsten, wenn sich ihre Mitglieder sicher fühlen können und das Work-Life-Balance Verhältnis stimmt. Diese Gefühle kann das Management steuern, wenn

es dementsprechend die Firmenkultur in positiver Weise anpasst. Agile Vorgehensweisen erwarten, dass man energiegeladen arbeitet. Dies ist möglich wenn keine unnötigen Überstunden gemacht werden oder ständig ein Gefühl des Druckes vorherrscht.

2.6 Geschäftsführung und Management

Die Grundlage jedweden Agierens im Unternehmen legt meist die Geschäftsführung und das Management. Wenn die Basis oder Unternehmenskultur nicht stimmt, kann sich in den Ebenen darunter nichts verändern. Als Basis ist gemeint, dass die Unternehmenskultur angepasst wird in Richtung selbstorganisierender Teams. Unternehmen der Gegenwart und Ihre Projekte unterliegen oft komplexen Hierarchien. Um diese Komplexität beherrschen zu können werden immer mehr Messinstrumente eingesetzt um die Leistung der einzelnen Bereiche feststellen zu können und PM-Methoden abgewandelt angewandt um zur eigenen Sicherheit irgendwelche Zahlen in Erfahrung zu bringen die Aufschluss geben soll, wie der Status eines Projektes ist.
Dies ist der Fehlgedanke des Management, nämlich zu glauben das sich hoch komplexe Projekte genau planen oder vorhersagen lassen und es nur die nötigen fachlichen Qualifikationen braucht um erfolgreich zu sein. Es ist auch oft von aussen schwer abzuschätzen wie sich Aufwände während der Arbeit gestalten. Vor allem wenn man nicht direkt in Software-Projekten aktiv ist sondern im Umfeld. Die Unsicherheit des Managements verlangt nach Kontrolle. Diese Kontrolle erzeugt Overhead in unnötigem Maße und verlangsamt Teams in Ihrer Arbeit. Die Strukturen werden starrer, Flexibilität geht verloren die Aufwände steigen. Kommunikation verläuft oft über viele Ebenen sehr komplex. Vorhandene Muster werden selten abgelegt und dem Wunsch nach Eigenverantwortlichkeit oder Verbesserungswünschen meist nicht nachgekommen. In solchen Unternehmen haben die Handelnden oftmals sehr viel Energie aufzuwenden um Ihre Positionen oder Handlungsweisen zu schützen. In einem rasanten Marktbereich wie der Software-Entwicklung und sich ständig ändernden Umgebungen ist diese Vorgangsweise nicht zielführend. [10, S.19] beschreibt diese Veränderung der Unternehmenskultur als Kultur der kontinuierlichen Verbesserung, die Kaizen-Kultur. Innerhalb dieser Kultur sind die MitarbeiterInnen ermächtigt selbständig Entscheidungen zu treffen. Die Angestellten kennen in dieser Kultur keine Angst. Fehler werden vom Management akzeptiert solange erneute Verbesserungsversuche der Prozessverbesserung dienen. Die Mitglieder sind ermächtigt Ihre Arbeit selbst zu organisieren. Es sind überall visuelle Kontrollen vorhanden (Kanban-Board) und für jedermann ersichtlich. Diese Kultur fördert in hohem Maße das soziale Kapital des Unternehmens. Es ist auch eine Kultur von hoher Kollegialität. Diese Art der Unternehmenskultur verflacht auch die Hierarchien. Es gibt weniger Overhead und die Gesamtorganisation wird übersichtlicher, flexibler und stärker in Ihrem Bewusstsein. Alles positive Faktoren welche konform mit den agilen Prinzipien gehen. In der westeuropäischen Sichtweise passt diese Unternehmenskultur oftmals nicht in das Gedankengut. Bei näherer Betrachtung überwiegen jedoch die Vorteile immens. Dazu ist es notwendig die grundlegenden Fragen zu stellen und Sie zu beantworten. Wie schon in 2.1 angeführt aus den Statistiken der Standish-Group, gibt es Erfolgsfaktoren die gemessen wurden. Die agilen Methoden beschreiben in Ihren Werten und Prinzipien gleichlautend diese Erfolgsfaktoren und bestätigen dadurch Ihre Vorgehensweise. Diese Ansicht ist jedoch sehr grob granulär und muss tiefergehend betrachtet werden um dem Management die nötige Information zu geben

Entscheidungen bezüglich Unternehmenskultur, Projekt-Management und Personalführung korrekt treffen zu können. Gemein kann gesagt werden das Unternehmen die sich mit agilen Methoden beschäftigen, auch Ihre Unternehmensentwicklung und die zugehörige Struktur um Auge haben. Ein Wandel im Wesen des Projektmanagement bedingt oftmals einen Wandel im gesamten Unternehmen

[13] beschreibt die Wege des Change-Management im Übergang von Ist zu Soll und den Versuch klassische Projekt-Management Methoden anzuwenden:
Zitat [13, S.7]:
„Einen Lösungsansatz stellt das klassische Projektmanagement dar. Es zeigte sich jedoch, dass trotz professionellem klassischen Projektmanagement die Transformation vom Ist- zum Soll-Zustand oft nicht erfolgreich bewältigt werden kann. Analysen haben ergeben, dass das Projektmanagement zu technisch orientiert ist und die menschlichen Aspekte eines sozialen Systems nicht ausreichend berücksichtigt“

und Zitat [13, S.8]:
„Eine Grundhaltung des Change-Management ist, dass es besser ist, aus Erfahrungen zu lernen, anstatt um jeden Preis Fehler zu vermeiden.“

Diese Aussagen stehen in Korrelation zu den agilen Prinzipien und vor allem den Werten hinter Kanban oder XP.

3 Ein neuer Ansatz auf Basis „Balancing Team Building“

Aus der Recherche und den grundlegenden Erhebungen ist ein Bild entstanden, welches eine Notwendigkeit ergibt, dass in Software-Projekten eine effektivere Projekt-Management Methode entworfen wird. Wenn man die handelnden Personen und Ihre unterschiedlichen Funktionen in Relation betrachtet, ergibt sich auf Basis der agilen Methoden ein Ansatz. Dazu müssen die nötigen Werkzeuge, die Rollendefinition und eine allgemeine Beschreibung entworfen werden. Diese Arbeit versucht dieses System ausreichend darzustellen. Vor allem soll es möglich sein das sich Teams selbst ausbalancieren können, zum Beispiel durch Verschiebung von Ressourcen. Die Grundlage solcher Entscheidungen sollen geschaffen werden durch die Aussagekraft der verwendeten Hilfsmittel zur Selbstreflexion.

3.1 Grundlagen und Motivation

Aus eigener Erfahrung in mehrjährigen großen Projekten, aus der Befragung, sowie der Studie von Fakten aus dem Bereich Software-Entwicklung ergibt sich ein vielschichtiges Bild. Die Prozesse innen und Außen rund um ein Software-Projekt sind komplex. Die Beziehungen zwischen Kunde, Management des liefernden Unternehmens sowie den Entwicklungs-Teams stehen in einem Dreieck. Es wurde festgestellt, das direkte Kommunikation einer der wesentlichsten Erfolgsträger ist. In erfahrenen Projekten scheint diese Grundlage jedoch noch nicht richtig angekommen zu sein. Entwickler-Teams haben oftmals keinen direkten Kontakt zu den Informationen des Kunden. Sie haben auch meist keinen direkten Kontakt zum Management, da meist mehrere Management-Ebenen über Ihnen aufgetürmt sind. Selbst die Kommunikation zwischen Teams (oft nur einen Raum weit getrennt) scheint nicht richtig zu funktionieren. [1, S.206] zitiert:

„Die meisten Teams sind gar keine Teams, sondern einfach eine Ansammlung von Einzelbeziehungen zum Chef, bei denen jede Person mit der anderen um Macht, Prestige und Position eifert“ (Douglas McGregor)

Dieses Zitat beschreibt sehr humorvoll, wie wir in geschäftlichen Dingen erzogen sind. Die bestehenden Strukturen fördern meist diese Umstände zusätzlich durch den Einsatz überzogener Hierarchien. Ein weiterer Erfolgsfaktor für den Einsatz von agilen Methoden wurde beschrieben mit *„Teams sollen selbständig organisiert sein, und die Mitglieder respektvoll miteinander umgehen“* . Wie schafft man es aus einem Team ein Team zu machen? Es beginnt ganz einfach beim Recruiting. Wenn in diesem Bereich die falschen Entscheidungen getroffen werden, können keine Teams entstehen. Die Betrachtung von 80% (oder mehr) fachlicher Qualifikation als Einstiegskarte in ein Software-Projekt ist viel zu kurzsichtig gegriffen. Eines der grundlegenden Probleme wurde noch nicht eingehend betrachtet. Es ist die Angst vor Entscheidungen. Jeder ist auf Sicherheit aus, und macht seine Arbeit so da ihm formell nichts vorgeworfen werden kann. Mut wird nicht belohnt. Obwohl dies eigentlich eine Fähigkeit aus agiler Sichtweise darstellt. Genauso wie Veränderung. Diese Angst oder Suche nach Risikominimierung ist es auch, die gute agile Methoden durch unverhältnismäßige Kontrolle oder Messinstrumente oder einfach falsche Anwendung (Estimation) zu Fall bringt. Somit können die besten Methoden nicht zur Wirkung

kommen. Oftmals sind es schon kleine Eingriffe die fatal sind. Ein Kanban-System ist keines, wenn man keinen Anfang und kein Ende definiert, den WIP nicht begrenzt und die Durchlaufzeiten nicht misst. Ein Scrum-Projekt wird scheitern, wenn man schulmäßig alle Daily-Scrums einhält, die Planning-Meetings und die Retrospektiven abhält, jedoch schon bei den Estimation-Meetings herauskommt, dass sich die Schätzungen nicht ausgehen werden. Optimismus beim schätzen ist fatal. Was herauskommt ist, dass die Scrum-Teams durch die ganzen Meetings zeitlich noch eingeschränkter sind als ohnehin bereits. XP wird ohne seine best-practices Ansätze nicht funktionieren. Und FDD wird nicht die richtige Wahl sein, wenn es ein Projekt ist wo bereits im Vorfeld bekannt sein wird, das sich Anforderungen ständig ändern. Eine Management-Ebene die agile Methoden nicht wirklich durchblickt hat und weiß was Sie entscheidet macht es fast unmöglich das sich Teams selbständig organisieren.
Es muss also bereits in der Geschäftsführung angesetzt werden. Eine Geschäftsführung die mit Weitblick eine Unternehmenskultur gestaltet welche leichtgewichtig und offen ist und Teams dabei unterstützt sich zu organisieren und weiter zu entwickeln ist die Grundvoraussetzung. Es ist auch die Geschäftsführung die das Risiko trägt. Kaizen [10, S.57] ist ein Beispiel für eine derart gelagerte Unternehmenskultur. Diese Kultur sollte von oben begonnen werden und durch die Ebenen weiter gelangen. In solch einer Unternehmenskultur würde man auch sehr flache Hierarchien vorfinden. Es gibt auch Beispiele in denen sich eine positive Veränderung eines Teams auf andere Teams weitervererbt hat und schließlich eine grundlegende Änderung der Unternehmenskultur brachte. [10, S.67, Kapitel 5.4 *„Kulturelle Änderungen ist vielleicht der größte Vorteil von Kanban“*] berichtet über solch einen Fall nach dem Einsatz eines Kanban-Systems wo das Top-Management die positiven Entwicklungen sehen konnte nachdem, ausgehend von einem einzigen Team, eine tiefgreifende Änderung in der Unternehmenskultur entstand. Es ist also von grundlegender Bedeutung das im Management Personen agieren die ein Verständnis, und viel wichtiger ein Bewusstsein, aufbringen können was die Veränderungen im Unternehmen auslösen und bewirken können. Im Beispiel von Kanban entwickelt sich in solchen Systemen nachgewiesener Maßen sehr viel soziales Kapital. Nachhaltige Unternehmensentwicklung kann so geschehen. Diese Veränderung im Denken muss natürlich im Unternehmen geschehen. Sie kann aber nur von außen in das Unternehmen gebracht werden. Dazu ist ein objektiver Blick nötig.

Der Einsatz von Veränderungsmaßnahmen bringt natürlich immer Widerstände mit sich. Personen beginnen Ihre Denkweisen und Bereiche zu schützen. Es gilt also einen Weg der positiven Beispiele von Anfang an zu finden um Widerstände zu überwinden. Idealerweise kann ein sanfter Übergang entstehen, wobei mit einem Team begonnen wird, und sich dann weitere Teams anschließen. Eine Art viraler Effekt scheint möglich zu sein, sobald Vorteile des Handelns und der Änderung der Philosophie ersichtlich werden. Dazu braucht es positive Beispiele und es soll an dieser Stelle auch erwähnt werden, das es besser ist in kleinen Teilbereichen zu beginnen als in einem Big-Bang quasi die gesamte Unternehmensstruktur umzubauen. Dazu gibt es positive Aussagen bzgl. der Vorgehensweise sowohl in [1] als auch [10]. [1, S.31] berichtet von einem Unternehmen welches durch Änderungen des Projektmanagement hin zu agilen Methoden im ersten Jahr der Einführung einen Zuwachs von 94% der Feature-Lieferungen hatte und 500% mehr Wert an Kunden auslieferte. [10, S.54] Gibt in Kapitel 4 die Ergebnisse eines Umstellungs-Projektes bekannt bei dem die Teams ein Kanban-System in eine bestehende Projektmanagement-Struktur eingeführt haben.

Das Team hatte es geschafft die Termintreue auf 98% zu steigern den Durchsatz an Anforderungen zu verdreifachen sowie die Durchlaufzeit um 90% zu reduzieren.

3.2 Eigene Erhebungen

Zugängliche Daten (z.b. der Standish-Group [16]) und deren Aussagen zu Erfolgsfaktoren oder Misserfolgsfaktoren wurden Eingangs in Kapitel 2.1 erörtert. Um zum jetzigen Zeitpunkt eine Blick zu bekommen wie die Situation von Mitarbeitenden in laufenden Projekten gesehen wird, war es notwendig eigene Befragungen durchzuführen. Aus diesem Grund wurden MitarbeiterInnen aus verschiedenen Unternehmen befragt. Es wurde versucht verschiedene Branchen zu erfassen um einen breiteren Blickwinkel zu bekommen. Dazu wurde ein Online-Fragebogen erstellt und anschliessend mit dem Statistik-Programm GNU-R auch graphisch ausgewertet.

3.2.1 Online Fragebogen

Um relevante Fragen im Projektumfeld im Vorfeld klären zu können, wurde ein Online-Fragebogen auf www.soscisurvey.de [25] eingerichtet. Der Fragebogen ist im Anhang I ersichtlich. Der Fragebogen war in einem Zeitraum von vier Wochen Aktiv gestellt. Es war eine anonyme Befragung, um für die Befragten die Möglichkeit zu schaffen frei antworten zu können. Der Fragebogen wurde mittels Link per Email/Skype verteilt und war mit einem Passwortschutz versehen. Damit war gewährleistet das keine Personen außerhalb den Fragebogen ausfüllen konnten, welche nicht zum Kreis der Menschen gehörten welche in Software-Projekten beschäftigt waren. Es wurden Menschen befragt welche in Software-Projekten aktiv arbeiten. Es wurden MitarbeiterInnen verschiedener österreichischer Unternehmen befragt. Teilweise mit internationalem Hintergrund. Es waren MitarbeiterInnen aus den Branchen Versicherung/Bankwesen, Telekommunikation, öffentliche Verwaltung und Industrie. Die Tätigkeitsbereiche sowie Alter und Projektgrößen ergeben sich aus der Auswertung. Auf Wunsch der Befragten können die Unternehmen an dieser Stelle nicht öffentlich genannt werden.

3.2.2 Statistische Auswertungen

Grundlegendes zu den Auswertungen

Es waren 38 Datensätze die in die Betrachtung eingeflossen sind. Da es sich um Daten von Personen handelt welche in unterschiedlichen Unternehmen und sowie unterschiedlichen Teams oder Projekten tätig sind, ergibt sich dadurch ein repräsentativer Querschnitt. Das ursprüngliche Ziel von 50 oder mehr Befragten konnte nicht erreicht werden. Möglicherweise war der Zeitraum der Befragung in der Urlaubszeit der Grund. Die Ergebnis-Daten wurden von www.soscisurvey.de als R-Skripte heruntergeladen und mittels GNU-R [26] ausgewertet. Www.soscisurvey.de stellt die Daten automatisiert als GNU-Skripte zur Verfügung. Die grafischen Auswertungen wurden ebenfalls in GNU-R erstellt. Grundsätzliche Abfragen wie Gruppen werden als Items mit entsprechendem Bezeichner in R zur Verfügung gestellt:

c("trifft nicht zu","trifft wenig zu","neutral","trifft zu", "trifft vollständig zu"), levels=c(1,2,3,4,5))

c("keine Kenntnis","kenne ich"), Levels=c(FALSE,TRUE))

c("NEIN","JA"), levels=c(1,2))

Fragestellungen mittels Schieberegler liefern als Beispiel:

attr(data$AG03_01,"1")<-"trifft nicht zu" , attr(data$AG03_01,"101")<-"trifft ganz zu"

Um die Fragestellungen mittels Schieberegler auswerten zu können wurden die ursprünglichen Werte zwischen 1 und 101 in 5-stufige Likert-Skalen mit folgenden Faktoren transformiert:

```
if(sus[[i]][j] >= 1 && sus[[i]][j] <= 10){ sus[[i]][j] <- 1}
else if(sus[[i]][j] >= 11 && sus[[i]][j] <= 31){ sus[[i]][j] <- 2 }
else if(sus[[i]][j] >= 32 && sus[[i]][j] <= 70){ sus[[i]][j] <- 3 }
else if(sus[[i]][j] >= 71 && sus[[i]][j] <= 91){sus[[i]][j] <- 4}
else if(sus[[i]][j] >= 92 && sus[[i]][j] <= 101){sus[[i]][j] <- 5}
```

Die Werte-Bereiche wurden so gewählt das „trifft nicht zu“ von 1-10 getroffen hat und ein „trifft vollständig zu“ von 92-101 getroffen hat. Die neutrale Mitte rund um den Median 51 wurde etwas verbreitert auch die Bereiche „trifft wenig zu“ und „trifft zu“ wurden verbreitert. Damit wurde der psychologischen Sicht bei der Fragebeantwortung Rechnung getragen, wo auf jeden Fall die Extreme einen genauen Trefferbereich benötigen und die neutrale Mitte bei Schiebereglern breiter ist. Es wurde das R-Package „Likert“ [27] verwendet um weiterführende Graphiken zur Verfügung stellen zu können um die Daten mittels Likert-Skalen auswerten zu können. Soscisurvey stellt die Daten für GNU-R in zwei Dateien zur Verfügung. Erstens ein R-Skript welches alle auswertbaren Daten in ein Table-Objekt liest. Es werden in diesem Skript auch die Datentypen und Attribute, wie Wertebereiche, korrekt zur Verfügung gestellt. Das Skript fragt nach einer Import-Datei welche als zweite Datei aus Soscisurvey geliefert wird. Nach dem einlesen dieser Daten stehen diese im objekt „data“ zur Verfügung und können mit den GNU-R Methoden weiter ausgewertet werden. Es wurde ein GNU-R Skript entwickelt um die Daten anzureichern. Es wurden alle Bezeichner von Fragen ergänzt (diese stehen im Startskript als Attribute zur Verfügung). Die Befragungsergebnisse (Levels wie „1,2,3,4,5“ oder „TRUE/FALSE“ sowie „NEIN/JA“) wurden mit sprechenden Attributen versehen:

```
 while(i<=ncol(sue)) {
sue[[i]] = factor(sue[[i]],labels = c("trifft nicht zu","trifft wenig zu","neutral","trifft zu", "trifft
 vollständig zu"), levels=c(1,2,3,4,5))
i <- i + 1
}
 i <- 1
while(i<=ncol(sue)) {
sue[[i]] = factor(sue[[i]],labels = c("keine Kenntnis","kenne ich"), levels=c(FALSE,TRUE))
i <- i + 1
}
i <- 1
while(i<=ncol(sue)) {
sue[[i]] = factor(sue[[i]],labels = c("NEIN","JA"), levels=c(1,2))
i <- i + 1
}
```

Anschliessend wurden die Daten als Charts geplottet mit den GNU-R Methoden *pie(items)* und *plot(likert(items))*.

AUSWERTUNGEN

Kategorien von Befragten

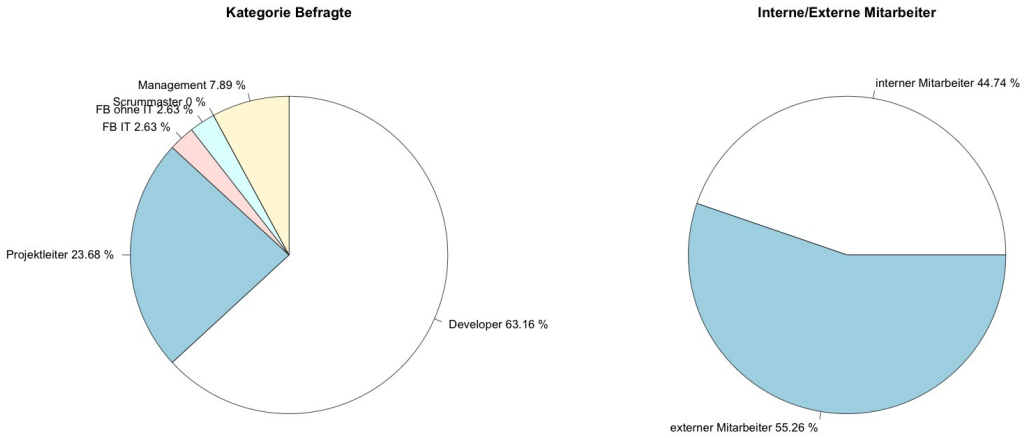


Abb. 9: Kategorie Befragter

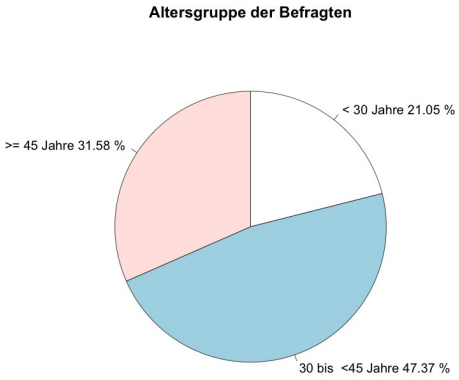


Abb. 10: Altersgruppe Befragter

Größenordnung von Projekten und Teams

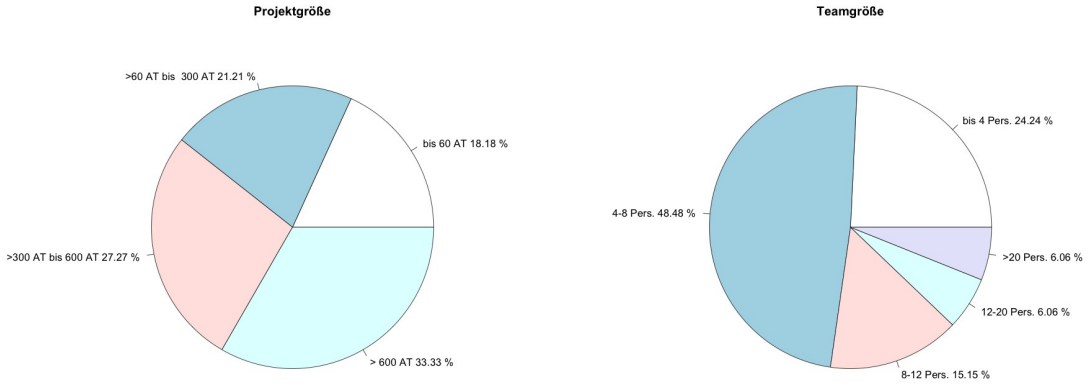


Abb. 11: Größenordnungen im Projekt

Aus der Betrachtung der Erhebung ergibt sich als Profil ein leichter Überhang bei externen MitarbeiterInnen. Dazu muss gesagt werden, dass auch externe MitarbeiterInnen oft sehr lange in Projekten tätig sind, aber durch die breite Streuung von Tätigkeitsbereichen einen sehr großen Erfahrungsschatz in Projekten mitbringen.
Der größte Anteil der Befragten mit gerundet 63% waren die Software-Entwickler, gefolgt von Projektleitung und Management von gerundet 32% und dem Fachbereich (Mit und ohne Bezug zu IT) von gerundet 5%. Es war keine Nennung als Scrum Master, was interessant ist. Da Projektleiter welche Scrum Master sind eher mit Projektleiter antworten sollten, mag dies der Grund sein. Der Zusammenhang Projektleiter als Scrum Master wird weiter unten erläutert. Der überwiegende Anteil von Befragten rund 79% war über 30 Jahre alt. Dies spricht ebenfalls für eine repräsentative Befragung da MitarbeiterInnen in diesen Alterskategorien die größere Erfahrung mitbringen. Die Befragten waren zum größeren Teil von gerundet 60% in langfristigen Projekten tätig (Bereich 300 Personentage und größer). Die Teamgrößen ergeben kein eindeutiges Bild. Mit rund 48% sind die Teams ideal groß, der überwiegende Anteil von rund 52% hat entweder zu kleine oder zu große Teams. Teamgrößen bis vier Personen mag dem Umstand geschuldet sein, dass doch rund 18% der Projekte bis maximal 60 Personentage abgefragt wurden, womit die Teamgröße plausibel erscheint. Ein wesentlich bedenklicher Umstand sind die großen Teams die größer als 8 Personen umfassen mit rund 27%.

Beurteilung des Projekterfolges

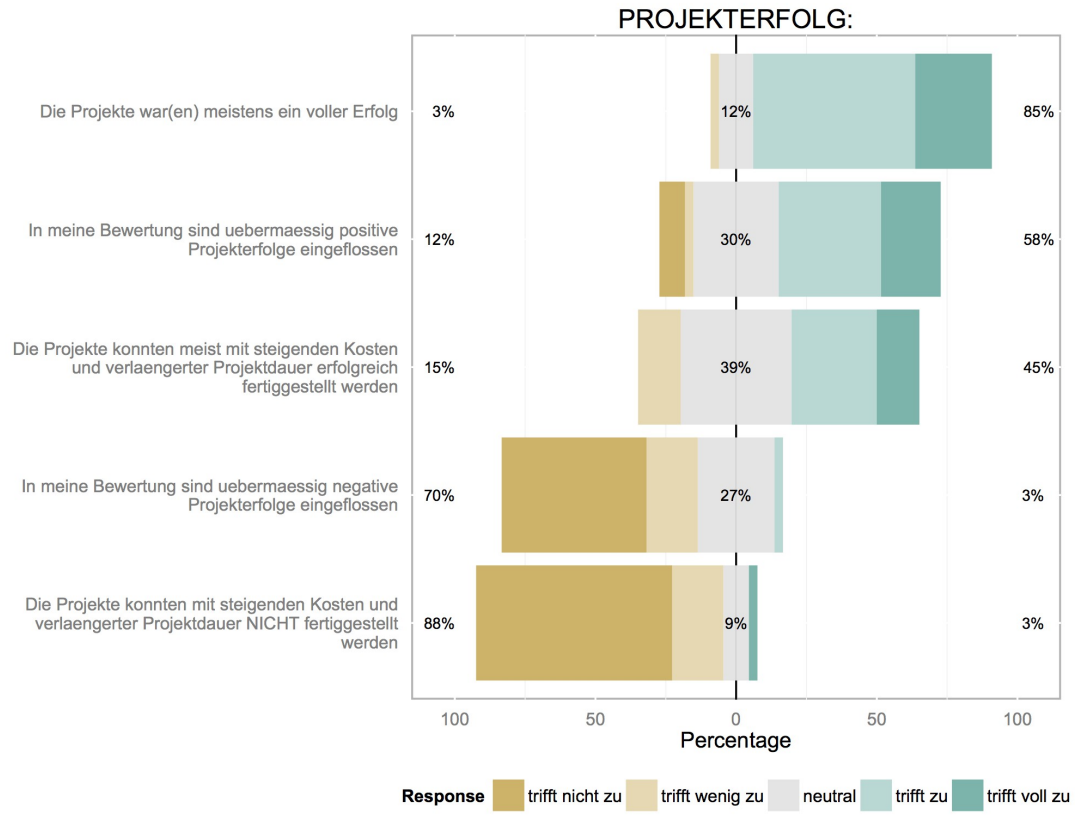


Abb. 12: Auswertung Projekterfolg

Ein wichtiges Ziel der Befragung war die Beurteilung der Projekterfolge. Es wurde abgefragt wie die Sichtweise über die letzten Projekte oder dem laufenden Projekt war. Die Befragung über Projekterfolge ergab,dass überwiegend positive Projekterfolge beurteilt wurden. Jedoch bewertet nur ein kleiner Teil der Befragten die Projekte als vollen Erfolg. Ebenfalls nennenswert ist die Beurteilung der Projekte welche mit steigenden Kosten und verlängerter Projektdauer fertiggestellt wurden. Dieser Anteil scheint für immerhin mit rund 45% gegeben zu sein. Generell ergibt sich ein Bild, welches grundsätzlich nicht negativ ist, aber jedoch weit unter dem liegt wie Software-Projekte laufen sollten. Es wurden mehrheitlich positive Projekte bewertet. Ebenfalls gibt es einen kleinen Teil von Projekten die nicht fertiggestellt werden konnten (Angabe mit 3%). In dieser Aussage ist nicht die Größe der Projekte vorhanden welche erfolgreich oder nicht erfolgreich waren.

Verteilung der Fähigkeiten der handelnden Personen

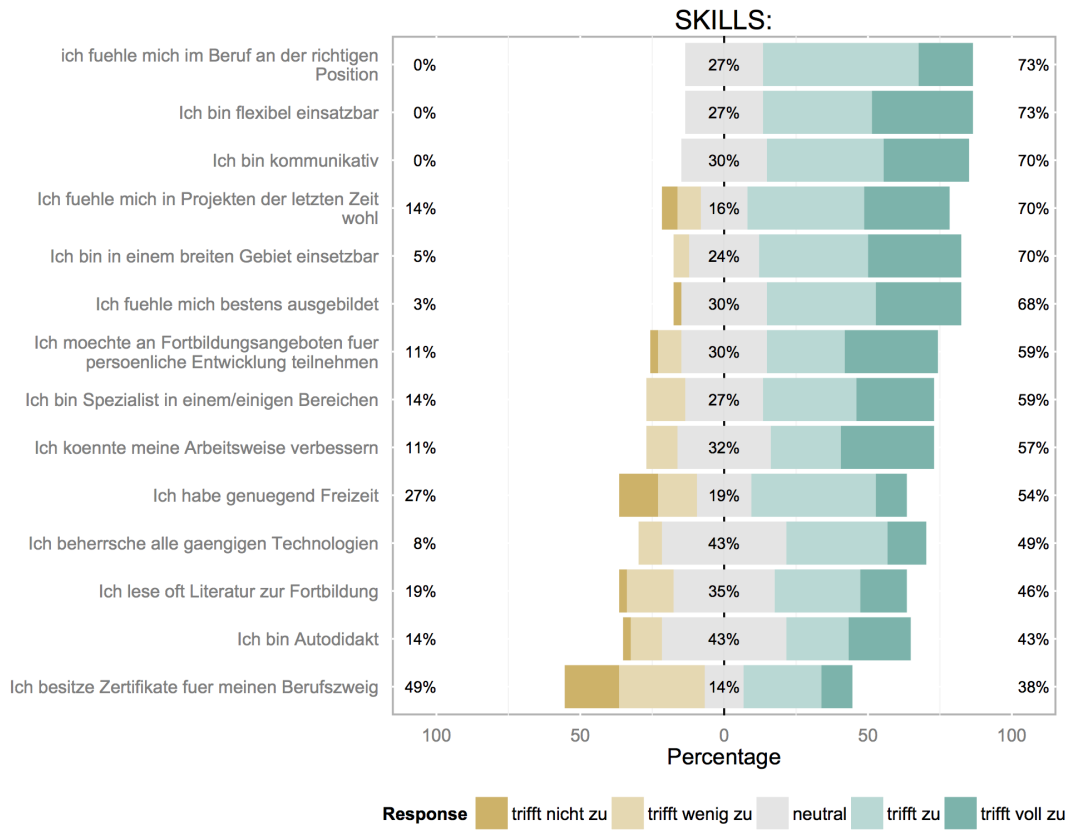


Abb. 13: Auswertung Skills

Grundsätzlich haben die Befragten ein positives Bild zu Ihrem Berufsstand sowie Ihrem Ausbildungsgrad oder im Umgang mit Freizeit, Fortbildung und Wohlfühlen. Die grundsätzliche positive Stimmung passt auch damit zusammen das eine grundsätzliche positive Haltung bei den Projekterfolgen angegeben wurde. Das der Fortbildungsgrad gut ist, sowie die Flexibilität positiv hervorsticht, ist wohl dem Umstand zu schulden das der größere Teil der Befragten externe MitarbeiterInnen sind. Dazu passen auch die Aussagen zu Ausbildungsgrad und Einsatzbreite. Ebenfalls bemerkenswert ist der Umstand, das die Frage nach dem Fortbildungsangebot in persönlicher Entwicklung mit 60% relativ hoch ist. Offensichtlich besteht der Wunsch und das Bewusstsein das emotionale Skills für die Arbeit wichtig sind. Dazu passt auch der Umstand das ein erheblicher Teil die eigene Arbeitsweise verbessern möchte, sowie die eigene Kommunikationsfähigkeit welche rund 70% als eher positiv bewerten. Wie in der Problemanalyse bemerkt ist einer der Erfolgsfaktoren Kommunikation. Da rund 70% der Befragten sich kommunikativ als positiv darstellen, scheint dieses Bild auch zu den positiv gesehenen Projekterfolgen zu passen. Ein weitere Erfolgsfaktor wurde angegeben als Menschen, welche gut ausgebildete emotionale Skills besitzen. Der Wunsch von rund 60% der Personen diese Fähigkeiten weiter auszubilden, bestätigt ebenfalls das Bild positiv gesehener Projekterfolge.

Aussagen über das Projekt-Management

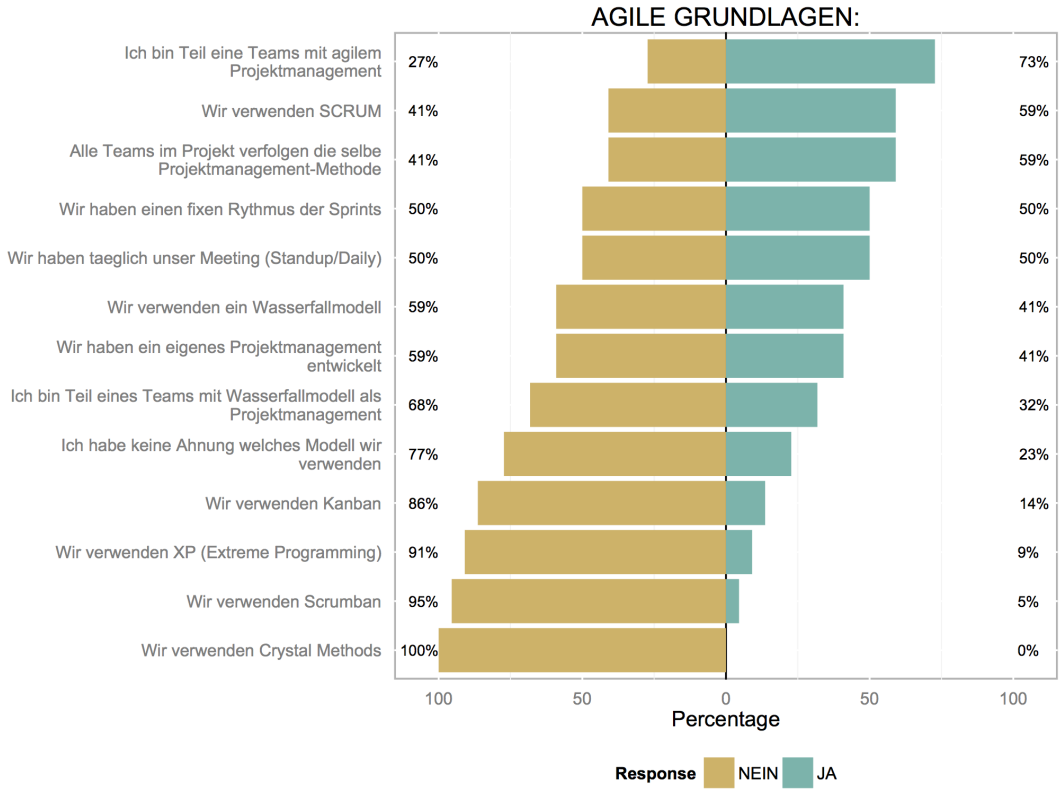


Abb. 14: Aussagen über Projekt-Management

Es war notwendig zu erfahren wie weit agile Methoden bereits angewandt werden. Grundsätzlich scheinen die Befragten mehrheitlich in Projekten mit agilen Projekt-Management-Methoden zu stehen (rund 73%). Die Mehrheit dieser Gruppe verwendet Scrum als Projektmanagement-Methode. Jedoch ist auch ersichtlich, dass die verwendeten Methoden nur zu 50% einen fixen Rhythmus der Sprints haben und nur 50% der Befragten in Projekten arbeiten, welche alle die gleiche Projektmanagement-Methode verwenden. Immerhin 41% verwenden ein Wasserfallmodell sowie eigene entwickelte Management-Methoden. Wie weiter unten ersichtlich wird, ist dies kein Zufall sondern hat vor allem bei agilen Projektmanagement-Methoden Methode, da vor allem im Bereich Scrum ein Mischsystem verwendet wird. Agile Projektmanagement-Methoden werden oft nicht in Reinkultur gelebt. Über die Auswirkungen dieser Vorgangsweise wird weiter unten noch ausführlich eingegangen. Kanban und „best-practices“ Methoden wie XP oder Scrumban (Seite 19) werden momentan nur zu einem kleinen Teil eingesetzt. Crystal-Methods (Seite 18) ist nicht verwendet worden. Man darf nicht vergessen das Crystal-Methods eine sehr abstrakte Methode ist und heutzutage keine große Verbreitung in Projekten hat.

Grundwissen über agile Vorgehensweisen

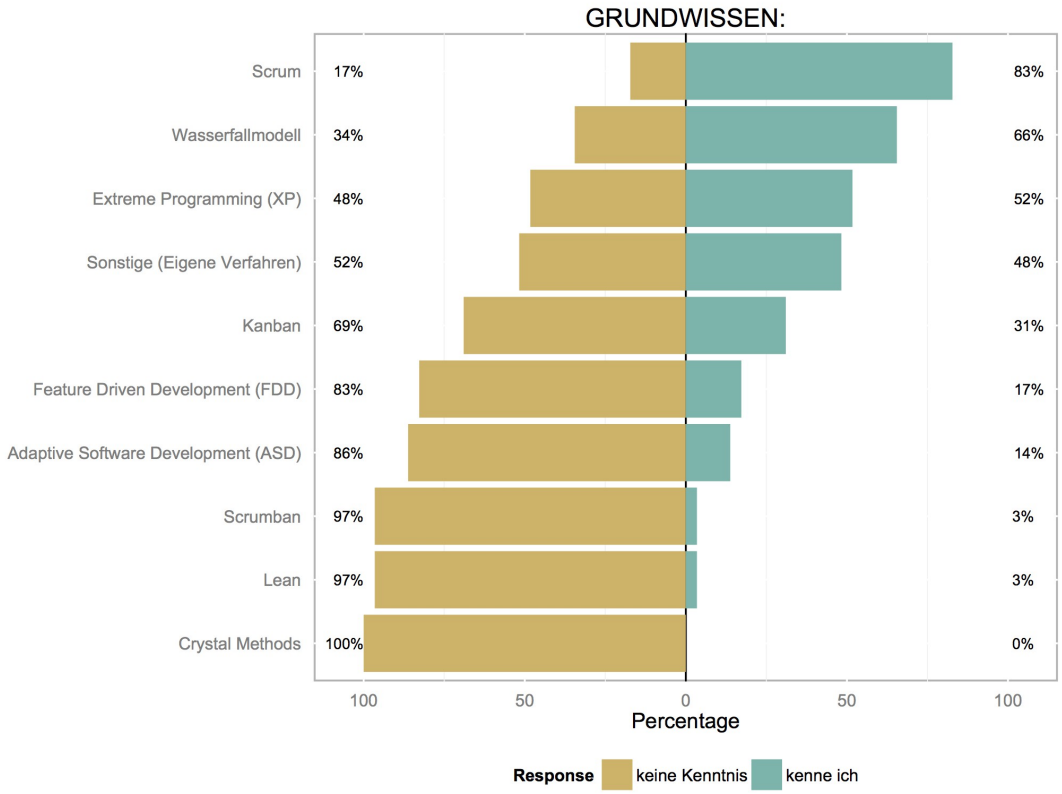


Abb. 15: Grundwissen über agile Methoden

Bei der Abfrage über agile Vorgehensweisen ist wie erwartet Scrum der eindeutige Führer gefolgt vom Wasserfallmodell. Auch XP ist bei rund 50% der Befragten ein Thema und Kanban bei immerhin 31% der Befragten. Dies ist eine Überraschung da Kanban als Projektmanagement-Methode noch nicht in Europa wirklich angekommen ist. Jedoch haben in [Abb. 15] immerhin 14% angegeben Kanban zu verwenden. FDD und ASD sind noch bekannt hingegen ist Crystal Methods kein bekanntes Thema. Auch dieser Abfrage passt ins Bild, wenn man berücksichtigt, dass in unseren Breiten die agilen Vorgehensweisen in der Softwareentwicklung, weitestgehend mit Scrum in Verbindung gebracht werden. Über diesen Umstand wird weiter unten noch ausführlich diskutiert werden, da es ebenfalls sehr wichtig ist zu betrachten, was Scrum genau ist. Leider werden in Softwareprojekten meistens abgewandelte Formen von Scrum angewandt, welche mit eigenen Ideen zur Kontrolle und Risikominimierung gepaart werden. Diese Umstände machen leider grundsätzlich gute Ideen von Scrum obsolet. Wie bei all diesen agilen Vorgehensweisen kann man die Wirkung nur dann vollumfänglich verstehen und erleben, wenn man sich auch an die Grundgedanken hält. Ein Reizthema in dieser Frage ist zum Beispiel „Estimation“. Oftmals wird Estimation unter zu Hilfename von zeitlichen Größen für die Aufwände betrieben, was aber grundsätzlich falsch ist. Kanban lebt „keine Zeitangaben“ in Reinkultur und macht nur Angaben über WIP mehr dazu weiter unten.

Empfindung in Bezug auf agile Methoden

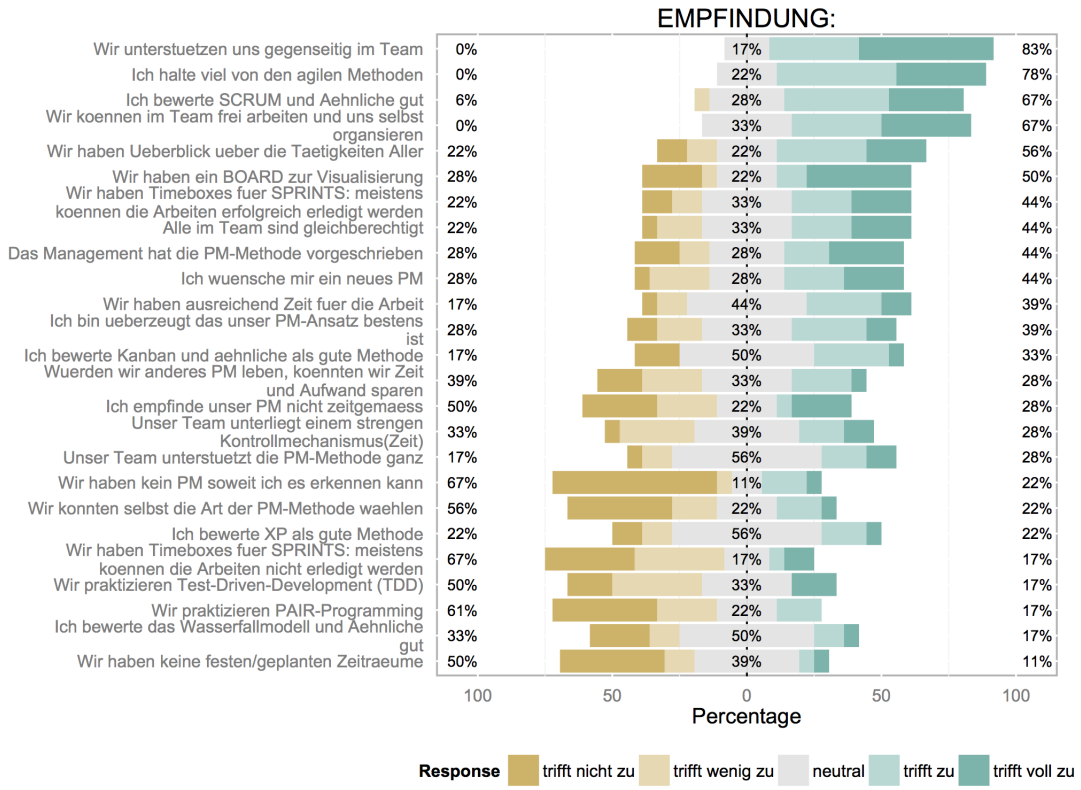


Abb. 16: Empfindung in Bezug auf agile Methoden

Die Befragten geben eine eher positiv gestimmtes Bild zu agilen Methoden ab. Ein Eckpfeiler der agilen Methoden ist die gegenseitige Unterstützung im Team, welche die Befragten mit immerhin 83% positiv bewerten. Der größere Anteil davon ist sogar vollständig davon überzeugt. Scrum bewerten immerhin 67% der Befragten positiv allerdings nur ein kleinerer Teil ist davon vollständig überzeugt. Dies ist ein interessanter Gesichtspunkt. Ebenfalls rund 67% geben an sich selbständig organisieren zu können sowie frei arbeiten zu können. Wenn es aber um Überblick und Visualisierung geht, sind nur mehr rund 56% bzw. 50% die positive Erfahrungen gemacht haben. Sehr interessant ist die Sichtweise der Befragten auf die Länge der Iterationen (Sprints). Nur 44% geben an das Time-boxes existieren, eingehalten werden und auch erfolgreich abgearbeitet werden können. Dies ist ein bekanntes Problem in Verbindung mit Estimation, nämlich der zu optimistischen Schätzung von Aufwänden (Story-points in User-stories) für die Sprints. Ebenfalls nur 44% geben positiver Weise an, das alle im Team gleichberechtigt sind. Dies würde sich mit der gegenwärtigen Verteilung von Rollen im Team in Softwareprojekten decken, welche nur allzu oft die Code-Führer oder andere Hierarchien im Team entwickeln. Ein Nachteil der hinterfragen lässt ob wirklich agile Methoden in Reinkultur gelebt werden. Wenn es um die Fragen ausreichende Zeit geht, bewerten dies nur lediglich 39% als positiv, 44% als neutral und 17% negativ. Dies scheint ein ähnliches Bild zu

sein wie im Bereich der Time-boxes. Die Frage nach einer neuen PM-Methode bewerten 44% positiv gegen nur 28% negativ. Nur 39% bewerten positiv das die momentane PM-Methode bestens ist. Es scheint also keinen nennenswerten Rückhalt für die bestehenden PM-Methoden zu geben. Gleichzeitig fehlt die Idee, was man anders machen könnte, ausgedrückt durch die Fragen wie „Würden wir ein anderes PM leben, könnten wir Zeit und Aufwand sparen“ ‚sowie „Ich empfinde unser PM zeitgemäß“, welche eher neutral bewertet wurden. Die „best-practices“-Ansätze wie bei XP nämlich Pair-Programming oder Test-Driven-Development (TDD) scheinen keinen wesentlichen Rückhalt bei den Befragten zu finden. Diese Methoden werden offensichtlich in den gelebten PM-Methoden nicht gelebt oder erkannt. Dies scheint ebenfalls ein Indiz dafür zu sein, das vielfach nicht wirklich verstanden wird was die agilen Methoden bedeuten. Es kann sich auch um interne Widerstände handeln oder einfach um den Umstand das man ein scheinbares agiles Projekt-Management lebt.

Allgemeine Erfahrungen in Projekten

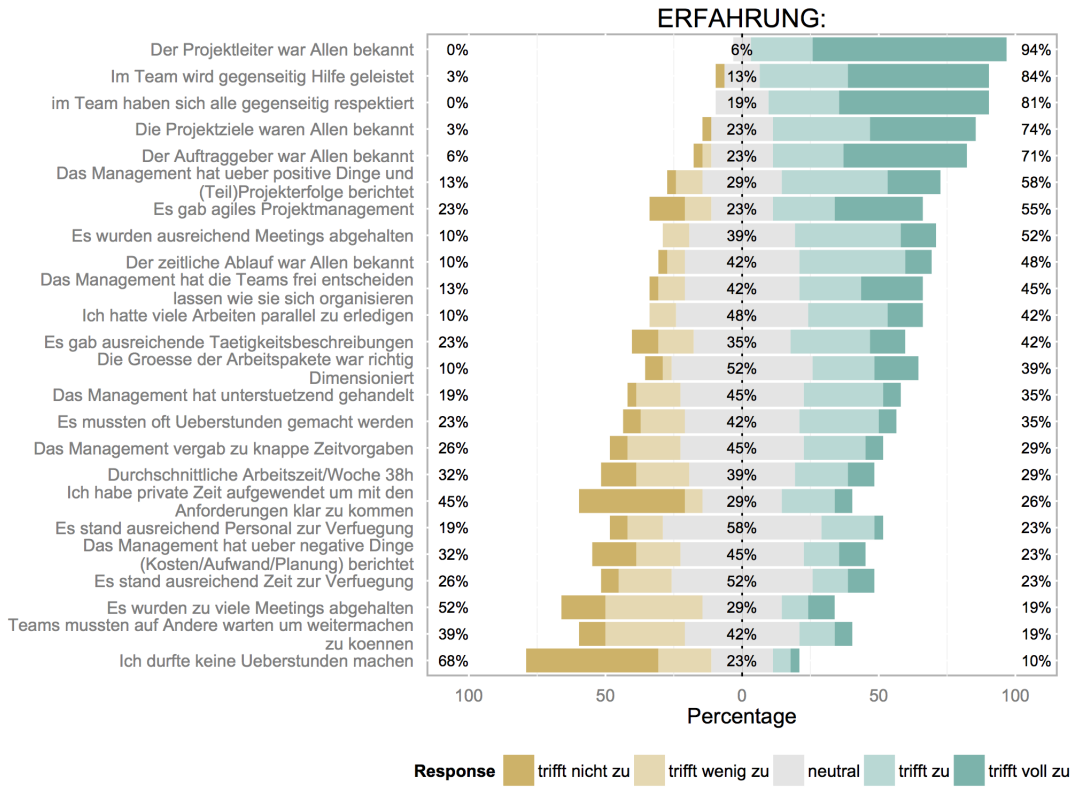


Abb. 17: Erfahrung in Projekten

Der Projektleiter war offensichtlich fast Allen bekannt (94%). Ebenfalls haben die Befragten das Gefühl, dass in der Mehrheit gegenseitig Hilfe geleistet wird und sich die Team-Mitglieder gegenseitig respektieren. Die Projektziele waren aber nur 74% der Befragten bekannt. In etwa der Hälfte davon vollumfänglich. Auch den Auftraggeber kannten die Meisten. Leider konnten nur 58% der Befragten berichten über positive Projekterfolge informiert zu werden. Grund dafür dürfte sein das diese

Art von Projekten momentan eher MitarbeiterInnen beschäftigt, die durch Ihre rein technischen Skills aufgenommen wurden. Möglicherweise kommt dabei die gesamtheitliche Betrachtung der Mitarbeitenden zu kurz. Dies würde auch die Mangelerscheinung positives Feedback und positive Motivation erklären. Auch der zeitliche Ablauf dürfte nur 48% wirklich bekannt gewesen sein. 42% hatten viele Arbeiten parallel zu erledigen und nur ein kleiner Teil von 42% positiver Meinungen konnten ausreichende Tätigkeitsbeschreibungen vorfinden. Nur ein kleiner Teil von 39% konnte behaupten das die Arbeitspakete richtig dimensioniert waren. Dies ist ein sehr wichtiger Aspekt, der weiter unten im Bereich Kanban vs. Scrum noch ausreichend Berücksichtigung finden wird. Die Unterstützung des Managements zu lediglich 35%, und davon kann nur der kleinste Teil behaupten vollumfänglich unterstützt worden zu sein, ist als absolut bedenklich einzustufen. Der Rest der Fragen wurde eher neutral bewertet bzw. ergeben keine Aussage die direkt zu betrachten wäre. Die wesentlichen Punkte in dieser Aussage sind das ein erheblicher Anteil der Mitarbeitenden viele Arbeiten parallel zu erledigen hatten, die Arbeitspakete offensichtlich mehrheitlich nicht richtig dimensioniert waren und die Unterstützung des Managements offensichtlich nicht vorhanden ist. Gerade diese hauptsächlich ins Auge stechenden Fakten stellen Umstände dar, die gegen die agilen Vorgehensmodelle sprechen. Obwohl die meisten Befragten offensichtlich in Projekten mit agiler Vorgehensweise arbeiten sind die Eckpfeiler wie zum Beispiel die Unterstützung des Managements nicht erfüllt. Es ergibt sich daraus eine Diskrepanz zwischen gelebtem und erlebtem. Dies passt zum Bild das nur ein kleiner Teil der Befragten volle Projekterfolge erleben konnte und bestätigt das Bild wie zum Beispiel Aussagen der Standish-Group zu (Miss)Erfolgsfaktoren.

Die Erfahrung im Team

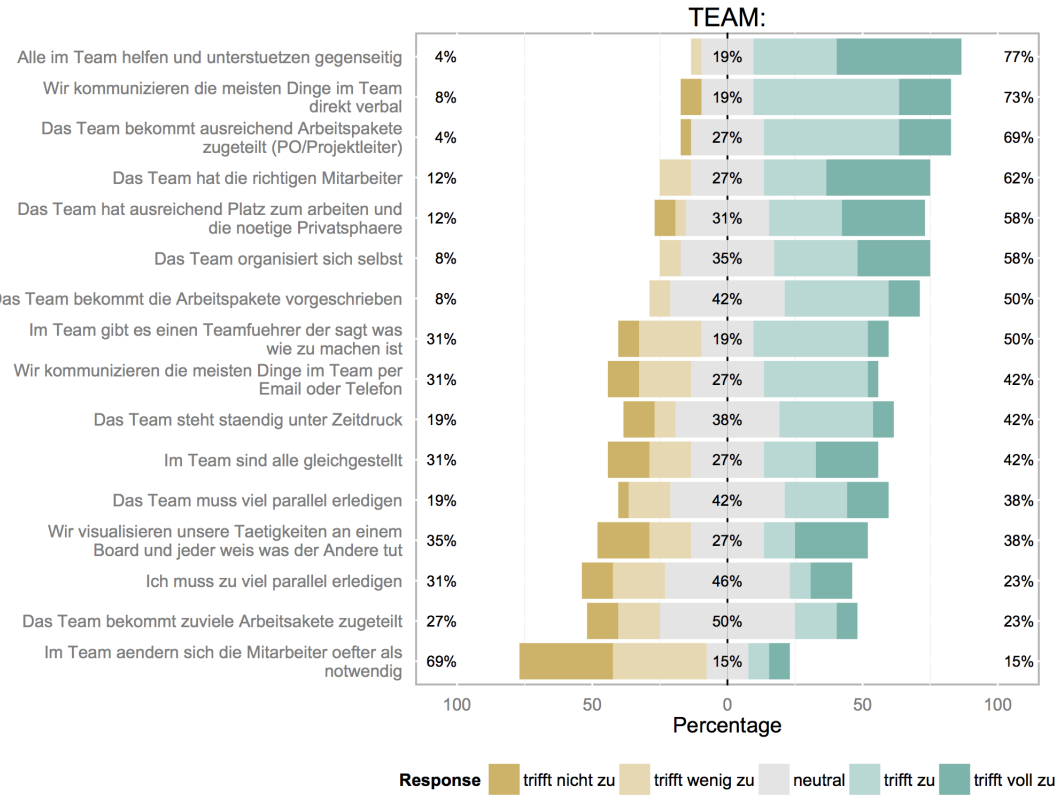


Abb. 18: Erfahrung im Team

Die meisten Befragten erfahren Unterstützung im Team (77%). Ebenfalls die Meisten kommunizieren im Team direkt verbal (73%). Die meisten Teams haben offensichtlich die richtigen MitarbeiterInnen (62%). Ausreichend Platz zum arbeiten und die nötige Privatsphäre scheint mit 58% positiver Meinungen etwas knapp im Überzeugungsgrad zu sein. Hier scheint Handlungsbedarf gegeben zu sein. Interessant ist auch der Umstand, dass 50% angeben, dass es einen Teamführer gibt. Dies ist ebenfalls ein Punkt der mit dem Rollenverhalten derzeitiger Teams zu tun hat und bereits diskutiert wurde. Wie schon besprochen sollten selbstorganisierende Teams auf Augenhöhe agieren können und keine Hierarchien intern bilden. Ein Punkt der absolut betrachtenswert scheint ist der Umstand, dass 50% angeben die Arbeitspakete vorgeschrieben zu bekommen, denn dies passt nicht so ganz zu dem Umstand das 73% der Befragten in agilen Teams beschäftigt sind und in agilen Methoden die Teams selbstorganisierend sein sollen. Diese Aussage passt allerdings zu dem Umstand, dass agile Methoden oft nicht in Reinkultur gelebt werden, und dann Mischmodelle entstehen, welche nicht die Vorzüge der jeweiligen Methode vollumfänglich nutzen können. Zeitdruck geben 42% als bestätigt an und ebenfalls 42% kommunizieren meist per Email oder Telefon. Hier ist Handlungsbedarf gegeben da ein wesentlicher Erfolgsfaktor die direkte Kommunikation ist. Nur 42% fühlen sich gleichberechtigt im Team. Diese Aussage passt zu der Aussage das 50% eine Teamführer haben. Wie

schon gesagt sollte dies bei richtiger Team-Bildung und ausreichendem Coaching nicht passieren. Immerhin 38% der Befragten müssen mehrere Dinge parallel erledigen. Beachtenswert verwenden nur 38% ein Board zur Visualisierung. Dies ist interessant da wie zuvor bemerkt 73% angeben in agilen Projektteams zu arbeiten. Diese Aussage passt also nicht mit agilen Methoden zusammen, da die meisten agilen Methoden Board's zur Visualisierung verwenden. Die Punkte mit zu vielen Arbeitspaketen (23%) und MitarbeiterInnen wechseln öfter als nötig (15%) scheinen im normalen Bereich zu liegen. Grundsätzlich relativiert die Aussage über die Erfahrung im Team einige Dinge die weiter oben positiver gesehen wurden. Dies dürfte dem Umstand geschuldet sein, dass diese Fragen hier offensichtlich kritischer beantwortet werden, da es Fragen zum Team sind und wahrscheinlich persönlicher gesehen werden.

FAZIT AUS DEN AUSWERTUNGEN

Es ergibt sich ein relativ eindeutiges Bild das durchaus in Korrelation zu den Recherchen gesehen werden kann. Die Aussage ist, dass grundsätzlich versucht wird in Projekten agile (moderne) Vorgehensmodelle einzusetzen. Die meisten Befragten kennen einige dieser Methoden und erleben Diese auch in ihren Projekten. Die Daten zu den grundlegenden Elementen die erforderlich sind um tatsächlich agile Methoden erfolgreich einzusetzen scheinen jedoch nicht vorhanden zu sein. Wichtige Elemente wie Unterstützung des Management oder die Daten zur Gleichberechtigung sowie zur Anwendung von Visualisierungs-Hilfen ergeben kein positives Bild. Es spricht eher gegen die korrekte Anwendung erfolgreicher agilen Projekt-Management Methoden. Diesen Umstand betrachtend soll versucht werden einen Ansatz zu entwickeln der die Erfolgsfaktoren in sich bündelt um die Defizite in den momentan gelebten Methoden auszugleichen.

3.3 Das Gerüst

Die agilen Prinzipien beschreiben ein idealisiertes Bild des Projekt-Managements in der Software-Entwicklung. Nachdem es auch in der Software-Entwicklung gute Schule ist bestehende Dinge wieder zu verwenden und um die nötigen Dinge zu ergänzen, kann dieser Ansatz auch dazu verwendet werden ein Set an Methoden und Maßnahmen festzustellen um ein neues Projektmanagement zu installieren. Dazu bietet sich an, die agilen Prinzipien zu betrachten und jede Aussage einzeln mit einer Antwort zu versehen. Es soll das Ziel sein aus bestehenden Methoden oder Verfahren einen best-practices Ansatz für das Projekt-Management zu entwickeln. Anschliessend (innerhalb des Punktes 3.3) werden jeweils die dazu passenden agilen Prinzipien zur Einleitung als *Zitat* aus [20] genannt und dazu eigens ergänzt welche Lösung sich aus bestehenden Methoden oder Hilfsmittel sowie neuen Ansätzen ergibt. Es werden nicht alle Möglichkeiten genannt sondern versucht im Gerüst die Erfolgsträchtigsten Methoden und Hilfsmittel zu verwenden.

3.3.1 Kontinuierliche Lieferung an den Kunden

Zitat aus[20] „Unsere höchste Priorität ist es,
den Kunden durch frühe und kontinuierliche Auslieferung
wertvoller Software zufrieden zu stellen.“

Ein Kanban-System erfüllt sämtliche dieser Anforderungen. Der kontinuierliche Fluss an bearbeiteten Tasks bietet viele Vorteile. Es gibt keine definierten Zeitpunkte wo Software zur Verfügung steht, wie bei SCRUM-Sprints. Durch continuos Deployment kann in der Testumgebung ständig eine lauffähige Applikation zur Verfügung gestellt werden. Neue Features fließen nahtlos in das System ein. Durch die Betrachtung des WIP und dessen Begrenzung kann gleichzeitig die Qualität reguliert werden. Kanban fügt sich gut in bestehende Systeme ein. Kanban-Systeme leben auch das Prinzip der Offenheit. Jeder soll den aktuellen Status des Systems einsehen können.

3.3.2 Anforderungsänderungen begegnen

Zitat aus[20] „Heisse Anforderungsänderungen selbst spät
in der Entwicklung willkommen.
Agile Prozesse nutzen Veränderungen
zum Wettbewerbsvorteil des Kunden.“

Kanban kann um continuos-planning und continuos-integration erweitert werden. Diese Möglichkeit erfüllt perfekt die Anforderung in diesem Punkt. Es gibt eigentlich keine andere Methode die ähnlich flexibel agieren würde. Reines SCRUM wäre zu schwerfällig diesbezüglich. Man kann SCRUM zu Scrumban erweitern, dies würde die Vorteile beinhalten. Durch die Möglichkeit in den Fluss eines Kanban-Systems über das priorisierte Backlog ständig neue Tasks einpflegen zu können entsteht ein sehr schnell reagierendes System in Bezug auf Veränderung.

3.3.3 Iterationen und regelmäßige Lieferungen

Zitat aus[20] „Liefere funktionierende Software
regelmäßig innerhalb weniger Wochen oder Monate und
bevorzuge dabei die kürzere Zeitspanne. „

Alle agilen Methoden versuchen den iterativen Ansatz zu leben. FDD und SCRUM arbeiten Ihre Feature-List oder User-stories ab und stellen Inkremente zur Verfügung. Wieder einmal weiter gedacht kann mit einem Kanban System die wahrscheinlich kürzere Zeitspanne gewählt werden. Durch den kontinuierlichen Fluss an Erweiterungen bräuchte man auch keine fixen Zeitpunkte wie Sprints abzuwarten. Ein ideales System hätte nach der Entwicklung die Testphasen technisch und fachlich nachgereiht und könnte in kurzen Etappen zu den Abnahmetests gehen.

3.3.4 Schnittstellen zum Kunden

Zitat aus[20] „Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten. „

Es ist erforderlich das Teams ungehinderten Zugang zum fachlichen Bereich bekommen. Das fachliche Verständnis bringt meist der Kunde mit. Die Vertretung des Kunden stellt in SCRUM der Product Owner dar. Diese Rolle lässt sich auch in ein Kanban System integrieren. Es soll also diese Rolle oder auch mehrere Personen mit dieser Rolle geben. Je eher der Kunde sein beauftragtes Produkt sieht und auch beurteilen kann, desto eher lösen sich auch Unklarheiten auf. Notwendige Änderungen können einfliessen und so ständig die Qualität des Produktes erhöhen. Die Kommunikationswege Kunde zu Entwickler sind von entscheidender Bedeutung.

3.3.5 Selbstorganisierte Teams

Zitat aus[20] „Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen. „

Selbstorganisierende hoch motivierte und highly sophisticated Teams sind einer der Erfolgsfaktoren. XP bringt als Grundsatz mit das MitarbeiterInnen dynamisch arbeiten sollen. Dies bedingt eine Unternehmenskultur die dem entspricht. Kaizen ist solch eine Unternehmenskultur. Sie bedingt eine Geschäftsführung die MitarbeiterInnen diese Möglichkeit gibt. Die Geschäftsführung stellt eine Management-Ebene zur Verfügung und baut eine Unternehmenskultur auf die selbstorganisierende Teams ermöglicht. Das Stichwort hierfür ist Vertrauen. Die MitarbeiterInnen fühlen sich sicher durch dieses erzeugte Umfeld und arbeiten daran es weiter zu verbessern.

3.3.6 Kommunikation

Zitat aus[20] „Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht. „

Teams sind am erfolgreichsten wenn die Mitglieder auf Augenhöhe miteinander Kommunizieren. Die Grundlage dafür wird durch die Unternehmenskultur gelegt. Durch das Recruiting werden die richtigen Personen in einem Team vereint. Es gibt einen Coach der das Team betreut, die Unternehmenswerte vertritt und darauf achtet das die Kommunikationsfähigkeit erhalten bleibt. Die Team-Mitglieder teilen sich ein Büro oder zumindest Räumlichkeiten die nahe sind. Es gibt ein Board zur Visualisierung. Es gibt Standup-Meetings am Board um sich auszutauschen. Direkte Kommunikation wird innerhalb des Teams gelebt.

3.3.7 Funktion und Qualität

Zitat aus[20] „Funktionierende Software ist das wichtigste Fortschrittsmaß. „

Kanban kennt ein Mittel die Qualität zu verbessern. Begrenzung des WIP. Qualität der Software während der Entwicklung beugt Zusatzkosten in der Zukunft vor. Dinge einfach zu halten erfordert das Verständnis der Teams. XP findet den Ansatz Dinge möglichst einfach zu beginnen und dann genau auf das Maß zu vergrößern um die Funktionalität zu erreichen. Dies ist wesentlich sicherer und einfacher als Dinge groß anzugehen und dann etwas wegzuschneiden um die Funktionalität passend zu machen.

3.3.8 Work Life Balance

Zitat aus[20] „Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können. „

XP fordert in seine Grundsätzen das Personen dynamisch mit Energie arbeiten sollen. Vor allem im Bereich Pair-Programming. Kanban fördert die Bedingungen des kontinuierlichen Flusses. Kontinuität kann nur erreicht werden, wenn Personen über einen langen Zeitraum kontinuierlich Leistung bringen kann. Wenn diese Voraussetzung (Belastungsgrenzen) durch begrenzen des WIP erreicht werden kann dient dies der nachhaltigen Unternehmensentwicklung. Ein negatives Beispiel wären Scrum-Teams in einem Scrum-Projekt, wo die Schätzungen falsch waren und die Teams ihre Sprints nicht schaffen. Die Velocity fällt, die Teams sind demotiviert und es gibt kaum die Möglichkeit verlorene Sprints wieder aufzuholen. Es kann zwar durch die PO's versucht werden im Folge-Sprint Story-Points freizuhalten, bringt aber wirklich nicht viel. Das Problem scheint im SCRUM – Prozess als Ganzes begraben zu sein. SCRUM möchte durch Planung und Iterationen das Risiko minimieren. Gleichzeitig hat es eine komplexe Rollen-Struktur um den SCRUM-Prozess zu überwachen. SCRUM hat aber keine Möglichkeit die Ursachen innerhalb des Teams auf fein granulärer Ebene zu sehen, sondern nur festzustellen das die Velocities der Teams sinken. Der springende Punkt ist es aber zu sehen wo es hakt. Man erkauft sich die vermeintlicher Sicherheit der Planung in SCRUM sehr teuer. Den Vorteil zu sehen wo die Probleme eigentlich sind hat ein Kanban-System. Dort ist durch die Visualisierung auf einem Kanban-Board täglich ersichtlich wo es hakt. Es können dann sofort Maßnahmen ergriffen werden wie zum Beispiel den WIP begrenzen und notfalls Ressourcen umzuschichten. Dieser balancierende Effekt wirkt sich positiv im Team aus und fördert nachhaltig die Energie der MitarbeiterInnen.

3.3.9 Fachliche Qualifikation und Weiterbildung

Zitat aus[20] „Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.“

XP verwendet gute Ansätze um technische Exzellenz zu fördern und zu vergrößern. Pair-Programming ist ein guter Ansatz. Vor allem in Verbindung mit Test-Driven-Development. Wenn man im Team auf die Typen der handelnden Personen achtet (Stichwort: Code-Held oder Perfektionist) erhält man selbst befruchtende Ansätze.
Wenn eine Unternehmenskultur Gemeinsamkeit fördert und dies einen Wert darstellt, dann können hierarchisches Denken in den Hintergrund treten und Team-Mitglieder

erkennen das gemeinsam mehr zu leisten ist. Die Team-Mitglieder können ihre stärken gegenseitig austauschen und so ihr Niveau angleichen. Dies fördert technische Exzellenz. Gleichzeitig kann das Unternehmen begleitend Schulungsmaßnahmen anbieten.

3.3.10 „Keep it simple“, oder die richtige Wahl des Teams

Zitat aus[20] „Einfachheit -- die Kunst, die Menge nicht getaner Arbeit zu maximieren -- ist essenziell.“

Die richtige Team-Zusammensetzung ist entscheidend für den Erfolg eines Teams. Es gibt die unterschiedlichsten Typen von Softwareentwicklern (Stichwort: Perfektionist). Das Team als Ganzes muss erkennen was dieses Prinzip der Vereinfachung bedeutet. Einfacher Code bedeutet wesentlich weniger Fehler und Nachbearbeitung. Wartung wird einfacher und die Entwicklung schneller. Hier sollte der Coach eingreifen um das Team immer wieder zu ermutigen einfach zu arbeiten. Einfach heißt auch notfalls auf eine andere Technologie umzusteigen. Mut ist eines der positiven genannten Attribute in agilen Methoden. XP gibt es sogar als Grundsatz heraus , lieber Code einfach zu beginnen um dann nötigenfalls zu erweitern. Kanban-Systeme sind von sich aus dazu geeignet mit den richtigen Maßnahmen den „waste“ (auf Halde produzieren) zu minimieren. Wichtiger das Richtige zu tun, als viel zu tun.

3.3.11 Gleichberechtigung in Teams

Zitat aus[20] „Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.“

Kanban unterstützt wahre selbstorganisierende Teams. Es bedingt natürlich eine Unternehmenskultur die dies fördert. Wenn ein Team in der richtigen Kultur und als Team lebt, kommen alle handelnden Personen zum Wort und Ideen oder Möglichkeiten werden gehört. Ein Umstand den andere Projektmanagement-Kulturen so nicht erleben. Oft haben Personen die besten Ideen oder auch Erfahrungen, wo man es nie vermuten würde. Wenn Teams begleitet vom Coach ihre Ideen untereinander austauschen können Verbesserungen passieren. Warum nicht die gesamte Kraft des Teams nützen als die von einzelnen schillernden Persönlichkeiten im Team.

3.3.12 Selbstreflexion

Zitat aus[20] „In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.“

Es gibt in SCRUM die Retrospektiven , die es dem Team ermöglichen über den letzten Sprint zu reflektieren. Die Frage ist, welche Aussage getroffen werden kann. Eigentlich kann nur die Velocity betrachtet werden und diese trifft keine wesentlichen Aussagen außer ob die nötige Punktezahl erreicht wurde oder nicht. Ein laufendes Kanban-System liefert wesentlich mehr Daten und dies ständig. Vor allem durch die Visualisierung haben die Teams jederzeit die Möglichkeit zu reflektieren. Eine erweiterte Form der Reflexion kann außerhalb des technischen oder fachlichen Bereiches geschehen. Der Coach hat die Möglichkeit die Teams darin zu unterstützen mehr als Team zu agieren. Team-Entwicklung ist ein laufender Prozess der sich ständig verändert. Es benötigt eine objektive Sicht von Außen, die der Coach anbieten kann.

FAZIT AUS DEM GERÜSTBAU

Es ergibt sich aus der Beantwortung der agilen Prinzipien eine positive Haltung für ein Kanban System (oder einen Lean-Ansatz). Die Vorteile von Kanban-Systemen überwiegen augenscheinlich. Kanban-Systeme dienen der nachhaltigen Unternehmensentwicklung und schaffen nachhaltig soziales Kapital. Ausserdem bieten Kanban-Systeme alle Eigenschaften zur Erfüllung der Kriterien der agilen Prinzipien. Kanban-Systeme sind für sich Alleine keine Projektmanagement-Methode sondern eher ein Change-Management Ansatz. Um solch ein System als Projektmanagement-Methode festzustellen, müssen Ergänzungen vorgenommen werden im Bereich der Rollenverteilung und der nötigen Hilfsmittel. Kanban führt aber keine expliziten Rollen an, da es geschaffen wurde um bestehende Methoden zu ergänzen und um wichtige Faktoren zu bereichern. Die Rollen müssen also neu definiert werden, und in einem neuen Ansatz zu bestehenden Verfahren abgegrenzt sein. Ebenso ist es erforderlich eine Anleitung zur Visualisierung der Prozesse anzugeben. Außerdem soll eine Beschreibung angeben werden wie ein Einsatz in eine bestehende Umgebung stattfinden kann.

3.4 Die Rollen

EINLEITUNG

Die folgenden Rollen, stellen eine vollständige Besetzung der Teams und Außenstehenden dar. Sie ist eine Empfehlung, die skalierbar gehalten ist. Wichtig ist die Rolle des Coach der, nicht mit einem Projektleiter zu verwechseln, eine Notwendigkeit vor Allem bei der Einführung einer neuen Projektmanagement-Methode darstellt. Weiters dient die Aufstellung der nötigen Rollen dazu die Grundlagen für selbstorganisierende Teams zu erstellen. Wichtig ist das Verständnis, das diese Rollen die Erfolgsträchtigkeit des Systems erhöhen. Sie sollen die Eckpfeiler darstellen, auf die eine geänderte Sichtweise aufbauen kann. Es wird im Grunde davon ausgegangen die Vorteile aus Kanban-Systemen zu übernehmen und zu ergänzen. Eine Anlehnung an bestehende Rollenbeschreibungen wie zum Beispiel aus SCRUM ist nicht beabsichtigt wenn auch die Rollennamen vielleicht ähnlich lauten. Wann immer möglich wurde eine neue Bezeichnung gewählt. Die Bezeichnungen sollen aber letzten Endes auch namentlich Sinn ergeben.

3.4.1 Request Manager

Der Request-Manager (RM) sammelt Anforderungen, von verschiedenen Personen oder Abteilungen oder Kunden. Er priorisiert und pflegt Sie in das Backlog ein. Er steht in Kontakt mit den Stakeholder's und unterstützt Ihre Interessen. Der Priorisierung kommt besondere Bedeutung zu, da Sie darüber entscheidet in welcher Reihenfolge die Anforderungen in das System fließen. Dies beeinflusst in hohem Maße die Ergebnisse. Der RM teilt gemeinsam mit dem Kunden die Arbeitspakete in einzelne Teilpakete auf und achtet darauf das die Teilpakete einen weiten Bereich verschiedener Größen abdecken sondern nur wenige Größenordnungen wie zum Beispiel „Klein, Mittel, Groß“. Er hält auch die Informationen und Termine für die Beteiligten bereit. Er stellt auch die Produkt-Vision dar. Dies ist die Sicht auf das Endergebnis im Sinne des Kunden und in Absprache zwischen ihm und dem Kunden.

Die Produkt-Vision vermittelt er den Teams in einfacher und klarer Art und Weise. Die Teams gehen auf ihn Proaktiv zu wenn Sie Informationen brauchen. Er ist nicht zu verwechseln mit dem PO aus SCRUM, welcher dem Team die Arbeitspakete aus dem Backlog vorgibt. Er priorisiert Sie nur. Das Team entscheidet in welcher Reihenfolge die Abarbeitung stattfindet. Damit ist gewährleistet, das ein Team selbstorganisiert bleiben kann. Der RM schirmt die Anfragen der Kunden vom Team ab und kommuniziert sie eindeutig weiter in Richtung Team. Er hält mir dem Team Synchronisierung-Meetings ab um die Priorisierung der Tasks zu erläutern. Er definiert mit dem Kunden die Termine für die Lieferungen von Testversionen. Er synchronisiert dieser Termine mit dem kontinuierlichen Fluss der Teillieferungen durch das Team. Er nimmt nicht an den täglichen Kurzbesprechungen teil. Dies erhält dem Team die Möglichkeit der Selbstorganisation. Er gibt das Feedback aus den Akzeptanz-Tests der Kunden direkt an das Team weiter. Dies dient dem Team zur Selbstreflexion.

3.4.2 Team Coach

Der Team Coach (TC) ist idealerweise eine Person von Außen. Eine Person von innerhalb des Unternehmen, kann möglicherweise nicht objektiv genug handeln. Vor allem in Hinblick auf positive Änderung der Unternehmenskultur. Der Coach soll objektiv auf die Teams blicken können, und idealerweise nur der Geschäftsführung unterstellt sein. Dies befreit den Team Coach von Einzelinteressen mittlerer Entscheidungsträger. Er behält so die Möglichkeit die Teams frei unterstützen zu können aber auch die Möglichkeit der Geschäftsführung direkt zu berichten. Der Coach ist eine Person welche die nötigen fachlichen, aber vor allem die psychologischen, emotionalen und sozialen Skills mitbringt. Idealerweise hat er weit greifende Erfahrung in Projekten sammeln können da er früher selbst Mitglied von Software-Entwicklungs-Teams war. Der Team Coach hilft dem Team das Bewusstsein zu entwickeln, dass als Gruppe zu handeln mehr Vorteile bringt als einzeln zu handeln. Er hilft dem Team in der Organisation des Boards und achtet darauf, das das Board richtig gehandhabt wird. Er erklärt dem Team die Vision eines geänderten Projektmanagements und der den Vorteilen der Selbstorganisation. Er hilft dem Team den Durchsatz zu messen und den WIP auf die richtige Größe zu beschränken. Der TC unterstützt ein oder mehrere (Alle) Teams im Projekt. Er hält die Prinzipien aufrecht die zur Einführung eines neuen Ansatzes gehören. Er begleitet langfristig. Er berät die Geschäftsführung in schwierigen Fragen, vor allem während der Eingangsphase. Er berichtet der Geschäftsführung von Veränderungen und versucht das gegenseitige Verständnis zwischen Geschäftsführung und Teams aufrecht zu halten. Im laufe der Zeit hilft er dem Team sich weiterzuentwickeln und die Sicht auf die eigene Arbeitsweise immer feiner zu strukturieren. Dazu bildet er mit dem Team die immer feiner werdende Arbeitsweise auch in den Visualisierungs-Hilfen aus. Die Board's werden komplexer, und die daraus resultierenden Aussagen vielfältiger und aussagekräftiger. Auch die Messinstrumente werden weiter ausgebaut, damit sich das Team selbständig Kenngrößen entwickeln kann um die eigene Leistung besser zu beurteilen. Dadurch hilft er dem Team zur Selbstreflexion und unterstützt damit die wachsende Perfektion.

3.4.3 Global Architect

Der Global Architect (GA) entwirft die technische Infrastruktur auf Basis von Kundenwünschen und den Umsetzungsplänen der Teams. Er hält Synchronisierung-

Meetings mit den Teams ab. In diesen Meetings soll eine gemeinsame Vorgangsweise diskutiert werden. Die Teams sollen in der Wahl Ihrer Werkzeuge und Mittel frei bleiben. Der GA bildet sozusagen die gemeinsame Linie ab an der die Teams entlang gehen. Der GA prüft auch ständig die technische Infrastruktur und versucht sie zu vereinfachen. Dies passiert jedoch nur auf der grundlegenden Struktur. Im Teilbereich vereinfachen die Teams selbstständig und permanent. Er hält sich an best-practices Ansätze und sein Ziel ist es funktionierende hochqualitative Produkte zu entwerfen mit dem Gedanken an Wartbarkeit und Einfachheit. Er wird auch vom Unternehmen in Form von Schulungsmaßnahmen unterstützt damit er immer am neuesten Stand sein kann. Vor Allem ist ihm wichtig einen breiten Blick zu bekommen in den Möglichkeiten Software zu entwickeln. Er verschliesst sich neuen Technologien nicht und testet neue Dinge aus. Er unterrichtet das Management dahingehend, wenn es zielführender wäre neue oder andere Technologien einzusetzen. Er scheut sich nicht Strukturen zu verwerfen wenn dies zielführend ist. Schliesslich soll Mut in der agilen Welt ja belohnt werden. Dazu braucht er allerdings den Rückhalt der Teams. Idealerweise ist er also eine Persönlichkeit die ebenfalls mehr Reife mitbringt als die reine fachliche Qualifikation.

3.4.4 Toolsmith

Die Aufgaben des Toolsmith (TS) sind dazu da, die Teams zu unterstützen. Es gibt immer wieder Standardaufgaben wir grundsätzliche Frameworks oder Strukturen aufzubauen oder anzulegen. Es gibt genügend Tätigkeiten im Bereich der Entwicklungs-Tools die ebenfalls zeitraubend sein können und nicht notwendigerweise von jedem einzelnen Team-Mitglied nochmals durchgeführt werden muss. Diese Tools konfiguriert der TS und stellt Sie den Teams einheitlich zur Verfügung. Damit gewährleistet er auch, das Dinge nicht doppelt gemacht werden oder es verschiedene Konfigurationen gibt. Er gewährleistet auch das wiederholbare Dinge einheitlich sind vor allem bei bestehenden Frameworks oder Basiselementen in der eigenen Software. Dabei achtet er aber darauf nicht zu generisch zu entwickeln, was einer Code-Vereinfachung entgegenstehen würde. Dies kommt der Code-Qualität und dem lesen des Code zu gute. Er kümmert sich auch um die Entwicklungsumgebungen und Datenbankzugänge sowie alle Art von benötigten Tools.
Die Teams haben somit homogene Arbeitsumgebungen und können sich auf die fachlichen Prozesse und die Anwendungsentwicklung konzentrieren. Die Zentralisierung von generalisierten Elementen in den Projekten hilft die Fehlerhäufigkeit zu vermindern und die Einheitlichkeit von Code zu verbessern. Ausserdem ist die Austauschbarkeit der Arbeitsplätze gewährleistet. Im Sinne des Pair-Programming oder Test-Driven-Development, sowie der Umverteilung von Ressourcen bei Bedarf, ein gewichtiges Argument. Er ist möglicherweise ein Mitglied des Teams oder eine Abteilung die für mehrere Teams zuständig ist.

3.4.5 Team

Das oberste Ziel des Teams ist es qualitativ hochwertige Software zu erstellen im Sinne des Kunden. Dazu wendet es alle Methoden an die dazu hilfreich sein können. Das Team setzt sich idealerweise aus interdisziplinären Persönlichkeiten zusammen (je nach Anforderung) mit den nötigen sozialen und emotionalen Skills. Es kann so bei Bedarf auf eine breite Wissensbasis zurückgreifen und die Fähigkeit entwickeln einzelne Team-Mitglieder bei Bedarf zu unterstützen und somit sich selbst zu stärken.

Das Team hält die Prinzipien Gleichwertigkeit und direkter Kommunikation aufrecht. Es entnimmt Tasks aus dem Backlog und pflegt Sie in das System zur Abarbeitung ein. Es entwickelt gemeinsam mit dem Coach die WIP-Beschränkungen und versucht die Formalismen der Visualisierung einzuhalten. Das Team misst den Durchsatz im System und reflektiert laufend über Veränderungen durch den Einsatz sinnvoller Metriken. Die Teammitglieder unterstützen sich gegenseitig und erkennen das die Gruppe leistungsfähiger ist als einzelne Individuen. Das Team entscheidet selbst, wie es die Dinge umsetzt. Es synchronisiert sich jedoch mit dem GA um die grundlegende Vorgangsweise zu klären, oder um Verbesserungsvorschläge einzubringen. Es nimmt die Hilfsmittel des TS zu Hilfe um keine unnötigen Aufwände zu generieren. Das Team übernimmt Verantwortung für die Tasks die es bearbeitet. Es wendet best-practices Ansätze an (Pair-Programming, TDD) um effektiver zu sein und die Qualität zu verbessern.

3.4.6 Kunde

Der Kunde steht im direkten Kontakt mit dem RM. Der Kunde bespricht mit dem oder den RM(s) seine Anforderungen und setzt sie gemeinsam mit dem RM(s) in Stories um. Aus diesen Stories werden einzelne Arbeitseinheiten abgeleitet und priorisiert.
Der Kunde macht dem RM die Priorisierung klar, das letzte Wort in der Einteilung und Priorisierung hat jedoch der RM. Darauf basierend muss es ein Vertrauensverhältnis zwischen Kunde und RM geben, damit die Gründe für anders lautende Priorisierung als gewünscht auch verstanden wird. Diese Vorgangsweise entkoppelt die Entwicklungsteams von der irrtümlichen Willkür mangels besseren Wissens von Seiten des Kunden. Letztendlich kann nur der RM , der GA, sowie die Teams wissen welche die effizienteste Reihenfolge der Abarbeitung von Arbeitseinheiten ist. Ausserdem soll der Einfluss des Kunden direkt auf die Entwicklungsteams entkoppelt sein. Dies erhält die Selbstorganisation der Teams. Idealerweise bringt der Kunde für die Ansätze des gewählten PM-Ansatzes Verständnis auf und versteht das ein komplexes Projekt nur erfolgreich ist wenn die Kräfte sinnvoll genutzt werden. Der Kunde kann vom Coach in der Meinungsbildung über PM-Methoden unterstützt werden wenn es dazu Bedarf gibt.

FAZIT AUS DER AUFTEILUNG DER ROLLEN

Die Aufteilung der Rollen wurde so kompakt wie möglich gehalten. Idealerweise sind die Rollen wie beschrieben besetzt. Die Beschreibung sollte auch in gewisser Weise die Philosophie hinter der Methodik klar machen.

Die grundlegende Aussage ist:
Bilde ein oder mehrere selbstorganisierende Teams aus. Dies ist die Rolle des TC. Bilde die grundlegende Struktur der Software aus, wenn auch nur als Beschreibung oder Anleitung. Dies ist die Aufgabe des GA. Stelle dem Team die geeigneten Hilfsmittel zur Verfügung und verhindere das Teams damit zu tun haben Basisaufgaben zu lösen und dies mehrfach. Dies ist die Aufgabe des TS. Achte darauf das die Arbeitseinheiten die in das Backlog einfliessen richtig dimensioniert und priorisiert sind. Dies ist die Aufgabe des RM. Halte die Selbstbestimmtheit von Teams aufrecht. Dies ist die Aufgabe von RM und dem Kunden. Unterstütze den Kunden bei der Entscheidungsfindung und Priorisierung. Dies ist die Aufgabe des RM. Unterstütze

den Kunden die eigene Wissensbasis bezüglich der gewählten PM Methode zu erweitern. Dies ist die Aufgabe des TC. Zusätzlich zur Rollenstruktur muss es auch geeignete Hilfsmittel geben.

3.5 Die Hilfsmittel

3.5.1 Backlog

Das Backlog ist die Basis des Workflows. Der RM pflegt und priorisiert das Backlog. Es ist idealerweise elektronisch geführt. Ein Tool der Wahl ist sicherlich Atlassian-JIRA [21] für professionelle Einsätze. Es gibt auch andere Open-Source Software zur Backlog Verwaltung wie [22]. Die Teams entnehmen die Tasks aus dem Backlog und reihen sie in den Bearbeitungsfluss ein. Die meisten Backlog-Systeme haben auch umfangreiche Mittel zur Analyse. Wichtig ist, das es aktuell ist. Dafür verantwortlich ist der RM. Wenn das Backlog nicht korrekt und ständig gefüllt ist leidet die Arbeitsleistung der Teams. Wichtig ist, dass das Backlog mit korrekt dimensionierten sowie korrekt priorisierte Aufgaben gefüllt ist. Dies beeinflusst maßgeblich wie der Berbeitungsfluss läuft. In Kanban-Systemen soll niemals ein Engpass entstehen. Es soll aber vor allem der Durchfluss konstant sein. Dazu braucht es einen ständig gefüllten Buffer an Arbeitseinheiten durch das Backlog. Stimm die Basis, kann das System reibungsfrei laufen.

3.5.2 Visualisierung

Der übliche Weg der Visualisierung ist ein Kanban-Board. Es hat grundlegende Elemente. Es kann sehr einfach aufgebaut sein um sich mit der Zeit zu entwickeln. Hier ein Beispiel:

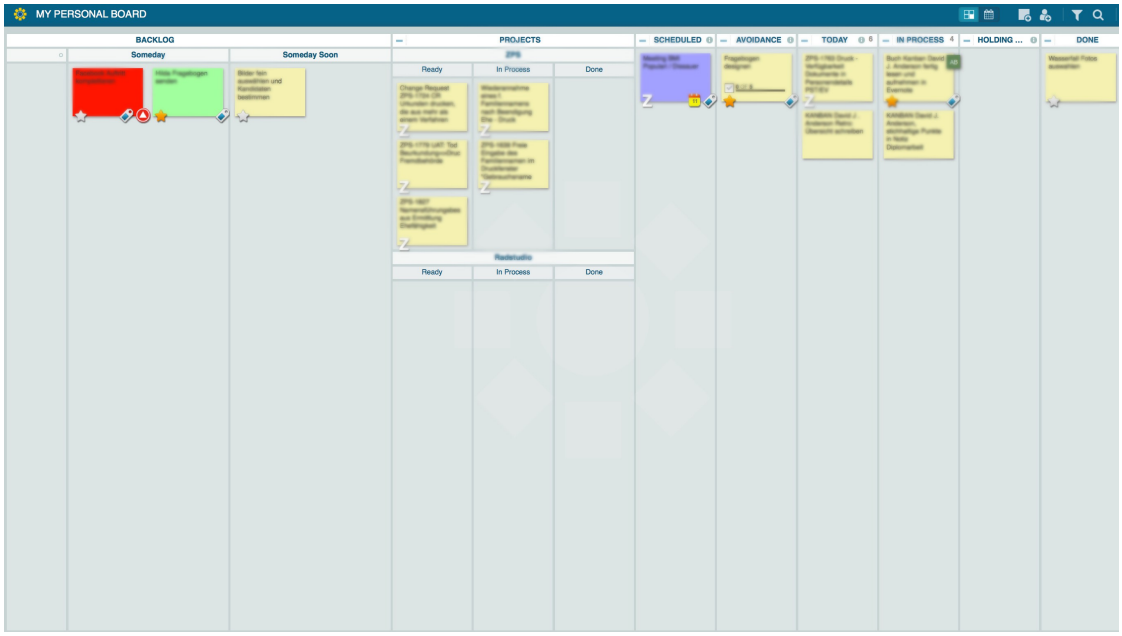


Abb. 19: Kanban-Board Beispiel,eigene Entwicklung.

Wichtig sind folgende Faktoren. Jedes Team entwickelt sein eigenes Board. Es besitzt einen Anfang und ein Ende. Dies ist wichtig um zu verstehen, wie Kanban

funktioniert. Zwischen Anfang und dem Ende wird die Durchlaufzeit betrachtet. Diese ist das Maß, der Durchsatz. Ein Task läuft von links nach rechts durch jede Spalte. Es kann jemand dafür verantwortlich sein. Ein Task kann auch einer Serviceklasse zugeordnet werden. Damit lassen sich bei Auswertungen die einzelnen Bereiche klarer trennen. Am Board steht bei manchen Spalten ein WIP-Indikator eingetragen. Dieser stellt die WIP-Begrenzung dar. Nur so viele Karten wie angegeben dürfen in die Spalte eingereiht werden. Das heißt, läuft das System nicht korrekt dann bildet sich automatisch ein Stau vor der vollen Spalte. Dies ist die Möglichkeit sofort zu erkennen wo die Probleme liegen. Ein weiteres gewichtiges Argument für die Begrenzung des WIP, ist die Regulierung der Qualität über den Durchsatz. Durch Umschichtung im Team oder beiseite räumen von Behinderungen, wird der Fluss wieder bereinigt. Der Status des gesamten Systems ist also hier ersichtlich. Kanban-Board's können hierarchisch angeordnet werden um Board's von Teams zu einem übergeordneten Board zusammenzufassen. Dies vor allem bei sehr großen Projekten sinnvoll. Ein Kanban-Board hat nicht nur Spalten. Es ist auch möglich horizontale Linien einzutragen, sogenannte Swim-Lanes. Diese dienen dazu Bereiche zu trennen wie zum Beispiel Change-Requests, Wartung usw. Man kann auch physisch durch die horizontale Reihengröße die Verhältnisse darstellen. Einzelne Tasks werden normalerweise mit Aufgabentypen versehen (farblich oder schriftlich). Es können Namen darauf stehen. Sie sollten ein Eingangsdatum besitzen um die Durchlaufzeit messen zu können. Es sollte wenn möglich ein Bezug zum Backlog-Eintrag bestehen (JIRA-Ticket) oder ein Bezug zu externen Quellen. Die zeitliche Größenordnung von einzelnen Tasks sollte die Gruppen klein – mittel – groß umfassen. Je nach Projekt mögen dies unterschiedliche Größen sein. Jedes Unternehmen wird einen eigenen Sinn dafür entwickeln wie die Größenordnungen sind. Spalten können zweigeteilt sein (Bsp. Analyse → in Arbeit | fertig). Es können auch sogenannte Queues eingeführt werden. Sie dienen als Puffer. Werden diese Puffer ständig voll, dann produzieren Teams auf Halde („Waste“). Dieser Umstand ist nicht erwünscht. Jedes Ticket wird für jeden Tag markiert an dem es nicht bewegt wird. So werden Blocker rasch erkannt. Manche Tasks können mehrere Aktivitäten gleichzeitig bedingen ohne feste Reihenfolge. Dieser Umstand kann auf zwei Arten dargestellt werden. Entweder man gibt diesem Task auf seiner Karte eine Liste von Aktivitäten mit, die dann nach der Bearbeitung abgehakt werden, oder man bildet diese Tätigkeiten wenn Sie mehrere Tasks umfassen in einer Spalte in Sektionen mit Swim-Lanes ab (Bsp.: Entwicklung → Development / Test).

Das Board wächst mit der Entwicklung des Teams mit. Je reifer ein Team ist, desto komplexer und dadurch feiner ist das Board entwickelt. Die Teams können dadurch auf einen Blick viel weitreichendere Aussagen bezüglich des Prozessflusses und der eigenen Leistung treffen als mit einfachen Board´s.

3.5.3 Cumulative Flow Diagram und mittlere Durchlaufzeit

In Abb.6 ist ein Beispiel für ein Cumulative-Flow Diagramm (CFD) angegeben. Das CFD zeigt an ob ein Kanban-System richtig funktioniert. Es zeigt die Menge der bearbeiteten Aufgaben über die Zeit an. Ein konstant laufendes System würde glatte Linien in Ihrer Höhe stabil zeigen. Es kann auch die Schwankungsbreite über die Zeit festgestellt werden. Veränderungen im Team oder der Aufgabenbereiche sowie Veränderung in anderen Bereichen werden im CFD direkt ersichtlich. Es zeigt deutlich den Status des Systems an. Die ist ein bestehendes Hilfsmittel aus Kanban wo keine Ergänzung nötig ist. In Abb.7 ist die mittlere Durchlaufzeit dargestellt aufgeteilt nach

Serviceklassen. Der Vergleich von Durchlaufzeiten zwischen Teams und über längere Zeiten ergibt starke Aussagen für die Vorhersage.

Weit entwickelte Teams und deren Organisation können aus diesen Daten Größenordnungen wie Zeit oder Aufwand ableiten. Hingegen können Burndown-Charts (in SCRUM gerne verwendet) nur die Aussage treffen ob das Team die Ideallinie verlässt oder nicht und dies noch dazu nur am Ende eines Sprints und nicht kontinuierlich. Die Informationen aus CFD und mittlerer Durchlaufzeit sind wesentlich weitreichender und vor Allem zeitnahe Aussagen. Einer der Vorteile dieses Systems ist es, das längerfristig und an Nachhaltigkeit gedacht wird. Dies hebt diese Art des Projektmanagements so positiv hervor.

Teams welche sich in dieses System einfügen lernen sich über die Messungen selbst zu balancieren. Dieser Effekt ist in keiner der bereits genannten agilen Methoden dermaßen erkennbar. Die eingesetzten Instrumente und Hilfsmittel sind einfach und unkompliziert mit gleichzeitiger kontinuierlicher Aussagekraft. Die Struktur wurde bewusst einfach gehalten und Instrumente gewählt, die das Team einfach und effektiv einsetzen kann. Damit der Einsatz dieser Methode erfolgreich gelingt braucht es aber auch eine zeitliche Abfolge wann welche Mittel und in welcher Reihenfolge eingesetzt werden.

3.5.4 Eine Beschreibung für den Einsatz

Um den Einsatz eines neuen Projektmanagements erfolgreich durchführen zu können sind ein paar weiterführende Gedanken nötig die mit den Menschen persönlich zu tun haben. Das Verhalten von Menschen zu ändern ist von außen nicht durchgehend möglich. Dies ist eine Erfahrung die durch namhafte Autoren aus dem Software-Projektmanagement mehrfach bestätigt wurde. Sowohl [1] als auch [10] berichten davon das sehr feinfühlig vorgegangen werden muss wenn neue Systeme eingeführt werden. Die handelnden Personen im Team, das Management und Andere beginnen teilweise alte Verhaltensweisen zu rechtfertigen und zu schützen. Die erzeugt wesentliche Widerstände die dem Erfolg des Einsatzes entgegen stehen. Die Veränderung geschieht wenn Teams und deren Handelnde einen Nutzen für sich persönlich erkennen. Wenn es nötig ist das Projektmanagement im Unternehmen oder in einem Projekt umzustellen, dann kann dieser Prozess nur funktionieren wenn Akteure ein baldiges Erfolgserlebnis daraus beziehen. Es empfiehlt sich ein mehrstufiges Einführungssystem wobei bei bestehenden erfolgreichen Beispielen eine Anleihe genommen wird. Die Kanban-Systeme haben den Vorteil das sie leicht in bestehende System integriert werden können und der Umstieg dadurch leichter fällt. Außerdem ist es eine sehr einfache Methode und die Visualisierungs-Hilfen unterstützen die Sichtweise auf Erfolg. Außerdem lassen sich die Details der Methode schrittweise einführen. Dies ist einer der wesentlichen Vorteile von Kanban-Systemen.
David J. Anderson nennt sechs einzelne Schritte [10, S.28] (beeinflusst von Donald Reinertsen und Eliyahu Goldratt mit seinen Werken zur Engpasstheorie) in folgender Reihenfolge:

- Fokussiere auf Qualität
- Reduziere den WIP
- Liefere häufig
- Sorge für Gleichgewicht zwischen Nachfrage und Durchsatz
- Priorisiere

- Nimm die Quellen von Variabilität ins Visier, um die Vorhersagbarkeit zu verbessern

Die Reihenfolge ist kein Zufall. Sie ist wohl durchdacht. Auf Qualität zu fokussieren ist ein Bereich technischer Natur, welcher durch den Einsatz von Standard-Methoden (Bsp. XP) auf technischer Seite verbessert werden. Methoden wie Pair-Programming und Test-Driven-Development helfen dabei. Das Management kann veranlassen durch bessere Testabdeckung mehr Qualität zu schaffen. Während dieser Phase ist das Board von entscheidender Bedeutung weil es hilft die Arbeiten besser zu strukturieren. Nächster Schritt ist den WIP zu begrenzen. Die ist eine wichtige Maßnahme, die Teams dazu bringt, sich auf das wesentliche zu konzentrieren. Wie in 2.2.2 festgestellt verhält sich die mittlere Durchlaufzeit linear mit dem Maß an Qualität. Den WIP begrenzen ist also das Mittel der Wahl um Qualität zu erzeugen. Daraus resultiert das Teams ihr Selbstbewusstsein stärken können. Die Motivation steigt. Es sollte ein Beruhigungseffekt eintreten. Die Teams können konzentrierter arbeiten und gewöhnen sich allmählich an das kontinuierliche Fluss-System von Kanban. In einem weiteren Schritt kann nach Stabilisierung der Qualität der Durchfluss gesteigert werden bis zu einem Maß welches die angestrebte Qualität halten kann. Der nächste Schritt „Liefere häufig“ sollte sich aus dem Kontext (höhere Qualität und WIP Begrenzung) automatisch ergeben. Es sollte öfter gut funktionierende Software zur Verfügung stehen als vorher. Dies sollte auch die Stimmung im Team merklich erhöhen. Äußerem stärkt es die Vertrauensbildung zum Kunden. Das Feedback sollte positiver werden. Der nächste Schritt in Richtung vollständige Motivation. Bis zu diesen Schritten sind die nötigen Rollen auf TEAM und COACH und GLOBAL ARCHITECT begrenzt. Im nächsten Schritt kommt die Rolle REQUEST OWNER verstärkt zum Einsatz. Es ist in dieser Phase von entscheidender Bedeutung, dass der RO das Backlog so befüllt, dass die Größenordnungen ungefähr den Größenordnungen des momentan möglichen Durchsatzes entsprechen. Der RO muss also die Arbeitspakete mit den Kunden so abstimmen das es für das Team leicht zu schaffen ist auf einen konstanten Durchfluss zu kommen. Dazu ist es nötig die Arbeitspakete in Größenklassen einzuteilen die überschaubar sind. Dies erleichtert die Beurteilung der Arbeitspakete durch die Teams wenn Arbeitspakete aus dem Backlog in den Produktionsfluss eingereiht werden. Wenn dies geschafft ist, kann der RO beginnen richtig zu priorisieren. Damit kann das Management die Reihenfolge der zu liefernden Artefakte besser auf den Kundenwunsch abstimmen. [10] bemerkt zu diesem Bereich (Zitat Seite 38) „*Es ergibt wenig Sinn sich der Priorisierung zuzuwenden, wenn es keine Vorhersagbarkeit bezüglich der Auslieferung gibt.“*. Das heißt es geht darum zuerst einen konstanten Fluss zu erzeugen und dann die Reihenfolge der Lieferungen zu kontrollieren. Das Team hat also mehr Zeit sich an dieser Arbeitsumgebung zu gewöhnen, und der Übergang geschieht sanfter und logischer. Idealerweise hat der RO versucht auf diesen Umstand Bezug zu nehmen und die Arbeitspakete dahingehend abgestimmt.

Die Betrachtung der Variabilität stellt ein sehr fortgeschrittenes Thema dar, welches nicht vollumfänglich in dieser Arbeit diskutiert werden kann.
Zitat [10, S.39] *„Grundsätzlich führt Variabilität zu mehr WIP und längeren Durchlaufzeiten“*.

Es gibt verschiedene Verursacher von Variabilität. Interne und Externe. Zu den Internen zählen:

- Größe von Aufgaben
  Diesen Faktor kann das Management und der RM beeinflussen. Es ist günstig immer ähnliche Größen in Tasks abzubilden und weniger Größenklassen anzubieten.
- Aufgabentypen
  Es ist gut, Arbeiten in Aufgabentypen einzuteilen. Die Betrachtung dieser Gruppen und deren klare Einteilung vermindert Variabilität.
- Serviceklassen
  [10] Beschreibt den Zusammenhang von Auswahl Serviceklassen und beschleunigten Anforderungen. Die Auswahl von Serviceklassen kann gezielt erfolgen. Gibt es aber ein breites Spektrum an beschleunigten Anforderungen (Externe Quelle der Variabilität), kann es passieren das die Auswahl von Serviceklassen eher zufällig erfolgt. Dies erhöht die Variabilität. Es besteht also ein Zusammenhang zwischen beschleunigten Anforderungen und der sinnvollen Auswahl von zu bearbeitenden Serviceklassen. Gefordert ist wieder einmal der RM um diese Faktoren zu optimieren. Er kann versuchen durch geeignete Priorisierung den Fluss wieder zu beruhigen. Dem RM kommt in dieser Hinsicht besondere Bedeutung und auch Verantwortung zu. Idealerweise ist sein Vorgehen mit dem Kunden abgestimmt und auch dementsprechend kommuniziert damit der Kunde die Änderung der Vorgehensweise versteht.

Externe Quellen der Variabilität sind:

- Unklare Anforderungen
  Stellen in vielen Projekten das größte Problem dar. Hier ist das Management, der Kunde und RM gefordert klare Anforderungen und Beschreibungen zu liefern. Es muss allen klar sein das diese Form der Variabilität zu den stärksten Formen gehört. Unklarheiten in der Anforderung wirken auch lange nach. Sie müssen irgendwann korrigiert werden wenn schon Arbeit in dieser Richtung geschehen ist. Dies bedeutet auch das die Teams im Nachhinein gesehen viel „waste“ (auf Halde) produziert haben.
- Beschleunigte Anforderungen
  Können passieren wenn sich Anforderungen unerwartet in ihrer Priorisierung geändert haben, oder die Anforderung in einem frühen Stadium nicht korrekt kommuniziert wurden. Solche massiven Änderungen sollten sich auch immer im CFD zeigen und auch im mittleren Durchfluss. Springen diese Indikatoren an muss das Management reagieren. Der RM und auch das Team haben die Verantwortung die Hintergründe zu hinterfragen.
- Infrastruktur
  Die Lauffähigkeit von Servern, Entwicklungssystemen, Datenbanken und sonstigen Tools ist eine wichtiger Faktor für die Variabilität. Hier sind die Rollen des TS und des GA gefragt, die verantwortlich für die Versorgung der Teams mit Ihren technischen Grundbedürfnissen ist. Sollte TS eine Abteilung sein (um mehrere Teams zu beliefern) dann sollte das Management darüber nachdenken diesen Bereich massiv zu verstärken.

FAZIT AUS BESCHREIBUNG FÜR DEN EINSATZ UND HILFSMITTEL

Es ist wichtig neue Methoden und Instrumente schrittweise und in der richtigen Reihenfolge einzuführen. Grundlage nach der Rollenverteilung ist die Visualisierung, die es überhaupt möglich macht die Funktion des Systems beurteilen zu können. Die Teams selbst können sehen wie alles läuft. Dies schafft Vertrauen in Veränderungen. Vor allem wenn positive Beispiele passieren. Wenn zu Beginn der WIP begrenzt wird und Qualität gesteigert wird ist dies am Kanban-Board ersichtlich. Der Fluss wird konstant. Die Teams motivierter und selbstbewusster. Sie erkennen die Vorteile der Selbstorganisation. Anschliessend wird die Arbeitsweise verfeinert. Metriken werden herangezogen um weitere Beurteilungen abgeben zu können. Vor allem das Feedback der Teams ist entscheidend. Es werden häufiger Lieferungen an den Kunden durchgeführt. Dies schafft weiter Vertrauen aus der Sicht des Kunden. Sind die Basis-Elemente richtig eingebettet und funktionieren, kann man weitere steigernde Maßnahmen einführen. Hier kommt dem RM und dem Kunden die besondere Rolle zu. Zuerst wird für einen konstanten Fluss gesorgt, und anschliessend wird die Priorität der Aufgaben verfeinert. Sind alle diese Maßnahmen erfolgreich eingeführt, wird es Zeit die Variabilität zu eliminieren. Dies ist ein längerer Prozess der an verschiedenen Ebenen durchgeführt werden muss, da es viele unterschiedliche Ursachen für Variabilität gibt. Gleichzeitig wird aber das gesamte System geglättet und inklusive aller Schnittstellen nach Außen in seiner Funktion wesentlich verstärkt.

4 Analyse

Es gibt in Software-Projekten Probleme bei der Umsetzung. Die untersuchten Projektmanagement-Methoden werden bereits zum Teil eingesetzt. Auf die Bedürfnisse und Fähigkeiten (Möglichkeiten) der Handelnden wird zuwenig eingegangen. Die Teams sind oft zu groß. Das Ergebnis waren schlechte Projekterfolge. Die Standish-Group beziffert in [16, S.3]: Erfolgreiche Projekte hatten 61% emotional reife Persönlichkeiten an Board, wogegen Projekte, die nicht oder nur durch Verzögerungen fertiggestellt wurden, lediglich 19% emotional reife Persönlichkeiten an Board hatten. Dies entspricht auch den Aussagen von [12, S.10] und [5, S.2], welche schlechte Kommunikation und Teamfähigkeit als die Hauptfehlerquellen in Projekten einstufen. Die Auswertung der Befragung hat ein stimmiges Bild ergeben. Nach diesem Bild ist die Landschaft der Software-Projekte nicht in Ordnung und entspricht dem Bild aus genannten Studien im Bereich Recherche. Die befragten Personen ergeben einen repräsentativen Querschnitt. Es wurde bei der Befragung auch ausgiebig auf die sozialen Faktoren eingegangen. Dies ist mit in die Entwicklung des neuen Ansatzes geflossen.

Die Aussagen der Standish-Group zu positiven Erfolgsfaktoren im Eingang, haben die Idee zu dieser Arbeit beeinflusst.
Zitat [16, S.5]:

„In other words, for success you need:
- *1) A highly skilled executive sponsor;*
- *2) A highly competent team;*
- *3) A good project manager;*
- *4) Prototypes with fast user feedback;*
- *and 5) Transparent progress tracking.“*

Möchte man von diesen Aussagen ableiten, ergibt sich ein Bild eines Softwareprojekt-Managements, welches auf optimale Teambildung fokussiert. Diese Arbeit hat versucht, diese Aussagen mit Methoden zu hinterlegen, welche die Wahrscheinlichkeit des Erfolges von Software-Projekten signifikant erhöhen. Dabei wurde auf bestehende Erfahrungen aus der Recherche von Beschreibungen positiver Beispiele als auch auf eigene Erfahrungen zurückgegriffen. Der vorgestellte Ansatz dient dazu, diese Erfolgsfaktoren im Sinne positiver Projektabschlüsse zu erfüllen. Nach Betrachtung der Vorteile der einzelnen Methoden wurde die Beschreibung für einen Ansatz gewählt, der in den agilen Bereich einzuordnen ist. Dieser Ansatz wurde aber von Beschränkungen befreit und um die Vorteile von Kanban-Systemen erweitert. Da Kanban-Systeme einen Change-Management Ansatz darstellen und keine Rollenbeschreibung vorgeben, war es erforderlich, die Rollen von Agierenden neu zu bestimmen. In Anlehnung an die Rollenbezeichnungen von bestehenden Methoden wurden Namen gewählt. Die Definitionen der Rollen wurden jedoch an die Philosophie einer neuen Methode geknüpft. Diese Methode beschreibt im wesentlichen einen Change-Management Ansatz in bestehende Strukturen von Software-Projekten. Dazu wurde die Betrachtungsweise beginnend von der Geschäftsführung und der Unternehmensstruktur über das Management bis zu den Teams und letztendlich dem Kunden gelegt. Es war erforderlich eine Beschreibung für alle Steakholder anzubieten. Dies bedingt eine Unternehmenskultur, die dem

entspricht. Kaizen ist solch eine Unternehmenskultur. Sie bedingt eine Geschäftsführung, die MitarbeiterInnen diese Möglichkeit gibt. Die Geschäftsführung stellt eine Management-Ebene zur Verfügung und baut eine Unternehmenskultur auf, die selbstorganisierende Teams ermöglicht. Das Stichwort hierfür ist Vertrauen. Die MitarbeiterInnen fühlen sich sicher durch dieses erzeugte Umfeld und arbeiten daran, es weiter zu verbessern.

Im Laufe der Vergangenheit wurden viele verschiedene Ansätze im Software-Projektmanagement versucht, teilweise vermischt mit Change-Management Ansätzen. Die einhundertprozentige Lösung wurde nicht gefunden. Entweder waren die Ansätze zu abstrakt oder viel zu technisch gegliedert.

Es wurde berücksichtigt, dass neue Projekt-Management Ansätze eigentlich Change-Management Ansätze darstellen. Wichtig war zu erkennen, dass von der Geschäftsführung bis hin zu den MitarbeiterInnen alle in den Prozess von Veränderung einbezogen werden müssen. Die rein technische Sicht des Projekt-Managements ist nicht erwünscht. Dies soll vor Allem beim Recruiting berücksichtigt werden. Eine nachhaltige Unternehmenskultur zu entwickeln, mit dem Ziel eines hohen sozialen Kapitals, ist der Schlüssel zur Stärkung des Unternehmens. Selbstorganisation ist der Schlüssel zu erfolgreichen Teams. Aber nicht um jeden Preis, sondern gut begleitet von dermaßen befähigten Personen und dem Management. Im Vergleich der bestehenden Projektmanagement-Methoden konnten die Unterschiede erklärt werden. Eine besondere Rolle kommt der Projektmanagement-Methode SCRUM, aus den agilen Ansätzen zu, da Sie so verbreitet ist. Es konnte nicht bestimmt werden, ob SCRUM eine qualitativ gute Methode ist. Immerhin haben in der Befragung (3.2.2, Abb. 14) 59% der Befragten bekannt gegeben, in einem Projekt mit SCRUM als Projekt-Management Methode zu arbeiten. Bei der Beurteilung der Projekterfolge gaben die Befragten (3.2.2, Abb. 12) zwar zu 85% Projekterfolge an, aber davon nur ein kleiner Teil (ungefähr 30%), dass es ein voller Projekterfolg war. Methoden wie SCRUM haben den Nachteil, dass Sie sehr starr in der Organisation sind. Auch die eingesetzten Metriken ergeben nicht genug Aussagekraft. Wird SCRUM noch dazu nicht vollumfänglich eingesetzt, sondern mit anderen Methoden vermischt, löst sich die Wirkung der Agilität von SCRUM fast vollständig auf. SCRUM scheint deswegen so verbreitet zu sein, da die Art zu arbeiten eher unserem westlichen Gedankengut entspricht, wo Wettkampf anstatt Teamgeist vorzufinden ist. Der Grundgedanke von SCRUM ist wie in allen agilen Methoden, einen iterativen Ansatz zu finden. In kleineren Abständen an den Kunden zu liefern, ist eines der Ziele. Scrum Teams beziehen Ihre Arbeitspakete (sogenannte User stories) aus dem Backlog, welches vom Product Owner aufbereitet wird, der wiederum die Schätzungen der einzelnen Aufwände mit dem Kunden vorgenommen hat (Estimation). Der Product Owner entnimmt die nächsten Schritte für den kommenden Sprint (Arbeitspakete aus dem Backlog) und das Team schätzt die Aufwände für den Sprint nochmals ab. Im laufenden Sprint wird eine sogenannte Retrospektive abgehalten. Sie soll dem Team ein Feedback über den letzten (vorhergehenden) Sprint ermöglichen. Feedback-Loops sind absolut erforderlich für die Entwicklung der Teams. Nur so können sie Fehler erkennen und ändern. Kanban-Systeme hingegen visualisieren ständig ihre Funktionsweise, den Durchsatz im System, die Schwachstellen und die Ergebnisse. Diese Mittel sind geeignet, die Teams und das Management mit Informationen zu versorgen, um ausreichendes Feedback zu generieren. Dies dienst erstens der

Motivation von Teams und eröffnet die Möglichkeit der Veränderung. Kanban Systeme haben durch die permanente Messung absolute Vorteile im Einsatz.

Grundsätzlich ist Scrum erdacht als Methode, welche Teams die Möglichkeit einräumt, sich selbstständig zu organisieren. Sprints sind in einem Projekt immer gleich lang. Als Maßeinheit für Arbeitspakete wird die Einheit sogenannter Story-Points genommen [1, S.332]. Dies ist eine zeitlose Einheit, welche lediglich die Aufwände von Arbeitspaketen in Verhältnis setzt. Jedes Paket wurde vom PO mit Story-Points versehen, um die Größe des Aufwandes darzustellen. Die Summe der geschafften Story-Points ist die sogenannte Velocity eines Teams. Leider wird Scrum als Methode oft korrumpiert durch falsche Schätzungen, welche wiederum dazu führen, dass Sprints nicht fertiggestellt werden können. In der Praxis ist zu sehen, das Teams, die einmal einen Sprint nicht schaffen, es bis an das Projektende nicht mehr ändern können. Ein Grund dafür mag in dem Umstand liegen, das Product Owner und das Management im direkten Kontakt zum Kunden oftmals zu optimistisch die Aufwände schätzen. Die Aufteilung der Story-Points und deren Beziehung zueinander wird dann verzerrt. Dies hat eine fatale Auswirkung auf die Teams. Teams die mit Ihrer Velocity ständig hinterher hinken, sind psychologisch gesehen sehr angespannt und deren Leistung sinkt weiter. Das Ziel kontinuierlicher Auslieferungen erreicht nach eingehender Betrachtung eine Methode wie Kanban wesentlich nachhaltiger, da nicht nur die Iteration in Betrachtung steht, sondern überhaupt der gesamte Fluss der vorzunehmenden Tätigkeiten in einem Projekt.

Der neue Ansatz lehnt sich an Kanban-Systeme an und die Rolle des PO wird ersetzt durch den Request Manager (RM), der jedoch nur die Pflege des Backlogs und die Priorisierung vornimmt. Das Team selbst entnimmt die Arbeitseinheiten aus dem Backlog. Damit ist die Selbstorganisation garantiert. Das Team hat mehr Möglichkeiten, die erforderliche Zeit richtig zu nutzen. Kanban-Systeme erzeugen einen kontinuierlichen Fluss und haben dazu verschiedene Hilfsmittel im Einsatz. Visualisierung und Metriken ergeben aus Kanban-Systemen eine wesentlich weitreichendere Aussage als die einfache Messung der Velocity in SCRUM. Grundsätzlich ist der Aufwand, SCRUM als Projekt-Management Methode einzuführen sehr hoch. Es erfordert meistens einen hohen Schulungsgrad der MitarbeiterInnen (aller Agierender) und der Geschäftsleitung sowie dem Management. Im SCRUM erfolgreich leben zu können, müssen die Grundparameter von SCRUM erfüllt werden, da sonst das System nicht gelebt werden kann und SCRUM mehr Behinderung wird als ein Vorteil. Wichtig ist die Bedeutung der Akzeptanz der Agierenden. [1, S.126] erklärt, dass der Einsatz agiler Methoden meist mit Widerständen der MitarbeiterInnen konfrontiert ist. SCRUM stellt die bisher gelebten Methoden grundsätzlich auf den Kopf, da so ziemlich alles anders gemacht wird als vorher. Dieser Umstand weist auf einen besonders hohen Überzeugungsaufwand der MitarbeiterInnen hin. Auch aus diesem Grund wurde auf eine möglichst einfache und selbsterklärende Arbeitsweise im System geachtet, um die Widerstände bei den Mitarbeitenden auf ein Minimum zu reduzieren. Es wurden Hilfsmittel zur Visualisierung und zur Messung vorgestellt, die einfach einzusetzen sind. Das Backlog ist die Basis des Workflows. Der RM pflegt und priorisiert das Backlog. Es ist idealerweise elektronisch geführt. Ein Tool der Wahl ist sicherlich Atlassian- JIRA [21] für professionelle Einsätze. Die Teams entnehmen die Tasks aus dem Backlog und reihen sie in den Bearbeitungsfluss ein. Wichtig ist, dass es aktuell ist. Dafür verantwortlich ist der RM. Wenn das Backlog nicht korrekt und ständig gefüllt ist,

leidet die Arbeitsleistung der Teams. Wichtig ist, dass das Backlog mit korrekt dimensionierten sowie korrekt priorisierten Aufgaben gefüllt ist. Dies beeinflusst maßgeblich, wie der Berbeitungsfluss läuft. In Kanban-Systemen soll niemals ein Engpass entstehen. Es soll aber vor allem der Durchfluss konstant sein. Dazu braucht es einen ständig gefüllten Buffer an Arbeitseinheiten durch das Backlog. Stimmt die Basis, kann das System reibungsfrei laufen. Zur Visualisierung wird ein Kanban-Board eingeführt, da zum Beispiel SCRUM-Boards keine genügend große Aussagekraft anbieten. Vor allem ist eine permanente Visualisierung des Durchsatzes erforderlich. Das Kanban-Board hat einen Anfang und ein Ende. Dies ist wichtig, um zu verstehen, wie Kanban funktioniert. Zwischen Anfang und dem Ende wird die Durchlaufzeit betrachtet. Diese ist das Maß, der Durchsatz. Ein Task läuft von links nach rechts durch jede Spalte. Ein Task kann auch einer Serviceklasse zugeordnet werden. Damit lassen sich bei Auswertungen die einzelnen Bereiche klarer trennen. Am Board wird an manchen Spalten ein WIP-Indikator eingetragen. Dieser stellt die WIP-Begrenzung dar. Nur so viele Karten wie angegeben dürfen in die Spalte eingereiht werden. Das heißt, läuft das System nicht korrekt, dann bildet sich automatisch ein Stau vor der vollen Spalte. Dies ist die Möglichkeit sofort zu erkennen, wo die Probleme liegen. Ein weiteres gewichtiges Argument für die Begrenzung des WIP, ist die Regulierung der Qualität über den Durchsatz. Durch Umschichtung im Team oder beseitigen von Behinderungen wird der Fluss wieder bereinigt. Der Status des gesamten Systems ist also hier ersichtlich. Kanban-Board's können hierarchisch angeordnet werden, um Board's von Teams zu einem übergeordneten Board zusammenzufassen. Dies vor allem bei sehr großen Projekten sinnvoll. An einem Kanban-Board ist es auch möglich, horizontale Linien einzutragen, sogenannte Swim-Lanes. Diese dienen dazu Bereiche zu trennen wie zum Beispiel Change-Requests, Wartung usw. Man kann auch physisch durch die horizontale Reihengröße die Verhältnisse darstellen. Einzelne Tasks werden normalerweise mit Aufgabentypen versehen und sollten ein Eingangsdatum besitzen, um die Durchlaufzeit messen zu können. Die zeitliche Größenordnung von einzelnen Tasks sollte die Gruppen klein – mittel – groß umfassen. Je nach Projekt mögen dies unterschiedliche Größen sein. Jedes Unternehmen wird einen eigenen Sinn dafür entwickeln, wie die Größenordnungen sind. Es können auch sogenannte Queues eingeführt werden. Sie dienen als Puffer. Werden diese Puffer ständig voll, dann produzieren Teams auf Halde („Waste“). Dieser Umstand ist nicht erwünscht. Jedes Ticket wird für jeden Tag markiert, an dem es nicht bewegt wird. So werden Blocker rasch erkannt. Das Board wächst mit der Entwicklung des Teams mit. Je reifer ein Team ist, desto komplexer und dadurch feiner ist das Board entwickelt. Die Teams können dadurch auf einen Blick viel weitreichendere Aussagen bezüglich des Prozessflusses und der eigenen Leistung treffen als mit einfachen Board´s. Um den Durchsatz (am Board ersichtlich) messen zu können, werden Diagramme eingeführt, die verschiedenen Aussagequalitäten mit sich bringen. Es ist zum Einen das Cumulative Flow Diagram (CFD). Das CFD zeigt an, ob ein Kanban-System richtig funktioniert. Es zeigt die Menge der bearbeiteten Aufgaben über die Zeit an. Ein konstant laufendes System würde glatte Linien in Ihrer Höhe stabil zeigen. Es kann auch die Schwankungsbreite über die Zeit festgestellt werden. Veränderungen im Team oder der Aufgabenbereiche sowie Veränderung in anderen Bereichen werden im CFD direkt ersichtlich. Es zeigt deutlich den Status des Systems an. Weit entwickelte Teams und deren Organisation können aus diesen Daten Größenordnungen wie Zeit oder Aufwand ableiten. Hingegen können Burndown-Charts (in SCRUM gerne verwendet) nur die Aussage treffen, ob das Team die Ideallinie verlässt oder nicht und

dies noch dazu nur am Ende eines Sprints und nicht kontinuierlich. Die Informationen aus CFD und mittlerer Durchlaufzeit sind wesentlich weitreichender und vor allem zeitnahe Aussagen. Teams, welche sich in dieses System einfügen, lernen sich über die Messungen selbst zu balancieren. Dieser Effekt ist in keiner der bereits genannten agilen Methoden dermaßen erkennbar. Der Fokus der Beschreibung dieses Ansatzes lag auf Einfachheit. Je einfacher und effektiver die Rollen-Struktur und die Hilfsmittel sind, desto geringer sind die Hürden.
Aus diesem Grund wurde ein einfacher, skalierbarer Ansatz von Rollen-Struktur gesucht. Der Request-Manager (RM), sammelt Anforderungen von verschiedenen Personen oder Abteilungen oder Kunden. Er priorisiert und pflegt sie in das Backlog ein. Er steht in Kontakt mit den Stakeholdern und unterstützt ihre Interessen.Der Priorisierung kommt besondere Bedeutung zu, da sie darüber entscheidet, in welcher Reihenfolge die Anforderungen in das System fließen. Dies beeinflusst in hohem Maße die Ergebnisse. Es gibt jedoch nicht die Arbeitsabfolge vor, dies macht das Team, anders als bei Methoden wie SCRUM. Dies ist ein besonders wichtiger Aspekt. Er entschärft quasi das Spannungsverhältnis, Wunschliste des Kunden vs. Sinnvolle und effektive Reihenfolge der Abarbeitung. Er stellt auch die Produkt-Vision dar. Dies ist die Sicht auf das Endergebnis, im Sinne des Kunden und in Absprache zwischen ihm und dem Kunden. Die Teams gehen auf ihn Proaktiv zu, wenn Sie Informationen brauchen. Der RM ist nicht zu verwechseln mit dem PO aus SCRUM, welcher dem Team die Arbeitspakete aus dem Backlog vorgibt. Der RM schirmt die Anfragen der Kunden vom Team ab und kommuniziert sie eindeutig weiter, in Richtung Team. Er hält mir dem Team Synchronisierung-Meetings ab um die Priorisierung der Tasks zu erläutern. Er definiert mit dem Kunden die Termine für die Lieferungen von Testversionen. Er synchronisiert dieser Termine, mit dem kontinuierlichen Fluss, der Teillieferungen durch das Team. Er gibt das Feedback aus den Akzeptanz-Tests der Kunden direkt an das Team weiter. Dies dient dem Team zur Selbstreflexion.

Um das hohe Maß an emotionaler Intelligenz und sozialen Fähigkeiten der Teammitglieder zu erfüllen, aber auch zu fördern (wie in den Erfolgsfaktoren erforderlich), kommt eine besondere Bedeutung der Rolle des Team-Coach zu. Er ist, idealerweise, eine Person von Außen. Eine Person von innerhalb des Unternehmens kann möglicherweise nicht objektiv genug handeln. Vor allem in Hinblick auf positive Änderung der Unternehmenskultur. Der Coach soll objektiv auf die Teams blicken können und, idealerweise, nur der Geschäftsführung unterstellt sein. Dies befreit den Team Coach von Einzelinteressen mittlerer Entscheidungsträger. Er behält so die Möglichkeit, die Teams frei unterstützen zu können. Der Coach ist eine Person welche die nötigen fachlichen, aber vor allem die psychologischen, emotionalen und sozialen Skills mitbringt. Der Team Coach hilft dem Team das Bewusstsein zu entwickeln, dass als Gruppe zu handeln mehr Vorteile bringt, als einzeln zu handeln. Er hilft dem Team in der Organisation des Boards und achtet darauf, das das Board richtig gehandhabt wird. Er erklärt dem Team die Vision eines geänderten Projektmanagements und den Vorteilen der Selbstorganisation. Er hilft dem Team den Durchsatz zu messen, sowie den WIP auf die richtige Größe zu beschränken. Er hält die Prinzipien aufrecht die zur Einführung eines neuen Ansatzes gehören. Er begleitet langfristig. Dies ist ein Ansatz zur Nachhaltigkeit. Er berät die Geschäftsführung in schwierigen Fragen, vor allem während der Eingangsphase. Er berichtet der Geschäftsführung von Veränderungen, und versucht das gegenseitige Verständnis zwischen Geschäftsführung und Teams aufrecht zu halten. Im laufe der Zeit hilft er dem Team sich weiterzuentwickeln und die Sicht auf die eigene Arbeitsweise immer feiner zu strukturieren. Dazu bildet er mit

dem Team die immer feiner werdende Arbeitsweise, auch in der Visualisierung, aus. Die Messinstrumente werden weiter ausgebaut, damit sich das Team selbständig Kenngrößen entwickeln kann, um die eigene Leistung besser zu beurteilen. Dadurch hilft er dem Team zur Selbstreflexion und unterstützt damit die wachsende Perfektion.
Ein ebenfalls zu beachtenswerter Punkt, vor allem in Hinblick auf gelebte agile Methoden und einer gewünschten Veränderung in der Unternehmenskultur. Die Rollen des Global Architect und des Toolsmith, sind Ergänzungen in Hinblick auf die technische Sicht des Projekt-Managements und nicht unbedingt verbindliche Rollen. Sie dienen dem Punkt der wachsenden Perfektion und um wiederkehrende Tätigkeiten zu vereinfachen.

Ein wesentlicher Punkt in der Veränderung von Software-Projektmanagement Methoden, ist die Verhinderung von Widerstand der Beteiligten. Aus diesem Grund, ist eine sensible Vorgehensweise bei der Einführung einer neuen Methode erforderlich. Diese Aussage ergibt sich aus den Erfahrungen von [1] und [10, S.28] (beeinflusst von Donald Reinertsen und Eliyahu Goldratt mit seinen Werken zur Engpasstheorie). Zuerst auf Qualität zu fokussieren, ist ein Bereich technischer Natur, welcher durch den Einsatz von Standard-Methoden auf technischer Seite verbessert werden. Methoden wie Pair-Programming und Test-Driven-Development helfen dabei. Nächstens den WIP zu begrenzen, ist eine wichtige Maßnahme, die Teams dazu bringt sich auf das Wesentliche zu konzentrieren. Den WIP begrenzen ist das Mittel der Wahl um Qualität zu erzeugen. Daraus resultiert, dass Teams ihr Selbstbewusstsein stärken können. Die Motivation steigt. In einem weiteren Schritt, kann nach Stabilisierung der Qualität, der Durchfluss gesteigert werden bis zu einem Maß welches die angestrebte Qualität halten kann. Es sollte öfter gut funktionierende Software zur Verfügung stehen, als vorher. Dies sollte auch die Stimmung im Team merklich erhöhen. Äußerem stärkt es die Vertrauensbildung zum Kunden. Das Feedback sollte positiver werden. Im nächsten Schritt kommt die Rolle REQUEST OWNER verstärkt zum Einsatz. Der RO beginnt richtig zu priorisieren. Damit kann das Management die Reihenfolge der zu liefernden Artefakte besser auf den Kundenwunsch abstimmen. Es geht also darum, zuerst einen konstanten Fluss zu erzeugen, und dann die Reihenfolge der Lieferungen zu kontrollieren. Das Team hat also mehr Zeit sich an dieser Arbeitsumgebung zu gewöhnen, und der Übergang geschieht sanfter und logischer. Am Ende werden die Quellen der Variabilität betrachtet (interne und externe). Dies dient ebenfalls der Veränderung der Unternehmenskultur, hin zu wachsender Perfektion, und verändert möglicherweise interne Strukturen. Wie schon beschrieben wurde bewusst ein Change-Management Ansatz gewählt, um auch die Rahmenbedingungen für erfolgreiche Projekte besser schaffen zu können.

Ziel war es eine Beschreibung zu liefern, wie man diesen Ansatz schrittweise in bestehende Projekt-Strukturen oder Unternehmens-Strukturen einführen kann, ohne einen Big-Bang Effekt. Der Ansatz baut auf das Verständnis aller Beteiligten und die Erkenntnis der Vorteile, auch im Sinne einer nachhaltigen Unternehmenskultur. Dieser Ansatz erhöht die Wahrscheinlichkeit auf Erhöhung des sozialen Kapitals. Dies dient der Nachhaltigkeit und der Wertschöpfung. Die Wahrscheinlichkeit, Software-Projekte erfolgreich abschliessen zu können, steigt durch die Vorgehensweise aus diesem Ansatz *(vgl. positive Beispiel in 3.1 von [1] und [10])*.

4.1 Evaluierung

Es wurde versucht, aus der Erfahrung bestehender Methoden zum Software-Projektmanagement, die best-practices Ansätze zu entnehmen und zu ergänzen, um sie in eine neue Struktur zu setzen. Diese Struktur soll die Wahrscheinlichkeit erhöhen ein Projekt erfolgreich abschliessen zu können. Es war im Umfang dieser Arbeit nicht möglich, die Arbeitsweise des vorgestellten Ansatzes vollumfänglich zu evaluieren. Dazu wäre ein sehr langer Zeitraum, sowie die Evaluierung mehrere Projekte parallel notwendig, um genügend Daten zu sammeln. Es wurde in einem laufenden Projekt die Visualisierung im Projektteam eingeführt und die Grundvision dargestellt. Es war jedoch auf Grund des Projektfortschrittes nicht möglich weitere Schritte einzuleiten. Daher war keine weitere Evaluierung möglich. Es kann aber, zur teilweisen Evaluierung, auf bestehende Daten von Beispielen verwiesen werden, welche die positive Wirkung der dargestellten Prinzipien und Hilfsmittel untermauern. Es wurde ein Ansatz auf Basis dreier Säulen vorgestellt: Gerüst(Werte) – Rollen – Hilfsmittel. Es soll durch recherchierte Beispiele versucht werden, diese Teilbereiche, einer Evaluierung auf Basis bekannter Fakten zu unterziehen.

Um den Bereich der Werte sinnvoll umsetzen zu können, bedarf es auch der geeigneten Unternehmenskultur. Es wurde an Kanban-Systemen Anleihe genommen, um einige, der für einen neuen Ansatz notwendigen Denkweisen und Hilfsmittel, zu implementieren. [10, S.67] berichtet über solch einen Fall, nach dem Einsatz eines Kanban-Systems, wo das Top-Management die positiven Entwicklungen sehen konnte nachdem, ausgehend von einem einzigen Team, eine tiefgreifende Änderung in der Unternehmenskultur entstand. Weiters waren Erfolge direkt messbar wie [10, S.54] die Ergebnisse eines Umstellungs-Projektes bekannt gibt, bei dem ein Team ein Kanban-System in eine bestehende Projektmanagement-Struktur eingeführt hat. Das Team hatte es geschafft, die Termintreue auf 98% zu steigern, den Durchsatz an Anforderungen zu verdreifachen, sowie die Durchlaufzeit um 90% zu reduzieren und dies in wenigen Monaten. Es war also sinnvoll aus Kanban-Systemen Teile zu integrieren. Die Philosophie dahinter, richtig eingesetzt, sollte nachhaltig im Unternehmen soziales Kapital entwickeln. Dies war auch die Sichtweise auf den Wert „Work-Life-Balance“.

Der Betrachtung der Menschen in Software-Projekten, kommt eine besondere Bedeutung, im Hinblick auf das Recruiting zu. Vor allem im Hinblick auf zusätzliche, nicht fachliche Skills, sowie emotionale Reife, oder soziale Kompetenz. Die Bedeutung einzelner, immer wiederkehrender Typen (Perfektionist, Code-Held und dgl.), haben [3] aus Ihren Projekterfahrungen heraus skizziert, sowie auch die Konsequenzen. Im Abschnitt „2.4 Recruiting – Neues Recruiting“ , wurde ein Beispiel von [10] genannt, welches in einem Unternehmen zwei Teams verglichen hat. In diesem Beispiel war das am Papier schwächere Team (in Erfahrung und Skills) erfolgreicher, da die Teammitglieder besser zusammengearbeitet hatten. Um die MitarbeiterInnen in Teams richtig zusammenzusetzen, ist also eine neue Art des Recruiting erforderlich und dieses Beispiel [10, S.31], kann als positive Bestätigung dazu gesehen werden. Herkömmliche Recruiting-Prozesse, die vor allem auf fachliche Skills abzielen, hätten möglicherweise die Teams falsch eingeschätzt. Eine weitere zur Evaluierung dienliche Information liefert [1, S.435], dass Organisationen eher Einzelpersonen als Teams rekrutieren. Dies liegt an der bestehenden

Unternehmenskultur westlicher Tradition. Es sollte also Vorteile bringen, die Recruiting-Prozesse gemäß dem vorgestellten Ansatz anzupassen. [5, S.29] beschäftigt sich mit der agilen Softwareentwicklung in Großprojekten und sieht den häufigsten Grund für negative Entwicklungen in mangelnder Kommunikation und zitiert [5, S.29] *„Es scheint eine Tatsache zu sein, dass so gut wie kein Projekt aufgrund von spezifischen Technologien, Werkzeugen oder Ähnlichem fehlschlägt. Der Hauptgrund für fehlgeschlagene Projekte ist fast immer fehlende oder nicht funktionierende Kommunikation“*. [13] sieht das klassische Projektmanagement zu technisch orientiert, Zitat [13, S.7] *„Es zeigte sich jedoch, dass trotz professionellem klassischen Projektmanagement die Transformation vom Ist- zum Soll-Zustand oft nicht erfolgreich bewältigt werden kann. Analysen haben ergeben, dass das Projektmanagement zu technisch orientiert ist und die menschlichen Aspekte eines sozialen Systems nicht ausreichend berücksichtigt. Diese Aspekte der Veränderung zu berücksichtigen und damit den Soll-Zustand erreichbar zu machen, ist Aufgabe des Change Management. Die Durchführung der Change Management Elemente erfolgt im Rahmen des Projektmanagements“*. Unter dem Hinweis all dieser Informationen und Projekterfahrungen daraus, ergab sich die Notwendigkeit, einen Change Management Ansatz mit einzubeziehen. Aus diesem Grund, wurde Anleihe an Kanban-Systemen genommen, die solch einen Ansatz darstellen. Es wurde beim Recruiting beginnend, die nötige Vorgangsweise im Hinblick auf selbstorganisierende Teams beschrieben. Es sollten damit die Werte „Selbstorganisierte Teams“, „Kommunikation“, „Die richtige Wahl des Teams“, „Gleichberechtigung“ und“Selbstreflexion“ ausreichend berücksichtigt sein.

Die Auswahl der Rollen, ergab sich aus der Sichtweise des neuen Ansatzes, nahezu natürlich. Es gibt bestehende Rollenbeschreibungen aus den bestehenden agilen Methoden. Diese Beschreibungen, wurden auf Basis bestehender Informationen evaluiert und nötigenfalls angepasst. [5, S.67] beschreibt die Rolle des Coach. Zitat [5, S.67] *„Jeder einzelne Coach hat (natürlich) die Aufgabe, vor allem Feedback in seinem Team einzuholen sowie Defizite und (potenzielle) Probleme zu entdecken (und am besten gleich zu lösen).“,* sowie Zitat [5, S.68] *„Die Coaches als MitarbeiterInnen des Kommunikationsteams müssen darauf achten, dass sie nicht als Kontrolleure oder Aufseher, sondern als Vertrauensperson wahrgenommen werden“* und Zitat [5, S.69] *„Die Notwendigkeit dieser Rolle wird um Übrigen in den wenigsten Projektorganisationen erkannt, was es schwierig macht, sie zu etablieren.“*. Es wurde darum auf den Einsatz eines Coach im beschriebenen Ansatz verwiesen. Auch im Hinblick auf die Güte für den Einsatz von Kanban-Systemen. Die Rolle des Global Architect wurde implementiert, da es sinnvoll erschien, für die Teams eine einheitliche Linie der Entwicklung aufzubauen. Es sollen auch keine gemeinsamen Aktivitäten doppelt getan werden. Die Rolle des Global Architect kann helfen diese Tätigkeiten zu bündeln. Es findet sich dazu ein Hinweis in Zitat [10, S.138] *„Bei den gängigsten agilen Verfahren, ist die Rolle eines Architekten nicht vorgesehen.“* und weiters Zitat [10, S.138] *„Dabei übernehmen alle Mitarbeiter eigenverantwortlich momentan anstehende Aufgaben. Auch wenn dies für große Teams ebenfalls ein Ziel sein könnte oder sollte, so stellt sich doch heraus, dass dies schlichtweg nicht möglich ist, da ansonsten die Entwicklung vollkommen unkoordiniert auseinanderläuft.“*. Die Rolle des Request Managers, ist angelehnt an die Rolle eines Product Owner aus SCRUM. Allerdings wurde die Rolle in einem wesentlichen Punkt abgeändert. Um ein funktionierendes, auf Kanban aufbauendes System, herstellen zu können, ist es unerlässlich das Teams vollständig selbstorganisierend arbeiten können. Darum muss

diese Rolle abgeändert werden, im Punkt der Einteilung der Arbeitspakete aus dem Backlog. Während der Product Owner aus SCRUM die Arbeitspakete vorgibt und sie in die Sprints zur Bearbeitung einpflegt, führt der Request Manager aus dem vorgestellten Ansatz lediglich die Priorisierung der Arbeitspakete im Backlog durch. Das Team selbst entnimmt sie, nach Wunsch, zur Bearbeitung. Die Rolle des Toolsmith wurde zur Arbeitserleichterung der Teams eingeführt, vor allem für Arbeitsbereiche in Entwicklung und Test, die außerhalb der Tätigkeiten des Global Architect stehen. Diese Rolle wurde aus Feature Driven Development (FDD) entlehnt. [4, S.74] erwähnt diese Rolle. Die Rolle des Teams wurde dahingehend aufgebaut, dass sie gemäß der Beschreibung im vorgestellten Ansatz, die Selbstorganisation in den Vordergrund stellt. Die Rolle des Teams wird ausreichend in allen agilen Methoden dargestellt und ist eine Entlehnung daraus. Im Gegensatz zu SCRUM wurden jedoch Änderungen vorgenommen, die wesentlich sind. Das Team selbst entnimmt die Arbeitspakete aus dem Backlog. Es sollte dabei die Priorisierung berücksichtigen, je nach Entwicklungs- und Einführungs-Stand des Teams. Das Team lernt auch, über Metriken und Visualisierung, sich ständig zu reflektieren. Dies im Gegensatz zu SCRUM, wo Reflexion in Abschnitten und nach einem Sprint stattfindet. Die Hilfe durch geeignete Visualisierung war erforderlich, um Engpässe im Moment des Entstehens zu entdecken. Dadurch ergibt sich auch die Möglichkeit des Ausbalancieren, auf Basis bekannter Fakten. Eine Beschreibung hierzu findet sich in [10, S.37] wo Bezug auf die positiven Effekte des Erkennens von Engpässen genommen wird. Die Rolle Kunde wurde im Grunde so definiert, wie sie in den meisten agilen Systemen gehandhabt wird. Allerdings mit einem Unterschied, der durch die Eigenart des vorgestellten Ansatzes bestimmt ist. Der Kunde priorisiert zwar mit dem Request Manager die Arbeitspakete, aber das Team ist durch seine Selbstbestimmtheit in der Art wie es die Arbeitspakete abarbeitet, vom Kunden entkoppelt. Nur durch die Priorisierung besteht ein Einwirken des Kunden. Dadurch sollt der Wert „Schnittstelle zum Kunden“ erfüllt sein. Die beschriebenen Rollen, unterliegen also zu einem gewissen Grad, den Erfahrungen aus den bestehenden agilen Methoden. Es wurden wo immer nötig die Änderungen und Ergänzungen eingesetzt und auch beschrieben, um die Grundlage der geforderten Werte erfüllen zu können.

Um die Werte „Anforderungsänderungen begegnen“ sowie „Funktion und Qualität“ ausreichend zu berücksichtigen, wurden Hilfsmittel wie Test-Driven-Development und Pair-Programming einbezogen. Dies sind anerkannte Methoden in der Software-Entwicklung, welche zur Steigerung der initialen Qualität einer Software eingesetzt werden. Der Wert „Anforderungsänderungen begegnen“ wird durch die Philosophie von Kanban-Systemen bestätigt. Die Erweiterung des Ansatzes um continuos-planning und continuos-integration erfüllt die Anforderungen hierzu. Dadurch werden die Werte „Kontinuierliche Lieferung an den Kunden“, sowie „Iterationen und regelmäßige Lieferungen“ erfüllt. Vor allem continuos-integration ist ein bekannter Ansatz, welcher bereits in bestehenden agilen Methoden erfolgreich zum Einsatz kommt.

Die nötigen Tools für den vorgestellten Ansatz wurden auf Basis folgender Information gewählt. [1, S.456] beschreibt die Problematik beim Einsatz von Metriken. Sie werden von außen angewandt und haben nur bedingte Aussagekraft. Die Messungen müssen auch über einen längeren Zeitraum durchgeführt werden. [10, S.31] beschreibt den Vergleich von Cumulative-Flow Diagrammen verschiedener Teams, und dass ein Zusammenhang zwischen WIP und der Qualität gelieferter Features besteht. Zitat in [10, S.34]: *„Es besteht ein kausaler Zusammenhang zwischen der*

Menge an WIP und der durchschnittlichen Durchlaufzeit – und dieser Zusammenhang verhält sich linear. In der Fertigungsindustrie ist dieser Zusammenhang als Littles Gesetz bekannt. Es besagt, dass in einem Produktionssystem, das alle Aufträge abarbeiten kann, der mittlere Bestand im System und die mittlere Durchlaufzeit direkt proportional zueinander sind. Je größer der mittlere Bestand im System ist, desto länger bleiben Stücke im System, und umso größer ist die mittlere Durchlaufzeit.“. Aus dieser Aussage liess sich ableiten, dass notwendigerweise WIP Begrenzungen eingeführt werden müssen und die durchschnittliche Durchlaufzeit bekannt sein muss, um die Qualität regulieren zu können. Außerdem soll es möglich sein, durch Visualisierung, den Status des laufenden Projekt-Systems einsehen zu können. Darum wurden Cumulative-Flow-Diagram und Messung der mittleren Durchlaufzeit eingeführt. Diese Messungen haben den Vorteil der Aktualität. Sie liefern im Moment und kumulativ, gute Aussagen bezüglich Funktion des Systems, sowie Qualität. Es wurde ein Kanban-Board in den Ansatz eingefügt, um den WIP begrenzen zu können, und die Teams die Möglichkeit zur Selbstreflexion haben. Die Visualisierung dient ebenfalls der Möglichkeit zum Ausbalancieren der Teams, da die Engpässe direkt ersichtlich werden, im Moment des Entstehens. Außer den Bereichen der Werte, Rollen, Hilfsmittel, wurde auch dem Thema der Betrachtung der Variabilität Rechnung getragen. [10, S.229] nimmt zu den Quellen von Variabilität dazu ausführlich Stellung. Zuletzt ist der Wert „Fachliche Qualifikation und Weiterbildung zu betrachten. [5, S.193] beschriebt dieses Thema und die Möglichkeiten der Weiterbildung. Es scheint also sinnvoll auf diese Möglichkeiten Bezug zu nehmen.

Abschliessend kann festgestellt werden dass, obwohl keine vollumfängliche Evaluierung möglich war, es genügend beschriebene positive Fakten zu den einzelnen Beschreibungen, wie Werte, Rollen und Hilfsmittel gibt. Es sollte sich also, beim Einsatz des vorgestellten Ansatzes, die Wahrscheinlichkeit erhöhen, dass Software-Projekte erfolgreich abgeschlossen werden können.

5 Anhang

5.1 Anhang I: Fragebogen

MULTIDISCIPLINARY
DESIGN
GROUP

0% ausgefüllt

1. Bitte geben Sie an welche Funktion Sie in Ihrem Unternehmen / Projekt bekleiden [PD01]

Sollten Sie freier oder externer Mitarbeiter sein, geben Sie bitte an welche Funktion sie hauptsächlich bekleiden.

Bspl: Sie sind Softwareentwickler und Projektleiter / Scrummaster zugleich , dann wählen Sie bitte die Funktion Projektleiter / Scrummaster!

- ○ Softwareentwickler
- ○ Projektleiter
- ○ Mitarbeiter des Fachbereiches ohne Bezug zur IT
- ○ Mitarbeiter des Fachbereiches mit Bezug zur IT
- ○ Scrummaster
- ○ Mitglied des Managements

2. Bitte geben Sie an, ob Sie in diesem Unternehmen / Projekt als externer oder interner Mitarbeiter arbeiten [PD02]

Sollten sie Angestellter eines Unternehmens sein, und andernorts in einem Projekt gemeinsam mit Unternehmensfremden Personen arbeiten, geben Sie bitte „externer Mitarbeiter an“

- ○ interner Mitarbeiter
- ○ externer Mitarbeiter

3. Bitte geben Sie an , in welcher Altersgruppe Sie sich befinden. [PD03]

- ○ < 30
- ○ 30 bis <45
- ○ >= 45

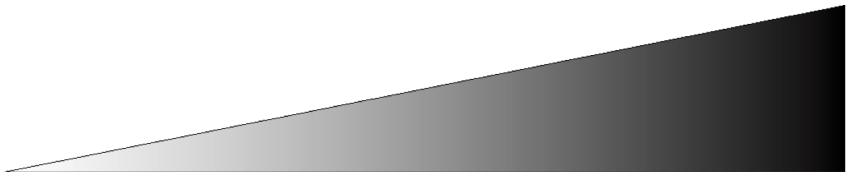
4. Bitte bewerten Sie Ihre Fähigkeiten [PD04]

Bewerten Sie durch klicken und oder schieben in die analoge Skala

Ich fühle mich bestens ausgebildet (HTL/FH/UNIVERSITÄT)	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich bin Autodidakt	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich lese oft Literatur zur Fortbildung	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich besitze Zertifikate für meinen Berufszweig	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich bin Spezialist in einem/einigen Bereichen	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich bin in einem breiten Anwendungsgebiet einsetzbar	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich bin flexibel einsatzbar	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich beherrsche alle gängigen Technologien soweit notwendig	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich fühle mich in den Projekten der letzten Zeit wohl	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich habe genügend Freizeit	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich bin kommunikativ	trifft überhaupt nicht zu		trifft voll und ganz zu
ich fühle mich in meinem Beruf an der richtigen Position	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich könnte noch etwas in meiner Arbeitsweise verbessern	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich möchte an Fortbildungsangeboten für persönliche Entwicklung teilnehmen	trifft überhaupt nicht zu		trifft voll und ganz zu

5. Bitte geben Sie an in welcher Projektgröße/Dauer sie normalerweise eingesetzt sind. [GT01]

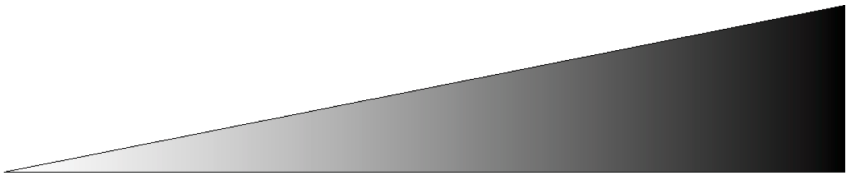
Geben Sie wenn möglich die Größe an die in den letzten Jahren am meisten zugetroffen hat. Die Angabe ist in AT = Arbeitstagen bewertet.



○ bis 60 AT ○ >60 AT bis 300 AT ○ >300 AT bis 600 AT ○ > 600 AT

6. Wir groß ist ihr Team im Durchschnitt [GT02]

Geben Sie die Teamgröße der letzten Jahre an die Sie am meisten vorgefunden haben. Gemeint ist die kleinste Einheit Team.



○ bis 4 Personen ○ 4 – 8 Personen ○ 8 – 12 Personen ○ 12 – 20 Personen ○ > 20 Personen

7. Wie würden Sie den Projekterfolg von Projekten bewerten, wo Sie mitgewirkt haben. Bewerten Sie bitte den Durchschnitt der letzten Jahre. Die Fragen sind nicht gegenseitig ausschliessend zu bewerten. Es kann in allen 3 Hauptgruppen bewertet werden. [EP01]

Als voller Erfolg würde ein Projekt bezeichnet werden welches in der geplanten Zeit im geplanten Kostenrahmen durchgeführt werden konnte. Als Mißerfolg würde ein Projekt bezeichnet werden welches trotz überlaufender Kosten und nichteinhalten des Zeitplans trotzdem NICHT fertiggestellt wurde. Alle anderen Projekte liegen zwischen diesen Extrema. Bitte bewerten Sie den Durchschnitt Ihrer Erfahrungen über die Zeit / Anzahl der Projekte in den letzten (bis zu 5) Jahren. Die letzten beiden Fragen dienen zur detaillierten Erfassung.

Das Projekt/ die Projekte war(en) meistens ein voller Erfolg	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Das Projekt/die Projekte konnten meist mit steigenden Kosten und verlängerter Projektdauer erfolgreich fertiggestellt werden	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Das Projekt/die Projekte konnten mit steigenden Kosten und verlängerter Projektdauer NICHT fertiggestellt werden	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
In meine Bewertung sind übermäßig positive Projekterfolge eingeflossen	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
In meine Bewertung sind übermäßig negative Projekterfolge eingeflossen	trifft überhaupt nicht zu	_____	trifft voll und ganz zu

8. Bitte bewerten Sie Ihre Erfahrungen in den Projekten der letzten Jahre. [EP02]

Im Team wird gegenseitig Hilfe geleistet	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Der Auftraggeber war allen bekannt	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Der Projektleiter war allen bekannt	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Die Projektziele waren allen bekannt	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Der zeitliche Ablauf des Projektes war allen bekannt	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Es standen ausreichend Personal zur Verfügung	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Es stand ausreichend Zeit zur Verfügung	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Es gab agile Projektmanagement-Methoden im Einsatz	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Es gab ausreichende Tätigkeitsbeschreibungen (Vorgaben, Workflow,Userstories)	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Die durchschnittliche Arbeitszeit pro Woche war 38 Stunden	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Es mussten oft Überstunden gemacht werden	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Ich durften keine Überstunden machen	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Ich habe private Zeit aufgewendet um mit den Anforderungen klar zu kommen	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
im Team haben sich alle gegenseitig respektiert	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Die größe der Arbeitspakete war richtig Dimensioniert, und damit kein Problem abzuarbeiten	trifft überhaupt nicht zu	_____	trifft voll und ganz zu

Ich hatte viele Arbeiten parallel zu erledigen	trifft überhaupt nicht zu		trifft voll und ganz zu
Teams mussten auf die Arbeit von anderen Teams/Mitarbeitern warten um weitermachen zu können	trifft überhaupt nicht zu		trifft voll und ganz zu
Es wurden ausreichend Meetings (Planen, Standup, Daily, Estimation, Retrospektive) abgehalten	trifft überhaupt nicht zu		trifft voll und ganz zu
Es wurden zu viele Meetings (Planen, Standup, Daily, Estimation, Retrospektive) abgehalten	trifft überhaupt nicht zu		trifft voll und ganz zu
Das Management hat unterstützend gehandelt	trifft überhaupt nicht zu		trifft voll und ganz zu
Das Management hat zu knappe Zeitvorgaben vorgelegt	trifft überhaupt nicht zu		trifft voll und ganz zu
Das Management hat über positive Dinge und (Teil)Projekterfolge berichtet	trifft überhaupt nicht zu		trifft voll und ganz zu
Das Management hat über negative Dinge (Kosten/Aufwand/Planung) berichtet	trifft überhaupt nicht zu		trifft voll und ganz zu
Das Management hat die Teams frei entscheiden lassen wie sie sich organisieren	trifft überhaupt nicht zu		trifft voll und ganz zu

9. Bitte benennen Sie welche Methoden des Softwareprojektmanagements Sie bereits kennen und auch Erfahrung damit gemacht haben [AG01]

Bitte geben Sie nur Methoden an, wo Sie auch aktiv mitgewirkt oder detailiertes Wissen dazu haben.

- ☐ Wasserfallmodell
- ☐ Adaptive Software Development (ASD)
- ☐ Crystal Methods
- ☐ Scrum
- ☐ Extreme Programming (XP)
- ☐ Feature Driven Development (FDD)
- ☐ Lean
- ☐ Kanban
- ☐ Scrumban
- ☐ Sonstige (Eigene Verfahren)

10. Bitte beantworten Sie die folgenden Fragen über Projektmanagement-Methoden mit Ja / Nein [AG02]

	nein	ja
Ich bin teil eine Teams mit agilem Projektmanagement	○	○
Ich bin Teil eines Teams mit Wasserfallmodell als Projektmanagement	○	○
Wir haben ein eigenes Projektmanagement entwickelt	○	○
Wir haben täglich unser Morgen-Meeting (Standup/Daily)	○	○
Wir haben einen fixen Rythmus der Sprints (Timebox, immer 1 Woche, 2 Wochen...)	○	○
Alle Teams im Projekt verfolgen die selbe Projektmanagement-Methode	○	○
Wir verwenden SCRUM	○	○
Wir verwenden XP (Extreme Programming)	○	○
Wir verwenden ein Wassefallmodell	○	○
Wir verwenden Crystal Methods	○	○
Wir verwenden Kanban	○	○
Wir verwenden Scrumban	○	○
Ich habe keine Ahnung welches Modell wir verwenden	○	○

11. Bitte beantworten Sie die folgenden Fragen zum Projektmanagement [AG03]

Bewertung durch klicken und oder schieben im Schieberegler.

Ich bin überzeugt das unser Projektmanagement Ansatz bestens ist	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich wünsche mir neue Ansätze des Projektmanagements	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich halte viel von den agilen Projektmanagement-Methoden	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich bewerte SCRUM und Ähnliche als gute Projektmanagement-Methode	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich bewerte das Wasserfallmodell und Ähnliche als gute Projektmanagement-Methode	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich bewerte Kanban und Ähnliche als gute Projektmanagement-Methode	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich bewerte XP (Extreme Programming) als gute Projektmanagement-Methode	trifft überhaupt nicht zu		trifft voll und ganz zu
Das Management hat die Projektmanagement-Methode vorgeschrieben	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir konnten selbst im Team die Art der Projektmanagement-Methode wählen	trifft überhaupt nicht zu		trifft voll und ganz zu
Unser Team unterstützt die Projektmanagement-Methode voll und ganz	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir haben kein Projektmanagement soweit ich es erkennen kann	trifft überhaupt nicht zu		trifft voll und ganz zu
Ich empfinde unser Projektmanagement nicht zeitgemäß	trifft überhaupt nicht zu		trifft voll und ganz zu
Unser Team unterliegt einem strengen Kontrollmechanismus (Zeit)	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir haben ausreichend Zeit für die Arbeit im Team	trifft überhaupt nicht zu		trifft voll und ganz zu
Würden wir anderes Projektmanagement leben, könnten wir Zeit und Aufwand sparen	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir haben ein BOARD zur Visualisierung unserer Tätigkeiten	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir haben ständig Überblick über die Tätigkeiten aller Beteiligten	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir praktizieren PAIR-Programming	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir praktizieren Test-Driven-Development (TDD)	trifft überhaupt nicht zu		trifft voll und ganz zu
Alle im Team sind gleichberechtigt	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir unterstützen uns gegenseitig im Team	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir können im Team frei arbeiten und uns selbst organsieren	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir haben feste/geplante Zeiträume(Timeboxes) für Teilarbeiten oder SPRINTS: meistens können die Teilarbeiten (SPRINTS) erfolgreich erledigt werden	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir haben feste/geplante Zeiträume(Timeboxes) für Teilarbeiten oder SPRINTS: meistens können die Teilarbeiten (SPRINTS) nicht erledigt werden	trifft überhaupt nicht zu		trifft voll und ganz zu
Wir haben keine festen/geplanten Zeiträume	trifft überhaupt nicht zu		trifft voll und ganz zu

12. Bitte bewerten Sie Ihren Umgang im Team.Gemeint ist ihr momentanes Projekt oder der Durchschnitt der Erfahrungen aus den letzten Projektjahren [TE01]

Im Team sind alle gleichgestellt	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Das Team hat die richtigen Mitarbeiter	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Alle im Team helfen und unterstützen gegenseitig	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Das Team organisiert sich selbst	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Das Team bekommt die Arbeitspakete vorgeschrieben	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Im Team ändern sich die Mitarbeiter öfter als notwendig	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Das Team hat ausreichend Platz zum arbeiten und die nötige Privatsphäre	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Wir visualisieren unsere Tätigkeiten an einem Board und jeder weis was der Andere tut	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Wir kommunizieren die meisten Dinge im Team per Email oder Telefon	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Wir kommunizieren die meisten Dinge im Team direkt verbal	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Im Team gibt es einen Teamführer der sagt was wie zu machen ist	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Das Team steht ständig unter Zeitdruck	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Das Team bekommt ausreichend Arbeitspakete zugeteilt (PO/Projektleiter)	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Das Team bekommt zuviele Arbeitsakete zugeteilt	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Das Team muss viel parallel erledigen	trifft überhaupt nicht zu	_____	trifft voll und ganz zu
Ich muss zu viel parallel erledigen	trifft überhaupt nicht zu	_____	trifft voll und ganz zu

MULTIDISCIPLINARY DESIGN GROUP

Vielen Dank für Ihre Teilnahme!

Wir möchten uns ganz herzlich für Ihre Mithilfe bedanken.

Ihre Antworten wurden gespeichert, Sie können das Browser-Fenster nun schließen.

Einladung zum SoSci Panel

Liebe Teilnehmerin,
lieber Teilnehmer,

das nicht-kommerzielle SoSci Panel würde Sie gerne zu weiteren wissenschaftlichen Befragungen einladen. Das Panel achtet Ihre Privatsphäre, gibt Ihre E-Mail-Adresse nicht an Dritte weiter und wird Ihnen pro Jahr maximal vier Einladungen zu qualitativ hochwertigen Studien zusenden.

E-Mail: Am Panel teilnehmen

Sie erhalten eine Bestätigungsmail, bevor Ihre E-Mail-Adresse in das Panel aufgenommen wird. So wird sichergestellt, dass niemand außer Ihnen Ihre E-Mail-Adresse einträgt.

Der Fragebogen, den Sie gerade ausgefüllt haben, wurde gespeichert. Sie können das Browserfenster selbstverständlich auch schließen, ohne am SoSci Panel teilzunehmen.

Bakk. Arnold Belohlavek, Institute of Design and Assessment of Technology, Technische Universität Wien - 2014

6 Literaturverzeichnis

1. *Mike Cohn, „Agile Softwareentwicklung, mit Scrum zum Erfolg“ , published 2010, ISBN 978-3-8273-2987-5*
2. *Dipl.Inform. Thomas Biskup, „Agile fachmodellgetriebene Softwareentwicklung für mittelständische IT-Projekte“, published 2009*
3. *Peter Hruschka, Gernot Starke, „Knigge für Softwarearchitekten“, published 2012*
4. *Jürgen Götzenauer, „Agile Methoden in der Softwareentwicklung“, published 2006*
5. *Jutta Eckstein, „Agile Softwareentwicklung in großen Projekten“, published 2011*
6. *Mario Mainz , Florian Bosten, Adam Cholewa, “Vergleich verschiedener Verfahren der agilen Softwareentwicklung“, http://winfwiki.wifom.de/index.php/Vergleich_verschiedener_Verfahren_der_agilen_Softwareentwicklung , accessed 17.03.2014*
7. *Muhammet Altindal, Frank Hinkel, Robert Zyla, „Abgrenzung der agilen Softwareentwicklung von klassischen Verfahren“, http://winfwiki.wifom.de/index.php/Abgrenzung_der_agilen_Softwareentwicklung_von_klassischen_Verfahren , accessed 17.03.2014*
8. *VERSIONONE, „7th annual state of Agile Development Survey“, published 2013*
9. *Stefan Berndes, Klaus Kornwachs, Uwe Lünstroth , “Softwareentwicklung: Erfahrung und Innovation, published 2002*
10. *David J.Anderson, „Kanban: Evolutionäres Change Management für IT-Organisationen“, published 2011*
11. *Günther Thoma: „Soziale Kompetenzen als Grundlage von Handlungskompetenzen in der betrieblichen Anwendungsentwicklung“, published 1994*
12. *Cleo Becker, Dr. Eberhard Huber, Die Bilanz des (Miss)-Erfolges in IT-Projekten published 2008*
13. *Ivo Eichler, Change Management für erfolgreiche IT-Projekte: Projektmanagement Praxis published 2010*
14. *Hans J. Jeebe, Diversity Management in IT-Projekten, published 2010*
15. *Sebastian Kammerer, Michael Amberg, Michael Lang, Führung im IT-Projekt: Fachliche und soziale Kompetenzen für den Projekterfolg, published 2012*
16. *The Standish Group International, Inc. http://blog.standishgroup.com/BigBangBoom.pdf, accessed 10.08.2014*
17. *Manifesto for agile Software Development, http://www.agilemanifesto.org/, accessed 10.08.2014*
18. *Feature driven development Feature Driven Development (FDD) and Agile Modeling , http://www.agilemodeling.com/essays/fdd.htm , accessed 10.08.2014*
19. *Alistair Cockburn, Crystal methodologies, http://alistair.cockburn.us/Crystal+methodologies, accessed 10.09.2014*
20. *www.agilemanifesto.org, Die agilen Prinzipien http://agilemanifesto.org/iso/de/principles.html , accessed 10.09.2014*
21. *Attlassian Jira, https://www.atlassian.com/de/software/jira , accessed 12.09.2014*
22. *Agile Tools (Open source), http://www.agile-tools.net/ , accessed 12.09.2014*
23. *Introduction to Test Driven Development http://agiledata.org/essays/tdd.html , accessed 12.09.2014*
24. *Feedback-Loops, http://www.extremeprogramming.org/map/loops.html, accessed 12.09.2014*
25. *Sosci Survey, https://www.soscisurvey.de/ , last accessed 12.09.2014*
26. *GNU-R , http://www.r-project.org/index.html, last accessed 12.09.2014*
27. *R Package Likert, http://jason.bryer.org/likert/, last accessed 12.09.2014*