Masterarbeit im Studiengang Computer Science and Media

Skalierung von Scrum

Erhebung und Evaluation von Herausforderungen und Vorgehensweisen

vorgelegt von

Pascal Naujoks

an der

Hochschule der Medien Stuttgart

am 01.03.2013

1. Prüfer:	Prof. Dr.-Ing. Oliver Kretzschmar
2. Prüfer:	M. Sc. Marc Bauer

HOCHSCHULE DER MEDIEN

NOVATEC make IT happen!

Eidesstattliche Erklärung

Hiermit versichere ich, Pascal Naujoks, an Eides Statt, dass ich die vorliegende Masterarbeit mit dem Titel: „Skalierung von Scrum“ selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der eidesstattlichen Versicherung und die prüfungsrechtlichen Folgen (§26 Abs. 2 Bachelor-SPO (6 Semester), § 23 Abs. 2 Bachelor-SPO (7 Semester) bzw. § 19 Abs. 2 Master-SPO der HdM) sowie die strafrechtlichen Folgen (gem. § 156 StGB) einer unrichtigen oder unvollständigen eidesstattlichen Versicherung zur Kenntnis genommen.

Stuttgart, den 01.05.2013

Zusammenfassung

Diese Arbeit evaluiert die Herausforderungen, denen Scrum bei der Skalierung für große Projekte gegenübersteht. Diese bestehen hauptsächlich aus der Synchronisation der Teams, dem effizienten Auflösen von Abhängigkeiten, dem Sicherstellen von regelmäßiger sowie direkter Kommunikation, der Reduktion von Komplexität sowie dem Fördern und Fordern von regelmäßigem Feedback aller Projektbeteiligten.

Die Herausforderungen unterteilen sich in die Skalierung der Scrum Artefakte, Rollen und Meetings. Für jeden der drei Bereiche werden in dieser Arbeit mehrere Lösungsmöglichkeiten vorgestellt, evaluiert und bewertet.

Neben den Scrum Artefakten, Rollen und Meetings gibt es weitere, querschnittliche Herausforderungen, welche es bei der Skalierung von Scrum zu beachten gilt. Dabei handelt es sich um die Gebiete Softwarearchitektur, Unternehmenskultur, Projekt- und Prozessmanagement sowie Unternehmensstruktur.

Die als starr angesehene Entwicklung einer Softwarearchitektur für ein großes Softwareprodukt steht der agilen Entwicklungswelt gegenüber und auch bei den unterschiedlichen Unternehmenskulturen gibt es Differenzen hinsichtlich der Kompatibilität mit Scrum und dessen Skalierung. Bezüglich der Softwarearchitektur im skalierten Umfeld wurde ein Mittelweg gefunden, der sich als „Enough Design Up Front" betitelt. Dieser stellt einen Kompromiss zwischen dem rein inkrementellen Ansatz des „No Design Up Front" und dem an das Wasserfallmodell angelehnten „Big Design Up Front" dar.

Hinsichtlich der in den Unternehmen vorgefundenen Unternehmenskulturen obliegt es den jeweiligen Vorständen, sich für eine Entwicklung der vorhandenen Unternehmenskultur zu entscheiden oder versuchen, eine gänzlich andere Unternehmenskultur anzustreben. Beide Möglichkeiten können je nach Grad und Ausprägung der vorliegenden Unternehmenskultur mehrere Jahre des Wandels in Anspruch nehmen.

Das Projekt- und Prozessmanagement muss für die Skalierung von Scrum geeignete Teamstrukturen sowie Integrationsstrategien finden, um die Produktinkremente der einzelnen Teams optimal zusammenzuführen. Die in dieser Arbeit vorgeschlagenen Möglichkeiten bieten den Verantwortlichen einige Optionen, müssen jedoch mit Bedacht gewählt und eingeführt werden. Viele dieser Lösungsmöglichkeiten fordern Veränderungen in der Unternehmensstruktur. Der Wandel der Unternehmensstruktur zur Unterstützung der Teams für eine effektive Produktentwicklung stellt somit eine weitere Herausforderung bei der Skalierung von Scrum dar.

Abstract

This paper evaluates the challenges that are faced in scaling Scrum for large-scale projects. These consist mainly of the synchronisation of the teams, the effective resolution of dependencies, the ensuring of regular and direct communication, the reduction of complexity and the support and demand of regular feedback from all stakeholders. The challenges can be divided into the scaling of Scrum artifacts, roles and meetings. For each of the three sections in this paper, several possible solutions are presented, evaluated and given a rating. In addition to the Scrum artifacts, roles and meetings, there are several other cross-sectional challenges that need to be considered in terms of scaling Scrum. These are the areas of software architecture, corporate culture, project and process management and corporate structure.

The development of a software architecture for a large-scale software product can be viewed in terms of rigid faces of an agile development world and also in terms of the different corporate cultures. There are also compatibility differences with Scrum and its scaling. Regarding the software architecture in the scaled environment, a balance has been found, which is referred to as "Enough Design Up Front". This represents a compromise between the purely incremental "No Design Up Front" approach and the style similar to the cascading "Big Design Up Front" approach.

With regard to the corporate cultures found in the companies, the question facing the respective boards of Directors is whether to opt for a development of the existing corporate culture or to instead strive to develop a completely different culture. Depending on the level and characteristics of the present corporate culture, both options can involve a change process lasting several years.

To succeed in scaling Scrum, the project and process management must find appropriate structures and integration strategies in order to optimally merge the needs of the product increments of each team. The options proposed in this thesis present business leaders with some options. However, these must be wisely chosen and introduced with care. Many of these solutions require changes in the corporate structure.

The necessary changes in the corporate structure required to support the teams in order to ensure effective product development thus represents a further challenge for the scaling of Scrum.

Inhaltsverzeichnis

I Vorwort **1**

1 Motivation **1**

2 Ziele der Arbeit **3**

3 Inhaltsangabe und Methodische Vorgehensweise **3**

4 Problemabgrenzung **4**

II Definition von Groß im Kontext der Skalierung von Scrum **5**

5 Ab wann ist ein Projekt als Groß einzustufen? **5**

6 Ab wann ist ein Team als Groß einzustufen? **5**

III Faktoren, die bei der Skalierung von Scrum beachtet werden müssen **7**

7 Scrum Artefakte **7**

7.1 Product Backlog 7

7.2 Sprint Backlog 8

7.3 Produktinkrement 8

7.4 Sprints und Sprint Ziel 9

7.5 Definition of Done 9

8 Scrum Rollen **9**

8.1 Product Owner 9

8.2 Scrum Master 10

8.3 Entwicklungsteam 11

9 Scrum Meetings **12**

9.1 Sprint Planning Meeting 1 12

9.2 Sprint Planning Meeting 2 12

9.3 Daily Scrum 12

9.4 Sprint Review Meeting 13

9.5 Sprint Retrospektive Meeting 14

IV Querschnittliche Faktoren, die bei der Skalierung von Projekten mit Scrum beachtet werden müssen **15**

10 Kommunikation **15**

10.1 Physische Distanz 16

10.2 Operative Distanz 17

10.3 Menschliche Distanz 18

10.4 Herausforderung Kommunikation 19

11 Softwarearchitektur **20**

11.1 Ziele von Softwarearchitektur 21

11.1.1 Strukturierung mit Softwarearchitektur 22

11.1.2 Alignment mit Softwarearchitektur 22

11.2 Scrum und Softwarearchitektur 22

11.3 Herausforderung Softwarearchitektur 23

12 Unternehmenskultur **24**

12.1 Zusammenarbeit 25

12.2 Kultivierung 26

12.3 Kompetenz 26

12.4 Kontrolle 27

12.5 Herausforderung Unternehmenskultur 27

13 Projekt- und Prozessmanagement **28**

13.1 Kollaborationswerkzeuge 29

13.2 Synchronisation von Teams 30

13.3 Herausforderungen an das Projekt- und Prozessmanagement . . . 30

14 Unternehmensstruktur **31**
14.1 Ziele einer Unternehmensstruktur 31
14.2 Ausprägungen von Unternehmensstrukturen 32
14.3 Herausforderung Unternehmensstruktur 34

V Lösungsmöglichkeiten für die Skalierung von Scrum **35**

15 Scrum Artefakte **35**
15.1 Product Backlog 35
15.1.1 Ein Product Backlog 35
15.1.2 Epic Backlog 37
15.1.3 Team Backlogs 38
15.2 Sprint Backlog 38
15.3 Produktinkrement 39
15.3.1 Kontinuierliche Integration 39
15.3.2 Merge Sprints 39
15.3.3 Hardening Sprints 40
15.4 Sprints und Sprint Ziel 40
15.4.1 Release-Kickoff 40
15.4.2 Asynchrone Sprints 41
15.4.3 Zeitlich abgestimmte asynchrone Sprints 41
15.5 Definition of Done 42

16 Scrum Rollen **43**
16.1 Product Owner 43
16.1.1 Kaskadierte Product Owner 43
16.1.2 Product Owner Gremium 44
16.1.3 Product Owner Gremium mit Super Product Owner . . . 44
16.2 Scrum Master 45
16.3 Entwicklungsteam 46
16.3.1 Horizontale Aufteilung 46
16.3.2 Vertikale Aufteilung 46
16.3.3 Gemischte Aufteilung 47
16.3.4 Mitglieder in mehreren Teams 48
16.3.5 Integrationsteams 49

17 Scrum Meetings **50**

17.1 Sprint Planning Meeting 1 50

17.1.1 Laufende Vorausplanung 50

17.1.2 Großraumverfahren 51

17.2 Sprint Planning Meeting 2 51

17.2.1 Sprint-Open-Space 51

17.2.2 Last responsible moment 52

17.3 Daily Scrum 53

17.3.1 Scrum of Scrums (kurz) 53

17.3.2 Scrum of Scrums (lang) 54

17.4 Sprint Review Meeting 55

17.4.1 Sprint Review Messe 55

17.4.2 Zeitlich abgestimmte Sprint Review Messe 55

17.5 Sprint Retrospektive Meeting 56

17.5.1 Herzschlag Retrospektiven 56

17.5.2 Virtuelle Retrospektiven 57

17.5.3 Meta Retrospektiven 57

VI Bewertung der vorgestellten Lösungsmöglichkeiten **58**

VII Fazit und Ausblick **62**

VIII Anhang 1 - Analyse der Vorgehensweisen bei der Skalierung von Scrum **64**

A Success Story Adobe Systems Inc. **64**

A.1 Kommunikation mit entfernten Teammitgliedern 65

A.2 Umwandlung von Produkt Backlog Items 66

A.3 Zusammenarbeit mit nicht agilen Teams 66

B Success Story Primavera **66**

B.1 Die Büroräume 67

B.2 Das Sprint Review Meeting 67

B.3 Sicherstellen von Software Qualität 68

IX Anhang 2 - Gedächtnisprotokolle 68

C Gespräch zum Thema Softwarearchitektur 68

C.1 Softwarearchitektur und die Rolle des Architekten 69
C.2 Skalierte Softwarearchitektur 69
C.3 Trennung von Architektur und Design 69
C.4 Team Aufteilung 69
C.5 Vorüberlegungen für Softwarearchitekturen 70
C.6 Erfahrungsbericht eines Projektverlaufs 70
C.7 Softwarearchitektur Qualität 71

D Gespräch zum Thema Softwarearchitektur 71

D.1 Situationsbeschreibung und Herausforderungen 71
D.2 Maßnahmen 73
D.3 Lessons Learned 74

E Gespräch zum Thema Scrum und Prozessmanagement 75

E.1 Prozessmanagement und BPM 75
E.2 Scrum im Rahmen von BPM 75
E.3 Erweiterungen von Scrum 75

F Gespräch zum Thema international verteilte Teams 76

F.1 Situationsbeschreibung 76
F.2 Herausforderung Zeitzonen 76
F.3 Herausforderung Kommunikation 76
F.4 Herausforderung Kulturelle Unterschiede 77
F.5 Lessons Learned 77

X Literaturverzeichnis 78

Abbildungsverzeichnis

1 Ausschnitt aus dem Chaos Report [Sta94] der Standish Group . . 1

2 Durchschnittlicher Produktivitätsindex über alle analysierten Projekte in Abhängigkeit zur Teamgröße nach [Put11] 11

3 Das „Virtual Distance Model" von K. Lojeski und R. Reilly . . . 16

4 Ziele, Aufgaben und Herangehensweisen von Softwarearchitektur nach [Fri10, Abb. 1] 21

5 Das „Schneider culture model" in Anlehnung an [Sch99] 24

6 Vor- und Nachteile einer (De)-zentralen Unternehmensstruktur nach [Rec11, S. 3] 33

7 Ein Product Backlog mit verschiedenen Ansichten. In Anlehnung an [Coh10, Abbildung 17.2] 36

8 Ein Epic Backlog, dass auf das allgemeine Product Backlog heruntergebrochen wurde. 37

9 Skalierung der Product Owner. In Anlehnung an [Coh10, Abbildung 17.1] 43

10 Entwickler sind Mitglied in mehreren Entwicklungsteams. In Anlehnung an [Coh10, Abbildung 17.4] 48

11 Weniger als drei gleichzeitige Aufgaben werden am effektivsten abgearbeitet [Whe10, S. 242] 49

12 Skalierung des Daily Scrums. In Anlehnung an [Coh10, Abbildung 17.5] 54

13 Zusammenfassung der vorgestellten Lösungsmöglichkeiten zu den Scrum Artefakten zur qualitativen Bewertung. 58

14 Zusammenfassung der vorgestellten Lösungsmöglichkeiten zu den Scrum Rollen zur qualitativen Bewertung. 59

15 Zusammenfassung der vorgestellten Lösungsmöglichkeiten zu den Scrum Meetings zur qualitativen Bewertung. 60

16 Exemplarische Lösungsmöglichkeit für die Skalierung von Scrum 61

17 Organigramm des Adobe Premiere Scrum-Teams bei Adobe . . . 65

18 Organisation des Softwareprojektes 74

Teil I

Vorwort

1 Motivation

Je größer das Projekt, desto höher die Wahrscheinlichkeit, dass das Projekt scheitert oder nur zu einem Teilerfolg geführt wird. Dies besagt der „Chaos Report“ (vgl. Abbildung 1), der seit 1994 jährlich von der Standish Group durchgeführt wird.

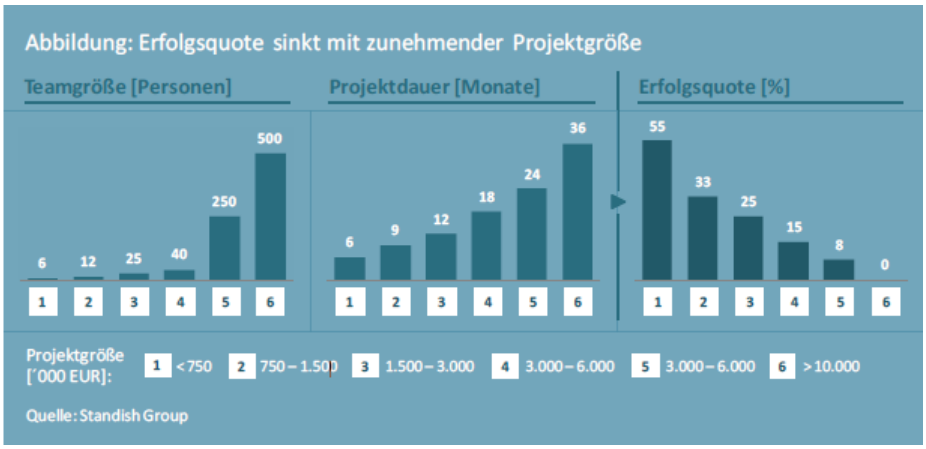

Abbildung 1: Ausschnitt aus dem Chaos Report [Sta94] der Standish Group

Die Hauptgründe hierfür liegen bei der mangelhaften Dokumentation der Anforderungen, bei der Kommunikation und sich während der Projektlaufzeit ändernden Anforderungen [Ber08, S. 9][Eck12, S. 29].

Scrum ist das erfolgreichste Framework für die Vorgehensweise bei der Softwareentwicklung [Amb11]. Eine der Stärken von Scrum ist seine Agilität. Ein Scrum-Team kann selbst nach längerem Projektverlauf besser und schneller auf Veränderungen der Anforderungen reagieren als ein Entwicklungs-Team, welches mit traditionellen (z.B. Wasserfall, Seriell, ...) Modellen arbeitet.

Je größer das Projekt, desto eher wird es mit einer klassischen Projektmanagement Methode durchgeführt. Dabei könnte Scrum gerade hier die größten Risikoträger bei großen Projekten eliminieren oder zumindest abschwächen. Verschiedene Umfragen und Studien belegen, dass agil durchgeführte Projekte signifikant häufiger zum Erfolg führen als klassisch durchgeführte Projekte [OOS08, Amb10] [Sta10, S. 15].

Nach einer von Scott Ambler 2009 durchgeführten Umfrage [Amb09a] sind die Hauptgründe gegen den Einsatz von Agilen Methoden die kulturelle Einstellung des Unternehmens, mangelndes Training der Projektverantwortlichen (aufgrund

von Ressourcen- und Zeitmangel), die mangelnde Unterstützung des Managements für die Einführung agiler Methoden sowie der generelle Widerstand gegenüber Veränderung. Die letzten beiden Hauptgründe werden von der „State of Agile Survey“ [Ver11] bestätigt und um den interessanten Fakt ergänzt, dass diese in nur 10% der kleinen Unternehmen gegen eine Einführung von agilen Methoden sprechen. Doch warum wagen gerade kleine Unternehmen den Schritt in die agile Welt eher als große? Leider geht dies bisher aus keiner Studie oder Umfrage hervor.

Gründe warum Entscheidungsträger in mittleren bis großen Firmen mit großen Projekten hier auf „bewährte Methoden“ statt auf agile wie z.B. Scrum setzen könnten die folgenden sein:

- Laut dem offiziellem Scrum Guide von Ken Schwaber und Jeff Sutherland [SS11], in dem das Scrum Framework dokumentiert wird, ist Scrum nur für Entwicklungs-Teams von drei bis neun Personen ausgelegt. Es wird erwähnt, dass häufig mehrere Teams an einem Produkt arbeiten jedoch nur angedeutet wie dies funktionieren könnte.
- Für größere Projekte und verteilte Teams werden von den Scrum Zertifizierungsstellen Scrum.org und verschiedenen Webseiten Ansätze für die Skalierung von Scrum vorgestellt. Diese sind jedoch nicht vollständig ausgeführt und nur sehr unscharf definiert.
- Für die „Skalierung von Scrum“ gibt es nur wenig deutsche Literatur, die genau dieses Thema ausführlich behandelt. Die deutschen Werke hierzu sind von Boris Gloger [Glo09] und Mike Cohn [Coh10]. Diese behandeln das Thema allerdings nur am Rande. Die beiden Bücher von Craig Larman und Bas Vodde [LV08, LV10] behandeln das Thema sehr ausführlich, allerdings auf über 1000 Seiten in Englisch und schießen damit über die Einfachheit von Scrum hinaus. Zum Vergleich: der Scrum Guide [SS11] erläutert das komplette Framework zur Entwicklung von Software in komplexen Umgebungen auf gerade mal 16 Seiten.
- Für die „Skalierung von Agilität im Unternehmen“ gibt es verschiedene Frameworks wie z.B. das „Scaled Agile Framework“ [Lef] oder die „Crystal Family“ [CRY]. Diese sind jedoch zu weit Weg von den Werten von Scrum oder nicht komplett ausgearbeitet[1].

Daher besteht die Motivation dieser Arbeit darin, ein Artefakt zu erstellen, wie Scrum im Enterprise Umfeld funktionieren könnte. So können Hemmnisse bei der großflächigen Einführung von Scrum abgebaut und die Verbreitung von agilen Methoden in großen Projekten ausgebaut werden. Es sollen die Herausforderungen, die große Projekte mit verteilten Teams mit sich bringen aufgezeigt und Vorschläge unterbreitet werden, wie diese mit Scrum gelöst werden könnten.

[1] Crystal Orange ist für Teams bis 40 Personen spezifiziert. Crystal Red, Crystal Blue und weitere Farben, die für größere Teams reserviert sind, sind bisher nicht näher spezifiziert. Stand 08/2012

2 Ziele der Arbeit

Diese Arbeit verfolgt drei Hauptziele:

1. Das Aufzeigen von Herausforderungen, die den Scrum Artefakten, Methoden und Rollen bei großen Projekten gegenüberstehen
2. Das Aufzeigen von querschnittlichen und nicht funktionalen Herausforderungen, die bei der Skalierung von Projekten mit Scrum beachtet werden müssen
3. Bereitstellen von Lösungsansätzen für die Skalierung der Scrum Artefakte, Methoden und Rollen

Das methodische Vorgehen um diese Ziele zu erreichen wird im folgenden Kapitel 3 erläutert.

3 Inhaltsangabe und Methodische Vorgehensweise

Zunächst wird eine Literaturrecherche durchgeführt, in der die wichtigsten Arbeiten zum Thema Skalierung von Scrum gesichtet werden. Mit der Literaturrecherche erfolgt die Evaluierung der bekannten Frameworks zur Skalierung von Agilität im Unternehmen sowie die Sichtung von Success Stories von Unternehmen, die Scrum bereits für sich skaliert haben. Die Analysen der Success Stories werden in Anhang 1 dokumentiert. Für das weitere Vorgehen ist es erforderlich, die Dimensionen zu erörtern, die den Begriff „Groß“ im Kontext der Skalierung von Scrum definieren. Dies geschieht in Teil zwei dieser Arbeit. Aus der Literaturrecherche und der Sichtung von Umfragen zu diesem Thema gehen die häufigsten Probleme bei der Skalierung von Scrum in Unternehmen hervor. Diese werden in Teil drei dokumentiert.

Um Lösungsansätze für die Skalierung von Scrum und den nichtfunktionalen Anforderungen zu erarbeiten, müssen zunächst die wesentlichen Herausforderungen bei der Skalierung erarbeitet werden. Dies ist der vierte Teil der Arbeit. Untermauert und ergänzt werden die Herausforderungen bei der Skalierung von Scrum durch Interviews von erfahrenen Fachexperten. Die Gedächtnisprotokolle hierzu befinden sich im Anhang 2 dieser Arbeit.

Die in Teil zwei und drei der Arbeit identifizierten Herausforderungen bei der Skalierung von Scrum werden im Teile fünf dieser Arbeit erneut aufgegriffen und Vorschläge unterbreitet, wie diese gelöst werden könnten. Dabei steht die gesamte Arbeit immer vor dem Hintergrund der Vereinbarkeit der Maßnahmen mit den Werten von Scrum, die aus dem Agilen Manifest[2] [Bec01] herrühren:

1. Individuen und Interaktionen gelten mehr als Prozesse und Tools.

[2] Das Agile Manifest definiert die Werte in der agilen Softwarteentwicklung. Sie wurde 2001 von 17 Vertretern agiler Projektmanagement Methoden und Frameworks unterzeichnet.

2. Funktionierende Programme gelten mehr als ausführliche Dokumentation.
3. Die stetige Zusammenarbeit mit dem Kunden steht über Verträgen.
4. Der Mut und die Offenheit für Änderungen steht über dem Befolgen eines festgelegten Plans.

Ergänzend hierzu gibt es die 12 Prinzipien des Agilen Wertesystems, die ebenfalls auf der Webseite zum Agilen Manifests[3] abrufbar sind.

In Teil sechs der Arbeit werden die zuvor dokumentierten Lösungsmöglichkeiten zusammengefasst und bewertet. Eine exemplarisches Framework für die Skalierung von Scrum schließt die Arbeit ab.

4 Problemabgrenzung

Die Arbeit wird lediglich im Rahmen der Skalierung von Scrum [SS11] die Grundsätze und Funktionsweise von Scrum erläutern. Ein grundsätzliches Verständnis über die Prinzipien und die Funktionsweise von Scrum wird vom Leser vorausgesetzt.

Die vollständige Erläuterung und Funktionsweise der untersuchten Frameworks wird ebenfalls den Autoren der jeweiligen Arbeiten überlassen und dem Leser nur hinreichend genau dargelegt.

Weiterhin befasst sich die Arbeit ausschließlich mit der Skalierung von Scrum gegenüber den agilen Werten und Methoden sowie der Philosophie des Lean Software Developments. Davon ausgeschlossen sind demnach agile sowie traditionelle Entwicklungsmethoden und Frameworks wie der Rational Unified Process, das V-Modell, Kanban, Extreme Programming, die Crystal Methods, Adaptive Software Development, usw.

Sämtliche in dieser Arbeit aufgeführten Maßnahmen, Techniken und Methoden sind (wie auch Scrum selbst) als Framework zu sehen. Scrum und die Lean-Philosophie verfolgen das Prinzip der *Inspektion* und *Adaption*. So sollen auch die in dieser Arbeit bereitgestellten Ideen und Möglichkeiten zur Skalierung von Scrum vom Leser *inspiziert* und der eigene Prozess bei Bedarf so *angepasst* werden, dass der Wert seines Produkts für den Kunden steigt. Auf keinen Fall sollen die hier aufgeführten Maßnahmen, Techniken und Methoden blind übernommen werden.

[3] Siehe http://agilemanifesto.org/iso/de/principles.html

Teil II

Definition von Groß im Kontext der Skalierung von Scrum

Wann macht es Sinn über die Änderung der etablierten Prozesse und die Skalierung von Scrum nachzudenken? Was sind die dafür nötigen Faktoren und Auslöser, die die Anpassung der Scrum Artefakte, Meetings und Rollen nötig machen? Es liegt auf der Hand, dass ein Projekt mit mehreren hundert Entwicklern, die auf verschiedene Ländern und Kontinente verteilt sind durchaus als Groß zu bezeichnen sind. Doch wo liegen hier die Grenzen? Die verschiedenen Dimensionen, Auslöser und Faktoren, die „Groß" in diesem Kontext ausmachen sollen in diesem Teil der Arbeit erläutert werden.

5 Ab wann ist ein Projekt als Groß einzustufen?

Die primären Faktoren, die ein Projekt als groß auszeichnen sind der *Umfang* und die *Komplexität* der Anforderungen. Aus diesen primären Faktoren lassen sich laut [Eck12] weitere Sekundärfaktoren bestimme, die von den Primärfaktoren abhängen.

Je größer der Projektumfang desto mehr Personen werden dem Projekt zugeteilt. Die *Personenanzahl* ist demnach eine weitere Dimension. Mit der Projektkomplexität steigt das *Risiko*. Auch die Anzahl an beteiligten Personen kann das Projektrisiko ansteigen lassen. Die *Dauer* des Projekts steigt ebenfalls mit den beiden Primärfaktoren und je länger ein Projekt andauert und je mehr Personen dabei beteiligt sind desto größer muss das Projektbudget sein. Die *Kosten* eines Projekts sind demnach ein weiterer Sekundärfaktor bei der Definition von Groß.

6 Ab wann ist ein Team als Groß einzustufen?

„6 plus or minus 3" ist die Definition für die Größe eines Entwicklungsteams im Scrum-Guide [SS11]. Bei dieser Teamgröße, so versprechen Ken Schwaber und Jeff Sutherland, funktioniert Scrum am besten. Ein Hauptgrund dafür ist, dass Teams dieser Größe in einem Raum arbeiten können und eine optimale Kommunikation der Teammitglieder untereinander gewährleistet ist: Sie können sich jederzeit von Angesicht zu Angesicht unterhalten und auftretende Probleme sowie das weitere Vorgehen verlustfrei (durch fehlende Kommunikationskanäle) besprechen. Fundiert wird diese Vorgabe durch die Forschungen von Doug Putnam [Put11] zur Produktivität von Teams in Abhängigkeit zur Teamgröße sowie der Miller Regel [Mil56] (näheres zu Putnam und der Miller Regel in Kapitel 8.3).

Ab einer Teamgröße von 20 Personen muss man davon ausgehen, dass die Teammitglieder nicht mehr im gleichen Raum sitzen und so automatisch häufiger auf

ein bequemeres Kommunikationsmittel (wie Telefon, E-Mail oder Instant Messaging) zurückgegriffen wird um den eigenen Platz, Raum oder gar Stockwerk nicht verlassen zu müssen. Bei mehr als 100 Personen ist die Wahrscheinlichkeit groß, dass der gewünschte Gesprächspartner nicht mehr im gleichen Gebäude sitzt. Spätestens jetzt findet die persönliche, direkte Kommunikation gar nicht mehr oder nur noch zu Pflichtterminen statt.

Ein weiterer Faktor, der die Kommunikation erschwert ist die Entfernung zwischen Personen. Thomas Allen hat bereits 1984 festgestellt [All84], dass sich die Kommunikation ab einer Distanz von 50 Meter wesentlich verschlechtert. Dabei ist es nach Thomas Allen egal ob sich die Personen in unterschiedlichen Stockwerken, Räumen, Städten oder Ländern befinden.

Wenn man diese Zahlen mit denen in verschiedenen Büros vergleicht, so wird man feststellen, dass diese relativ und ungenau sind. Fest steht aber dass es Schwellwerte gibt, deren Überschreitung unterschiedliche Auswirkungen auf das Projekt und die Projektorganisation hat.

Teil III

Faktoren, die bei der Skalierung von Scrum beachtet werden müssen

Bei großen Projekten werden neue Herausforderungen an die Scrum Artefakte, Rollen und Meetings gestellt. Viele der Artefakte, Rollen und Meetings können in einem skalierten Umfeld nicht wie im Scrum Guide [SS11] definiert eingesetzt werden sondern müssen erweitert oder in einen Rahmenprozess[4] eingebettet, verwendet werden. Die Herausforderungen, die sich dem Scrum Prozess bei der Skalierung stellen werden in diesem Teil der Arbeit erläutert. Zur besseren Übersichtlichkeit wurden diese in den Kapiteln „Scrum Artefakte", „Scrum Rollen" und „Scrum Meetings" kategorisiert.

7 Scrum Artefakte

7.1 Product Backlog

Das Product Backlog ist eine priorisierte Liste an Wünschen an das zu entwickelnde Produkt. Die Wünsche im Product Backlog können sich zur Projektlaufzeit ändern, neu hinzukommen oder aus dem Product Backlog entfernt werden. Die Verantwortung für das Product Backlog trägt ausschließlich der Product Owner.

Das Hauptziel des Product Backlogs ist die Maximierung des Wertes des Produkts. Es hilft dem Scrum Team ein gemeinsames Verständnis für das Produkt und dem Entwicklungsteam dabei, den Fokus auf die wesentlichen Aufgaben zu erhalten. Durch die iterative Arbeit am Produkt Backlog werden unwichtige Features aus dem Product Backlog entfernt, was zu einer Reduzierung von Verschwendung (vergleichbar mit „Waste" aus der Lean Production [PP03, S. 4, Table 1.1]) bei der Entwicklung führt.

Ein weiteres Ziel des Product Backlogs ist der Prozesserfolg. Das Entwicklungsteam schätzt die Aufwände für Anforderungen aus dem Product Backlog und entnimmt so viel Anforderungen aus dem Product Backlog wie es glaubt bis zum nächsten Sprint umsetzen zu können (vergleichbar mit dem Pull-Prinzip aus der Lean Production [PP03, S. 71]). Ist die Arbeitsgeschwindigkeit des Entwicklungsteams bekannt, so lässt sich hieraus eine Voraussage darüber treffen welche Features bis zum nächsten Sprint umgesetzt werden können.

Bei der Skalierung des Product Backlogs für mehrere, große oder verteilte Teams muss sichergestellt werden, dass die beiden Hauptziele „Produkterfolg" und „Prozesserfolg" des Product Backlogs stets erhalten bleiben. Es muss für den

[4] Ein Prozess, welcher den ursprünglichen Prozess oder Ablauf beinhaltet oder erweitert und ihn somit skaliert.

oder die Product Owner jederzeit einfach pflegbar sein und dem Entwicklungsteam zur gemeinsamen Einsicht mit dem Product Owner an den relevanten Meetings (insbesondere im Sprint Planning Meeting) zur Verfügung stehen. Es muss weiterhin geklärt werden, wie Product Backlog Einträge für mehrere Teams aufgeteilt werden. Eine Kennzeichnung für Product Backlog Einträge, die von mehreren Entwicklungsteams in Kooperation bearbeitet werden ist ebenfalls empfehlenswert.

Eine Stärke von Scrum ist die klare Definition und Zuweisung von Verantwortlichkeiten. Bei einem Product Backlog ist der Besitzer einer Anforderung stets der Product Owner. Lediglich zu Beginn eines Sprints, im Sprint Planning Meeting 1, wird die Verantwortlichkeit für die einzelnen Product Backlog Einträge an das Entwicklungsteam übergeben und am Ende des Sprints vom Product Owner wieder ab- bzw. zurückgenommen. Die Verantwortlichkeit über das Product Backlog und dessen Product Backlog Items muss immer klar definiert und ersichtlich sein.

7.2 Sprint Backlog

Das Sprint Backlog wird vom Entwicklungsteam erstellt und enthält die Anforderungen aus dem Product Backlog, welche das Entwicklungsteam im aktuellen Sprint umsetzen möchte. Die Anforderungen werden vom Entwicklungsteam während dem Sprint Planning Meeting 2 in kleinere Tasks unterteilt, die idealerweise jeweils an einem Tag bewältigbar (gemäß der Definition of Done, Kapitel 7.5) sind. Bei der Auswahl der Anforderungen aus dem Product Backlog für den aktuellen Sprint wählt das Entwicklungsteam stets die Anforderungen mit der höchsten Priorität aus. Die Priorität der Product Backlog Einträge wird durch den Product Owner festgelegt (siehe Kapitel 7.1 und 8.1). Des weiteren sollten alle durch das Entwicklungsteam ausgewählten Product Backlog Einträge innerhalb des aktuellen Sprints vollständig bewältigbar sein.

Davon ausgehend, dass das Sprint Backlog von einem übergeordnetem Product Backlog (ganz gleich wie dieses skaliert wurde) herrührt so ergeben sich bei der ersten Betrachtung der Thematik hier keine wesentlichen Änderungen in Form von Anzahl und Aufbau der einzelnen Sprint Backlogs.

Beim Einsatz von verteilten interdisziplinären Entwicklungsteams kann es vorkommen, dass Sprint Backlog Einträge eines Teams Abhängigkeiten zu Funktionalitäten eines anderen Teams besitzt. Hier muss eine Möglichkeit gefunden werden, mit dem die Zeit in der das Team durch diesen Umstand blockiert ist, minimiert wird. Näheres zu den Herausforderungen bei der Synchronisation von Teams befindet sich in Kapitel 13.2.

7.3 Produktinkrement

Ein Scrum Entwicklungsteam sollte nach jedem Sprint in der Lage sein ein vorzeigbares und potentiell auslieferbares Produkt vorweisen zu können. Bei einem Team ist das Produktinkrement die Summe aller fertiggestellten Produkt Backlog Einträge des aktuellen Sprints sowie der vorangegangenen Sprints [SS11, S. 16]. Bei mehreren Teams muss für die Definition eines „Produktinkrements"

etwas weiter ausgeholt werden. Nachdem die verschiedenen Teams an einer Software arbeiten muss ein Zusammenführen der einzelnen Inkremente erfolgen. Folgende Fragen müssen geklärt werden um Probleme beim Zusammenführen von Produkt Inkrementen zu minimieren:

- *Wann* erfolgt das Zusammenführen der Produktinkremente?
- *Wie* erfolgt das Zusammenführen der Produktinkremente?
- *Wer* ist für das Zusammenführen der Produktinkremente verantwortlich?

Für jeden der drei Punkte gibt es verschiedene Ansätze, die je nach Projektgröße und Projektumfeld zu evaluieren sind.

7.4 Sprints und Sprint Ziel

Ein gemeinsames Sprint Ziel kann es dem Team erleichtern, die zur Erreichung dieses Ziels nötigen Product Backlog Einträge aus dem Product Backlog zu extrahieren und in das Sprint Backlog zu übernehmen. Es gibt die Richtung vor und eine gemeinsame Vision für das Team für den aktuellen Sprint. Verwenden mehrere Teams nun unterschiedliche Sprint Ziele so kann es bei der Auswahl von Product Backlog Einträgen vorkommen, dass einzelne Aufgaben Abhängigkeiten zu anderen Teams besitzen die jedoch ein anderes Sprint Ziel verfolgen und sich erst zu einem späteren Zeitpunkt der entsprechenden Aufgabe widmen. Um diese Abhängigkeiten zu minimieren ist eine Synchronisation, Abstimmung oder Aufteilung der Sprints und deren Ziele erforderlich.

7.5 Definition of Done

Die Definition of Done eines Scrum Teams enthält ein gemeinsames Verständnis davon, wann die Arbeit an einem Sprint Backlog Item vollständig abgeschlossen ist. Das Entwicklungsteam selbst ist für die Erstellung seiner Definition of Done verantwortlich. Diese Spezifikation aus dem Scrum Guide führt bei mehreren Teams zwangsläufig dazu, dass unterschiedliche Definition of Done's entstehen. Dies führt zu einer unterschiedlichen Qualität der fertigen Produktinkremente und kann Probleme beim Zusammenführen der Inkremente verursachen. Um ein teamübergreifendes Verständnis eines fertigen Produktinkrements zu erhalten ist eine Synchronisation, Aufteilung oder eine Erweiterung der Definition of Done erforderlich.

8 Scrum Rollen

8.1 Product Owner

Der Product Owner ist für die Maximierung des Geschäftswertes des Produkts verantwortlich. Er tritt als Einzelperson auf oder repräsentiert ein Gremium, dass für die Anforderungen an das Produkt verantwortlich ist [SS11, S. 6]. Er

pflegt das Product Backlog und ist für die Priorisierung der Anforderungen in diesem verantwortlich. Das Entwicklungsteam entnimmt dem Product Backlog Anforderungen, um sie in einem Sprint umzusetzen - somit ist der Product Owner die einzige Person, die dem Entwicklungsteam über das Product Backlog Arbeit zuteilen darf. Bei Fragen zum Produkt aus dem Entwicklungsteam steht der Product Owner dem Entwicklungsteam zur Seite.

Ist das zu entwickelnde Produkt größer als dass es mit einem Scrum Team in absehbarer Zeit zu bewältigen ist steigt auch der Aufwand, der für die Pflege des Product Backlogs anfällt. Noch problematischer wird es, wenn unterschiedliche Produkte gemeinsame Schnittstellen haben oder der Einfluss zu einem Produkt aus mehreren Abteilungen kommt. Hier muss eine Möglichkeit gefunden werden, dass der Product Owner eines Entwicklungsteams weiterhin seine Aufgaben wahrnehmen und zusätzlich eine Koordination der anfallenden Anforderungen erfolgen kann. Zur Koordination der anfallenden Anforderungen gehören:

- Priorisieren der Anforderungen aller Abteilungen
- Aufteilung der Anforderungen an die zuständigen Scrum Teams
- Klare Zuweisung von Besitzern zu einzelnen Product Backlog Einträgen

Hier muss jeweils eine Lösung gefunden werden, die besagt wer für diese Aufgaben verantwortlich ist und wie die Punkte adressiert werden. Weiterhin muss eine Möglichkeit gefunden werden, wie ein effektiver Informationsaustausch stattfinden kann, wenn es zu Fragen zu einer funktionsübergreifenden Komponente eines Produkts kommt.

8.2 Scrum Master

Der Scrum Master unterstützt den Product Owner, das Entwicklungsteam sowie die Organisation als dienende Führungskraft [SS11, S. 7]. Er sorgt dafür, dass die Scrum-Prozesse eingehalten werden, etabliert Techniken zur effektiven Verwaltung der Scrum Artefakte, strebt einen maximalen Geschäftswert des Produktes an und beseitigt Hindernisse die diesen Zielen im Wege stehen. Die Aufgabengebiete des Scrum Masters sind vielseitig und können bereits bei einem Scrum Team ein ganztags Job sein. Es ist daher empfehlenswert einen dedizierten Scrum Master pro Scrum Team einzusetzen. Bei mehreren Scrum Teams mit mehreren Scrum Mastern sollte evaluiert werden, wie sich die Scrum Master untereinander koordinieren können.

Geht ein Projekt über ein Team und einen Standort hinaus so werden nicht nur neue Prozesse benötigt sondern auch neue Hindernisse auftreten, die mehrere oder alle Scrum Teams betreffen. Hier muss eine Möglichkeit gefunden werden die Rolle des Scrum Masters so zu erweitern oder aufzuteilen, so dass sowohl die Problematiken bei vielen und verteilten Teams adressiert werden als auch der Scrum Master eines Scrum Teams seine Aufgaben für sein Team weiterhin voll erfüllen kann.

Die Erweiterung von Scrum zur Arbeit mit großen und verteilten Teams wird neue Artefakte, Rollen und Meetings hervorbringen. Bei dieser Skalierung muss stets beachtet werden, wer für die Einhaltung der neuen Prozesse verantwortlich ist.

8.3 Entwicklungsteam

Das Entwicklungsteam ist für die Umsetzung der Anforderungen in ein Produkt Inkrement verantwortlich. Nach der Präsentation des Product Backlogs und des Sprint Ziels durch den Product Owner entnimmt das Entwicklungsteam so viele Anforderungen aus dem Product Backlog, wie es im Sprint umsetzen kann. Das Entwicklungsteam arbeitet selbst organisiert und muss alle Fähigkeiten besitzen um alle Anforderungen aus dem Product Backlog in ein Produkt Inkrement umzusetzen. Nach der „Miller Regel" [Mil56, S. 81-97] (der auch der Scrum Guide [SS11, S. 6] folgt) sollte ein Entwicklungsteam nie mehr als neun Entwickler umfassen. Untermauert wird die Miller Regel durch eine von Doug Putnam durchgeführte Studie [Put11] zur Produktivität von Teams in Abhängigkeit zur Teamgröße. Diese zeigt ebenfalls, dass Teams zwischen drei und neun (7 +/- 2) Personen am Produktivsten sind.

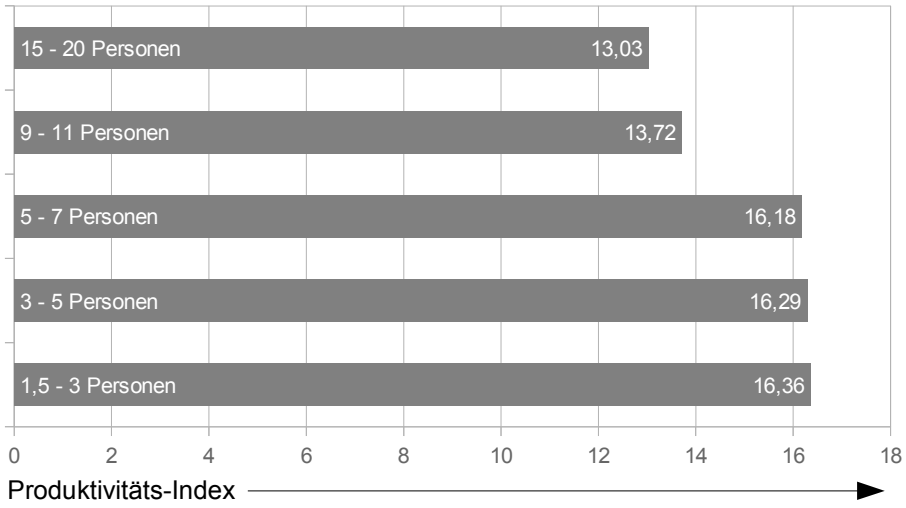

Abbildung 2: Durchschnittlicher Produktivitätsindex über alle analysierten Projekte in Abhängigkeit zur Teamgröße nach [Put11]

Die Abbildung zeigt, dass bei Teams mit 1,5 bis sieben Mitarbeitern die Produktivität am höchsten ist und auch die geringsten Abweichungen bezüglich des Produktivitäts-Index aufweist. Ab einer Mietarbeiterzahl von neun Personen pro Team sinkt die Produktivität. Eine Skalierung kann demnach nur über die Anzahl an Scrum Teams erfolgen.

Unter Berücksichtigung der genannten Merkmale eines Scrum Entwicklungsteams muss bei einer Skalierung geklärt werden wie die Entwicklungsteams aufgeteilt werden um möglichst wenig Abhängigkeit untereinander und einem maximalen Zusammenhalt der Teams (Prinzip der *„high cohesion and low coupling"* [SI08, S. 163] von Ken Schwaber) zueinander zu erhalten. Es muss weiterhin evaluiert werden wie sich die unterschiedlichen Entwicklungsteams am besten koordinieren sowie wie querschnittliche Aufgaben bei der Entwicklung gelöst werden können.

9 Scrum Meetings

9.1 Sprint Planning Meeting 1

Im Sprint Planning Meeting 1 stellt der Product Owner des Teams dem Entwicklungsteam das aktuelle Product Backlog vor. Er erklärt dem Entwicklungsteam die Einträge aus dem Product Backlog mit der höchsten Priorität. Das Entwicklungsteam stellt so lange Fragen zu den vorgestellten Product Backlog Einträgen bis diese vollständig verstanden wurden. Gemeinsam wird ein Sprint Ziel erarbeitet. Anschließend entnimmt das Entwicklungsteam so viele Anforderungen aus dem Product Backlog wie ihrer Meinung nach in den folgenden Sprint passen. Die hierfür nötige Schätzung der Aufwände erfolgt durch das gesamte Entwicklungsteam im Gruppenkonsens.

Nach den Vorgaben des Scrum Guides erfolgt das Sprint Planning Meeting 1 innerhalb eines Scrum Teams. Anwesend sind demnach ausschließlich der Product Owner des Teams, der Scrum Master des Teams (optional) sowie das Entwicklungsteam selbst. Enthält das Product Backlog jedoch Einträge, die mehrere Teams betreffen und/oder eine enge Kooperation mit einem oder mehreren Entwicklungsteams erfordern, so muss eine Lösung gefunden werden wie die entsprechenden Product Backlog Einträge mit allen beteiligten Teams diskutiert werden können. Nur durch den Austausch mit allen beteiligten Entwicklern kann ein gemeinsames Verständnis über die Anforderung sowie deren Umfang erarbeitet werden. Gibt es weiterhin Product Backlog Einträge, welche einem teamfremden Product Owner gehören, so muss geklärt werden, wie und wann die Übergabe der entsprechenden Product Backlog Einträge erfolgt.

9.2 Sprint Planning Meeting 2

Im Sprint Planning Meeting 2 bespricht das Entwicklungsteam, wie es die dem Product Backlog entnommenen Einträge in ein Produkt Inkrement umsetzen wird. Hierzu unterteilt das Entwicklungsteam die Anforderungen in kleinere Aufgaben mit einem Aufwand von maximal einem idealen[5] Tag. Im Sprint Planning Meeting ist der Product Owner des Teams anwesend und steht für Fragen vom Entwicklungsteam zur Verfügung. Das so entstandene Sprint Backlog ändert sich während des Sprints nicht.

Wie das Sprint Planning Meeting 1, findet das Sprint Planning Meeting 2 in Projekten mit nur einem Entwicklungsteam gekapselt innerhalb dieses Teams statt. Gibt es teamübergreifende Anforderungen, so muss auch hier ein Weg gefunden werden, wie die Teams miteinander kooperieren können, um die Anforderungen in kleine Tasks mit klaren Zuteilungen, auf die Entwicklungsteams (nicht die einzelnen Entwickler) zu erstellen.

9.3 Daily Scrum

Das Daily Scrum ist ein kurzes, auf 15 Minuten begrenztes täglich stattfindendes Ereignis. Die Teilnehmer sind das gesamte Entwicklungsteam sowie der Scrum

[5] Ein idealer Tag ist die effektive Arbeitszeit, die an einem Arbeitstag übrig bleibt wenn man Meetings, Unterbrechungen, Kaffeepausen usw. von der gesamten Arbeitszeit abzieht.

Master eines Teams. Die Entwickler berichten sich aktiv gegenseitig was sie seit dem letzten Arbeitstag getan haben, was sie planen bis zum nächsten Arbeitstag umzusetzen und ob sie etwas in Ihrer Arbeit behindert. Es dient somit der Synchronisation des Entwicklungsteams [SS11, S. 11].

Der Scrum Master unterstützt das Entwicklungsteam, indem er die Entwickler bei der Lösung von Problemen (Impediments) unterstützt, die sich nicht durch die Hilfe anderer Entwickler beheben können. Gibt es mehrere Entwicklungsteams, so können Probleme auftreten, die nicht vom Scrum Master eines Teams ausschließlich lokal gelöst werden können. Hier ist eine Koordination der Scrum Master erforderlich, so dass zentrale Probleme gemeinsam adressiert werden können und lokale Probleme von verschiedenen Entwicklungsteam nicht mehrmals gelöst werden müssen.

Das Daily Scrum verbessert zudem die Kommunikation, fördert schnelle Entscheidungsprozesse und verbessert das Projektwissen im Entwicklungsteam. Dadurch ist es ein Schlüssel-Meeting für die Inspektion und Adaption [SS11, S. 12]. Bei mehreren Entwicklungsteam mit Zeitlich und/oder örtlich versetzten Daily Scrums sind diese Vorteile sowie eine Synchronisation der Entwickler nicht mehr gewährleistet. Hier muss ebenfalls eine Möglichkeit gefunden werden, *wie* das gegenseitige Reporting Team übergreifend stattfinden kann.

9.4 Sprint Review Meeting

Das Sprint Review Meeting findet unmittelbar am Ende jedes Sprints statt. Es dient dem Entwicklungsteam dazu, ihr Produkt Inkrement dem Product Owner und allen interessierten (z.B. Stakeholder) vorzustellen. Auf Basis des aktuellen Produkts wird das Product Backlog angepasst und das weitere Vorgehen diskutiert. Das Sprint Review Meeting dient hauptsächlich dem Zweck Feedback hervorzubringen und die Zusammenarbeit zu fördern. Ein weiterer Punkt im Sprint Review Meeting ist die Abnahme der umgesetzten Anforderungen des Entwicklungsteams durch den Product Owner. Damit wird die Verantwortlichkeit für die Anforderungen wieder auf den Product Owner übertragen und das Entwicklungsteam entlastet.

Bei mehreren Teams steuert jedes Team sein eigenes Inkrement zu einem neuen Produktinkrement bei. Geht man davon aus, dass die einzelnen Produktinkremente zu einem späteren, separatem Zeitpunkt zusammengeführt werden, so kann das vorgestellte Inkrement eines Scrum Teams in ihrem Sprint Review Meeting lediglich ein Teilprodukt darstellen. Im zweiten Fall, wenn das Zusammenführen des Produkts vor dem Sprint Review Meeting stattgefunden hat, stellt jedes Team in ihrem Sprint Review Meeting ihr Inkrement anhand des gleichen Gesamtprodukts vor. In beiden Fällen können beim Sprint Review Meeting folgende Probleme auftreten:

- Das eingeholte Feedback betrifft einen Teil des Produkts, welches ein anderes Team entwickelt hat: In diesem Fall muss ein Weg gefunden werden wie das Feedback mit dem entsprechendem Team diskutiert werden kann.

- Das eingeholte Feedback betrifft das Gesamtprodukt: Wenn sich das Feedback nicht auf eine einzelne Teilkomponente bezieht sondern auf mehrere oder das Gesamtprodukt so muss geklärt werden *wer* das Feedback entgegennimmt (Verantwortlichkeit) und *wie* es zum entsprechendem Scrum Team zur Diskussion gelangt.
- Das eingeholte Feedback betrifft eine Komponente, die von mehreren Teams bearbeitet wird: Hier muss ein Weg gefunden werden wie das Feedback mit den entsprechenden Teams diskutiert werden kann.

Regelmäßig und häufig Feedback einzuholen sowie die Zusammenarbeit zu fördern sind Kernkomponenten von Scrum und dem Sprint Review Meeting. Bei einer Skalierung von Scrum über mehrere und verteilte Teams dürfen diese Kernkomponenten nicht verloren gehen. Somit muss ein Weg gefunden werden, welcher effektiv das Feedback aller Interessenten aus allen Teams mit einbeziehen kann.

9.5 Sprint Retrospektive Meeting

Das Sprint Retrospektive Meeting findet unmittelbar nach dem Sprint Review Meeting statt. Es ist eine Möglichkeit für das Team sich selbst zu reflektieren und einen Plan für Verbesserungen aufzustellen, die im folgenden Sprint umgesetzt werden sollen. Das Sprint Review Meeting legt seinen Fokus auf die Scrum Prinzipien, Inspektion und Adaption. Somit bespricht das gesamte Team, was im letzten Sprint gut gelaufen ist, was sie beibehalten wollen und was verbessert werden könnte. Das Team stellt anhand diesen Erkenntnissen einen Plan auf, wie sie die Verbesserungen im nächsten Sprint umsetzen möchten. Gab es bereits ein Sprint Retrospektive Meeting so wird der Status der Verbesserungen geprüft, die beim letzten Sprint Retrospektive Meeting besprochen wurden.

Lokale Probleme oder Verbesserungen, welche bei der Inspektion identifiziert werden, können entweder durch das Entwicklungsteam selbst oder mit Hilfe des Product Owners und des Scrum Masters des jeweiligen Teams adressiert werden. Gibt es Probleme oder Verbesserungen, die mehrere Teams betreffen so muss ein Weg gefunden werden, wie diese mit den entsprechenden Teams besprochen und umgesetzt werden können. Es gilt abermals zu klären *wer* für die Weitergabe der Adaptionen verantwortlich ist und in welcher Form (*wie*) die Weitergabe erfolgt.

Teil IV

Querschnittliche Faktoren, die bei der Skalierung von Projekten mit Scrum beachtet werden müssen

Bei der Skalierung von Scrum für große Projekte mit vielen und/oder verteilten Teams ergeben sich weitere Herausforderungen, die über die bekannten Scrum Prozesse hinaus gehen. Diese ergänzenden Herausforderungen werden in dieser Arbeit als „querschnittliche Faktoren" bezeichnet. Dies sind exemplarisch die Verteilung der Verantwortlichkeiten, die Aufteilung der Teams hinsichtlich, eine konsistente Architektur der Software, der Aufbau eines Basissystems, das einheitliche Design der Benutzeroberfläche oder das Team übergreifende Qualitätsmanagement. In diesem Teil der Arbeit werden die Anforderungen, welche bei der Skalierung der Scrum Artefakte, Rollen und Meetings sowie weiteren querschnittlichen Tätigkeiten beachtet werden müssen, erläutert.

10 Kommunikation

Scrum fordert und fördert gute Kommunikation zwischen allen Projektbeteiligten. Die Wünsche an das Produkt vom Product Owner werden in den Sprint Planning Meetings sowie den Sprint Review Meetings besprochen (je nach gewählter Sprintlänge jeweils zwei bis vier mal im Monat) und die Mitglieder des Entwicklungsteams tauschen sich sogar täglich aus. Das „Systems and Software Consortium" (SSCI) widmete sich der Frage was die Hauptthemen bei der verteilten Entwicklung [ND05] sind. Auf Rang drei der „Top issues in distributed development" steht die Kommunikation. Auch Jutta Eckstein (in [Eck09] und [Eck12, S. 2]) sowie K. Lojeski und R. Reilly (in [LR08]) beschreiben in Ihren Büchern wie wichtig gute Kommunikation insbesondere bei großen und verteilten Projekten ist. In „Uniting the Virtual Workforce" [LR08] betrachten K. Lojeski und R. Reilly die Hemmnisse bei der Kommunikation mit anderen Projektmitgliedern aus verschiedenen Blickwinkeln und unterteilen diese in drei Kategorien. Die Summe der Hemmnisse wird als „Virtual Distance" (Virtuelle Distanz) bezeichnet. Je größer die virtuelle Distanz zwischen zwei Personen ist umso schlechter ist die Qualität der Kommunikation und desto größer ist das Risiko, dass ein Projekt aufgrund dessen scheitert. Die drei Kategorien, in die die Autoren K. Lojeski und R. Reilly die Virtuelle Distanz unterteilen sind im folgenden Schaubild um das Zentrum angeordnet.

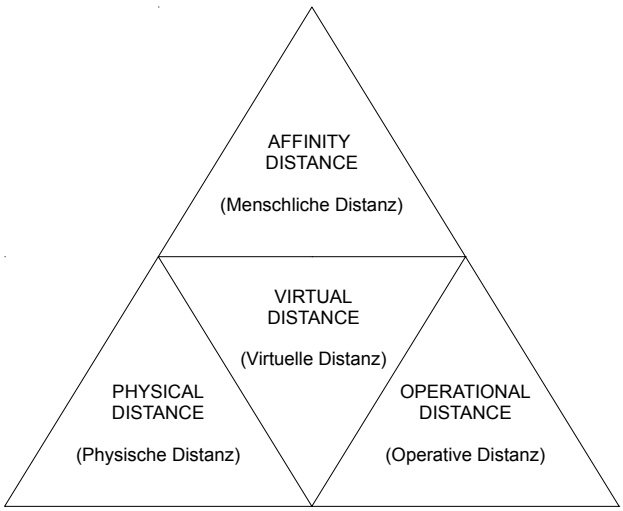

Abbildung 3: Das „Virtual Distance Model" von K. Lojeski und R. Reilly

Die Kommunikation innerhalb eines einzelnen Scrum Entwicklungsteams ist (nach den Teambildungsphasen) als sehr gut einzustufen, da die Teams üblicherweise mit einer Teamgröße von drei bis neun Personen im gleichen Raum arbeiten, häufig der gleichen Nationalität angehören und auch sonst auf vielen Ebenen direkt und effektiv miteinander kommunizieren können. Bei großen und verteilten Teams sieht die Situation anders aus. Hier wird es jetzt wichtig die Kommunikation und die virtuelle Distanz, die zwischen den einzelnen Teammitglieder herrscht besser zu verstehen und die gewohnte Qualität der Kommunikation hinreichend genau sicherzustellen. Hierbei hilft das *Virtual Distance Model*. Im folgenden werden die einzelnen Bestandteile des Modells beschrieben.

10.1 Physische Distanz

Die Physische Distanz (*Physical Distance*) ist das offensichtlichste Hemmnis bei der Arbeit mit verteilten Teams. Sie besteht aus der räumlichen Distanz (*Geographic Distance*), da-einhergehend mit der zeitlichen Distanz (*Temporal Distance*) sowie der organisatorischen Distanz (*Organisational Distance*).

Arbeiten Teams an verschiedenen Orten so verhalten sich die Teammitglieder untereinander anders gegenüber ihrem Gesprächspartnern, denen sie direkt gegenüber sitzen. Dies belegt das Buch „Why distance matters: Effects on cooperation, persuasion and deception" [BM02] in verschiedenen Studien. Besonders negativ beeinflusst in der Kommunikation mit entfernten Teammitgliedern sind die ethischen Entscheidungen sowie die emotionale Verbundenheit mit dem Gegenüber. Dies führt in unseren Entscheidungen zu weniger effektiven und weniger rationellen Handlungen. Um die räumliche Distanz zwischen den Teammitgliedern zu reduzieren gibt es verschiedene Ansätze. Der effektivste ist das temporäre physische Treffen der Teammitglieder während der Teambildungsphasen (siehe auch [LR08, S. 30]). Besonders während der Orientierungs- und Konfrontationsphase (nach [Tuc65, S. 63, S. 384-399]) kann dies zu einer nachhaltigen

Verbesserung der Kommunikation führen. Es empfiehlt sich zudem die Teammitglieder auch während der Projektphase an einen Tisch zu bringen. Dies kann, je nach Projekt, zu jedem Sprint-Wechsel oder nur zu Sprint-Wechseln z.B. bei Major Releases erfolgen. So kann die emotionale Verbundenheit zwischen den Teammitglieder wieder aufgefrischt werden. Technisch kann die räumliche Distanz durch das hinzufügen von möglichst reichhaltigen Kommunikationskanälen reduziert werden.

Arbeiten die einzelnen Teammitglieder in Ländern mit unterschiedlichen Zeitzonen so können Sie diesem Hemmnis durch eine Sensibilisierung des Teams auf die Problematik sowie durch zeitlich geschicktes platzieren der Scrum Meetings entgegenwirken. Die Sensibilisierung des Teams bewirkt, dass sich die Teammitglieder jederzeit bewusst sind, dass eine Antwort auf Ihre E-Mail mitunter mindestens einen Tag dauern kann. Zeigen Sie des weiteren durch geschicktes, und abwechselndes platzieren der Termine, dass jeder im Team gleichermaßen involviert ist. So wird vermieden, dass einzelne Teammitglieder immer zu Bürounüblichen Zeiten präsent sein müssen während andere immer bequem zu den regulären Geschäftszeiten operieren. Praktikabel wäre hier z.B. die Verlegung der Uhrzeit des Daily Scrum zu jedem Sprint-Wechsel (auf keinen Fall häufiger).

Die dritte physische Distanz, die organisatorische Distanz, beschreibt das menschliche verhalten einzelner Personengruppen aufgrund verschiedener Kriterien abzugrenzen. Obwohl diese für die gleiche Firma oder das gleiche Projekt arbeiten so werden diese in unterschiedliche Gruppen unterteilt. Dies merkt man häufig daran, dass von einem „denen" statt von einem „uns" oder „wir" gesprochen wird. Provoziert wird dieses Verhalten durch unterschiedliche Standorte, nachträglich hinzugekommene Kollegen im Entwicklungsteam oder wenn das Entwicklungsteam zum Teil aus Mitarbeitern aus unterschiedliche Abteilungen besteht.

10.2 Operative Distanz

Die operative Distanz (*Operational Distance*) kann gleichermaßen in kleinen und großen Projekten auftreten. Sie besteht aus der kommunikativen Distanz (*Communications Distance*), dem bekannten *Multitasking*, der unterschiedlichen Handlungsbereitschaft (*Readiness Distance*) sowie der asymmetrischen Verteilung von Teammitgliedern *(Distribution Asymmetry*).

Wissen wir nicht in welcher Stimmung oder Umfeld sich eine Person befindet so fehlen wichtige Kommunikationskanäle, die für das vollständige Verständnis einer Nachricht erforderlich sind. Auch ein fehlender gemeinsamer Kontext kann die Qualität einer Nachricht beeinträchtigen. Diese kommunikative Distanz kann dazu führen, dass wir unser Gegenüber missinterpretieren und so Missverständnisse entstehen, die sich negativ auf den Projektfortschritt auswirken. Am geringsten ist die kommunikative Distanz, wenn man sich gegenüber steht. Versuchen Sie dies also so oft wie möglich. Da dies jedoch in großen, verteilten Projekten nur selten möglich ist empfiehlt es sich den gemeinsamen Kontext, die Stimmung oder das Umfeld entsprechend zu vermitteln. Ein kurzer einleitender Satz in einer E-Mail über das Klima im Büro oder der persönliche Hintergrund zu einer Frage kann helfen. Falls nicht empfiehlt sich der Wechsel auf das nächst reichhaltigere Kommunikationsmittel.

Länger anhaltende Scrum „Impediment” werden im Virtual Distance Model als „Readiness Distance” bezeichnet. Im Kontext der Kommunikation reicht es hierbei noch etwas weiter: Es führt dazu, dass Teammitglieder mit ihren Gedanken abdriften und sich sogar vom Projekt distanzieren, sollte ein Problem länger anhalten. Dies ist besonders kritisch, wenn sich Teammitglieder in einem Multitasking Umfeld bewegen. Ein **dedizierter** Scrum Master, der sich so schnell wie möglich um Probleme kümmern kann, die das Entwicklungsteam nicht eigenständig lösen kann ist demnach besonders wichtig. Ein weiteres Hindernis in diesem Kontext kann die unterschiedliche Erwartungshaltung bei der Kommunikation sein: Darf eine E-Mail mehr als zwei Tage unbeantwortet bleiben? Wann sollte lieber auf Instant Messaging und wann auf das Telefon zurückgegriffen werden? Klären Sie dies möglichst zu beginn der Teambildungsphase mit allen beteiligten ab, so dass hier ein gleiches Verständnis über adäquate Reaktionszeiten und Kommunikation herrscht.

10.3 Menschliche Distanz

Die Menschliche Distanz tritt besonders zutage wenn Teammitglieder aus unterschiedlichen Abstammungen und Sozialschichten aufeinander treffen. Dies ist natürlich häufiger in großen und verteilten Projekten anzutreffen als in kleinen und lokalen. Ein besonderes Augenmerk gehört hierbei den folgenden Unterkategorien der *Affinity Distance* an: Die kulturelle Distanz (*Cultural Distance*), die soziale Distanz (*Social Distance*), den Grad der kognitiven Verbindung (*Relationship Distance*) sowie den Grad der Abhängigkeit zueinander (*Interdependence Distance*).

Die kulturelle Distanz zeichnet sich durch die Werte ab, die unser Denken und Handeln bestimmen. Sie werden von K. Lojeski und R. Reilly [LR08, Abbildung 2.5, S. 43] in folgende Werte unterteilt (aufsteigende Reihenfolge):

- Moralische Werte: Persönlicher Satz von absoluten Werten, die zwischen „richtig“ und „falsch“ unterscheiden. Wenn diese verletzt werden neigen wir zu drastischen Maßnahmen.
- Arbeitswerte: Die interne Straßenkarte, die wir als Ableitung aus unseren moralischen Werten erstellen und nach denen wir unsere Arbeitsgewohnheiten ausrichten.
- Persönliche Werte: Basierend auf internationalen Gruppenwerten kombiniert mit unserem persönlichem Weltbild
- Kulturelle Werte: Gruppenwerte, basieren auf der Teilhabe in bestimmten Gruppen oder Vereinen sowohl Privat als auch Geschäftlich.

Die Entscheidungen, die die Projektbeteiligten treffen sind also Stark abhängig von ihrer Herkunft und moralischen Werten. Persönlich gehen wir immer davon aus, dass unser Gegenüber in etwa so denkt wie wir und unser Anliegen sofort versteht. Dies ist natürlich nur in den seltensten Fällen und wenn überhaupt nur bei eingespielten Teams der Fall. Je größer die kulturelle Distanz desto ausführlicher und eindeutiger müssen wir unser anliegen kommunizieren, damit

es nicht zu Missverständnissen kommt. Dies kann mitunter sehr anstrengend sein und die Effektivität des Teams negativ beeinflussen. In [ND05] befindet sich die „Kulturellen Unterschiede" auf Rang vier der „Top Issues in distributed Management".

Je wichtiger in einer Firma der Status, Rang oder Position einer Person ist desto größer kann die soziale Distanz zwischen den Mitarbeitern werden. Dabei ist es nicht wichtig, dass eine beliebige Person eine bestimmte Position oder einen bestimmten Rang in einer Firma einnimmt sondern wie stark dieser betont wird und wie wichtig dem Unternehmen ein striktes Vorgehen gemäß der Firmenhierarchie ist. Sind derartige „Machtkämpfe" im Spiel, bei denen womöglich schlechte Entscheidungen von höherrangigen Personen getroffen werden, so stößt dies bei den restlichen Teammitgliedern bitter auf. Achten Sie deshalb wie gewohnt darauf, dass ein Scrum-Team interdisziplinär aufgestellt ist und innerhalb des Team Rangordnungen keine Rolle spielen.

Der Grad der kognitiven Verbindung beschreibt wie stark wir mit einer Person in Verbindung stehen. Haben wir schon viele Projekte gemeinsam durchgeführt und kennen wir viele Personen gemeinsam so ist die *Relationship Distance* zwischen uns niedrig. Arbeiten wir zum ersten mal zusammen so kann der Grad der kognitiven Verbindung zwischen uns ebenfalls initial einen niedrigen Wert haben wenn wir viele Personen gemeinsam kennen. Die sozialen Netzwerke im Internet haben dies schon länger erkannt. Sie nutzen dies zu ihrem Vorteil, indem Sie uns Kontaktvorschläge zu Personen anzeigen, die über mehrere Wege über zwei bis drei Ecken mit uns in Verbindung stehen. Bedenken Sie dies, wenn Sie das nächste mal ein neues Scrum-Team zusammenstellen.

Bei vielen großen Banken ist der Begriff des „Interdependence Risk" als Grad der Abhängigkeit von Personen zueinander bekannt. Er beschreibt das Risiko, wenn zwischen Dienstleistern und Auftraggeber nur eine geringe Abhängigkeit zueinander herrscht und keine gemeinsame Vision verfolgt wird. Operieren die beteiligten Personen innerhalb eines Projekts zu losgelöst zueinander so steigt dieses Risiko an. Sie können dieses Risiko niedrig halten, indem Sie die Scrum Meetings in gewohnten Zyklen einhalten und das gemeinsame Ziel zu den Sprint Planning Meetings etappenweise und wiederholt klar vermitteln.

Bei allen Unterpunkten der Menschlichen Distanz ist es ratsam mit dem Team offen darüber zu sprechen. So hilft auch hier eine der stärken von Scrum, die Transparenz. Hat das Team die Problematik hinter der interkulturellen Kommunikation im Hinterkopf, so kann es besser mit dieser umgehen (siehe Interview im Anhang F).

10.4 Herausforderung Kommunikation

Kommunikation stellt demnach in vielerlei Hinsicht eine Herausforderung bei Projekten mit vielen oder verteilten Teams dar. Die unterschiedlichen Ausprägungen von Hemmnissen bei der Kommunikation wurden in den vorherigen Kapiteln erläutert. Bei der Skalierung der Scrum Artefakte, Rollen und Meetings muss stets sichergestellt werden, dass die Informationsflüsse im Großen wie bisher im Kleinen aufrecht erhalten werden. Es müssen Wege der Kommunikation gefunden werden, die ...

- ... das Projektwissen der Mitglieder der Entwicklungsteams stets umfassend und aktuell halten
- ... die Synchronisation der Entwicklungsteams gewährleisten
- ... die Koordination und Priorisierung der Wünsche und Anforderungen an das Produkt gewährleistet
- ... Hindernisse bei der Produktentwicklung möglichst effektiv adressiert und löst
- ... die Produktvision klar an alle Projektbeteiligten kommuniziert

Eine Erweiterung des Scrum Frameworks muss mindestens diese Punkte adressieren, um den gewohnten Umfang und die Qualität der Kommunikation wie bei kleinen Scrum Projekten (mit nur einem Scrum Team) gerecht zu werden.

11 Softwarearchitektur

Ein wesentlicher Bestandteil der traditionellen Softwareentwicklung ist die vorausschauende Planung. Die Vorgehensweise, eine Softwarearchitektur im Rahmen einer Entwurfsphase konzeptionell vorab zu entwickeln, wird diesem Ansatz gleichgesetzt. In diesem Zusammenhang stellt sich die Frage, ob ein planender Ansatz wie Softwarearchitektur mit der iterativen Vorgehensweise von Scrum vereinbar ist.

Es gibt viele Definitionen für Architektur in der Softwareentwicklung. Allgemein beschreibt Softwarearchitektur die zentralen Komponenten und Systeme einer Softwarelösung sowie deren Abhängigkeiten untereinander (nach [Fow03]). Einhergehend mit dem Begriff Softwarearchitektur ist auch stets von dem Begriff „Softwaredesign" (kurz: Design) die Rede.

Wie sich Architektur und Design definiert und wo die Grenzen[6] untereinander und in der Implementierung liegen ist in anderen Arbeiten beschrieben. Das Carnegie Mellon Software Engineering Institute hat unterschiedliche Definitionen von Architektur zusammengetragen und auf ihrer Webseite[7] veröffentlicht. Die Definition zu Softwarearchitektur nach ISO/IEC 42010 [ISO] lautet jedoch wie folgt:

> „*Architecture: (system) fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution*"

Das Design einer Software wird nach Leonard Fehskens in „Re-Thinking Architecture" [Feh08] eher dem Entwickler, der im Rahmen der durch die Architektur vorgegebenen Grenzen die ingenieursmäßige Implementierung und Optimierung des Codes vornimmt, zugeschrieben. Ein Softwarearchitekt ist demnach für das

[6] Wobei selbst diese Grenzen „fließend" sein können [Fri10]
[7] Abrufbar unter http://www.sei.cmu.edu/architecture/start/glossary/community.cfm

Gesamtsystem in seinem umgebenden Kontext verantwortlich während ein Softwaredesigner sich innerhalb der Komponenten und Subsysteme bewegt, welche durch den Architekten vorgegeben wurden. Weitere Beispiele für die Differenzierung von Architektur und Design befinden sich in [Feh08, S. 58].

11.1 Ziele von Softwarearchitektur

Das Ziel einer Softwarearchitektur sollte die Minimierung der Gesamtkosten sowie die Vertretung der Interessen aller Stakeholder am System jeweils über den gesamten Produkt Lebenszyklus [Fri10] sein. Architekturentscheidungen geben den Handlungsspielraum für die Softwareentwickler vor und bewegen sich innerhalb definierter Grenzen. Diese Randbedingungen für Architekturentscheidungen können technischer oder organisatorischer (auch Budget) Natur sein und sowohl funktional („Was soll das System leisten") als auch nicht funktional („Wie gut soll das System die Leistung erbringen") sein [Sch11, S. 52]. Uwe Friedrichsen hat die Ziele, Aufgaben und Herangehensweisen von Softwarearchitektur in einer Grafik zusammengefasst:

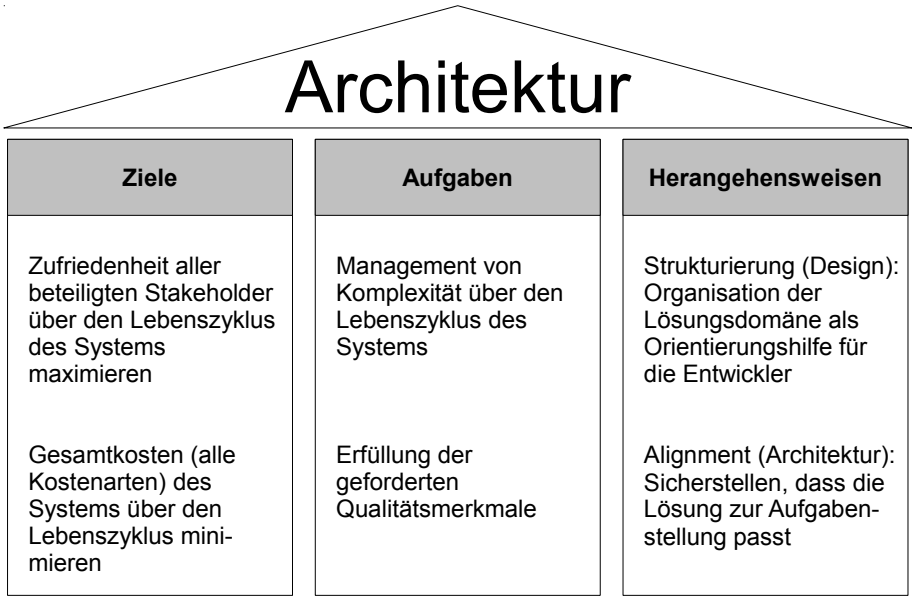

Abbildung 4: Ziele, Aufgaben und Herangehensweisen von Softwarearchitektur nach [Fri10, Abb. 1]

Friedrichsen unterteilt die Herangehensweisen im Kontext der Softwarearchitektur in die Aufgaben Strukturierung und Alignment, welche im folgenden beschrieben werden.

11.1.1 Strukturierung mit Softwarearchitektur

Wie in Abbildung 4 beschrieben ist Strukturierung im Kontext der Softwarearchitektur die Organisation der Lösungsdomäne als Orientierungshilfe für die Entwickler. Doch wieso wird Strukturierung in großen Projekten wichtig? Strukturieren wir etwas, so reagieren wir auf eine Beschränkung des menschlichen Gehirns nur eine bestimmte Anzahl unstrukturierter Informationen erfassen und bearbeiten zu können[8]. Übertragen auf die Entwicklung von Software bedeutet dies, dass Änderungen in einer Teilkomponente einer Software nur dann effektiv und richtig erfolgen können wenn die Zusammenhänge der Teilkomponente mit anderen Komponenten dem Entwickler klar vor Augen liegen. Die Voraussetzung hierfür ist, dass dem Entwicklungsteam die Struktur der Software klar ist, es sich auf die Konsistenz der Architektur verlassen kann und sich jedes Entwicklungsteam an die vorgegebenen Konventionen hält.

11.1.2 Alignment mit Softwarearchitektur

Das Alignment von Software beschreibt die Ausrichtung der Software auf deren Anforderungen. Beim Alignment von Software wird sichergestellt, dass möglichst viele Anforderungen der Stakeholder erfüllt werden und wenn nötig Kompromisse eingegangen und festgehalten werden. Das Alignment in der Softwarearchitektur erfordert demnach viel Kommunikation, Moderation und Konfliktmanagement. Sind die Anforderungen selbst noch nicht bekannt müssen diese ebenfalls im Rahmen des Alignment der Softwarearchitektur erarbeitet werden.

Während Strukturierung in der Softwarearchitektur theoretisch sowohl implizit während der Entwicklung als auch explizit erfolgen kann so findet Alignment ausschließlich explizit statt. Der Grund hierfür ist, dass die zum Alignment nötigen Dialoge mit allen Interessenvertretern der Stakeholder erfolgen müssen, um einen geeigneten Konsens finden zu können.

11.2 Scrum und Softwarearchitektur

Der Scrum Guide setzt keine Richtlinien wie das Thema Architektur in der Softwareentwicklung behandelt werden soll. Das Agile Manifest besagt, dass das Reagieren auf Veränderung mehr Wert ist als das Befolgen eines Plans. Scrum folgt dem agilen Manifest und trägt diesem Umstand Rechnung, indem Anpassungen an der Software inkrementell und immer auf Basis des aktuellen Stands der Software (aktuelles Inkrement) und Anforderungen (Product Backlog) erfolgt. Auch Architekturentscheidungen werden nach diesem Prinzip abgearbeitet und dann vom Entwicklungsteam implementiert, wenn sie benötigt werden. Diese Herangehensweise wird durch das Paradigma „*Deliver early and decide late*" beschrieben.

Ken Schwaber und Mike Beedle prägen in [SB08] des weiteren den Ausdruck „*Cut through the noise by taking action*". Beedle und Schwaber spielen hierbei

[8] Auch vergleichbar mit der Dunbar-Zahl, die besagt, dass wir durchschnittlich maximal 150 Beziehungen unter Freunden zuordnen können [Dun93, 681-735]

auf die langen Diskussionen an, die bei Architekturentscheidung vor der Entwicklung der Software entstehen. Ein weiterer Auszug aus dem agilen Manifest besagt, dass „Menschen und Interaktionen einen höheren Stellenwert haben als Prozesse und Werkzeuge". Folgt man den Empfehlungen von Schwaber, Beedle und dem Agilen Manifest so sollten Architekturentscheidungen kooperativ und erst wenn sie wirklich benötigt werden getroffen werden.

Kontrovers hierzu schreiben Ken Schwaber und Jeff Suthlerland in ihrem „A playbook for achieving enterprise agility" [SS12, Appendix 3, S. 181], dass bei größeren Projekten eine zusätzliche Releaseplanung welche einen „Architectural Runway" vorgibt, angebracht ist. Auch das Scaled Agile Framework [Lef] hält eine solche voraus laufende Architektur Roadmap bei größeren Projekten für angebracht und hat es in sein Framework entsprechend aufgenommen[9].

11.3 Herausforderung Softwarearchitektur

Scrum setzt keine explizite Architekturplanung voraus, jedoch ist bei großen Projekten eine Strukturierung und Ausrichtung (Alignment) der Software auf die Anforderungen erforderlich. Dies besagen auch zwei Prinzipien des Agilen Manifests [Bec01]:

> *„Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität."*

Die Dimensionen von „Groß" in diesem Kontext wurden in Kapitel II erläutert und werden durch die Erkenntnisse in [Fri10] weiter bekräftigt: Hat ein Projekt einen langfristigen Betrachtungshorizont, ist die Komplexität hoch, das Projekt groß, die Erfahrung in der Aufgabenstellung gering oder sind viele Risikofaktoren vorhanden so wird ein Architekt benötigt. Dass eine Architekturplanung bei großen, agilen Projekten sinnvoll ist wird in [SS12, Appendix 3] durch die Erfinder von Scrum untermauert. Wie dieser im Kontext von Scrum geartet ist, ist noch zu evaluieren. Es sollten jedoch folgende Faktoren beachtet werden:

1. Die Wünsche und Anforderungen der Stakeholder müssen sich sowohl in den fachlichen als auch in den technischen (Architektur) Anforderungen wiederfinden
2. Die fachlichen als auch technischen Anforderungen und Wünsche sollten gleichgestellt behandelt werden können
3. Eine Überplanung von Architektur muss vermieden werden um den produzierten Waste (siehe [PP03, S. 4, Table 1.1]) in der Software gering zu halten
4. Die Verantwortlichkeit für Architekturfragen muss klar definiert sein

[9] Im Detail abrufbar unter http://scaledagileframework.com/architectural-runway/

Die Praxis zeigt weiterhin, dass der Wasserfall-ähnliche Ansatz des umfangreichen vorausplanens (*Big-Design-Up-Front*. Siehe [Coc03]) vor der Implementierung nicht oder nur eingeschränkt funktioniert[10]. Auch das ganzheitliche vernachlässigen von vorausschauender Architekturplanung (*No-Design-Up-Front*. Siehe [Fow04]) führt nach den in den vorangegangen Kapiteln genannten Gründen nicht zum gewünschten Erfolg. In der Umsetzung muss demnach ein Mittelweg gefunden werden, welcher sich zwischen dem Big-Design-Up-Front und dem des No-Design-Up-Front befindet und auf einem Prinzip des Agilen Manifests aufbaut:

> *„Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams."*

12 Unternehmenskultur

Eine Unternehmenskultur beschreibt die Sammlung von Traditionen, Werten, Regeln, Glaubenssätzen und Haltungen, die einen durchgehenden Kontext für alles bilden, was in einer Organisation getan und gedacht wird [MM85, S. 2-20]. Oder wie es Bright und Parkin [BP97] 1997 kurz und prägnant beschrieben haben:

> „*This is how we do things around here*".

William Schneider beschreibt in seinem Buch „The Reengineering Alternative" [Sch99] vier unterschiedliche Arten von Unternehmenskulturen. Schneider stellt diese vier Kulturen in einer 2x2 Matrix wie folgt dar:

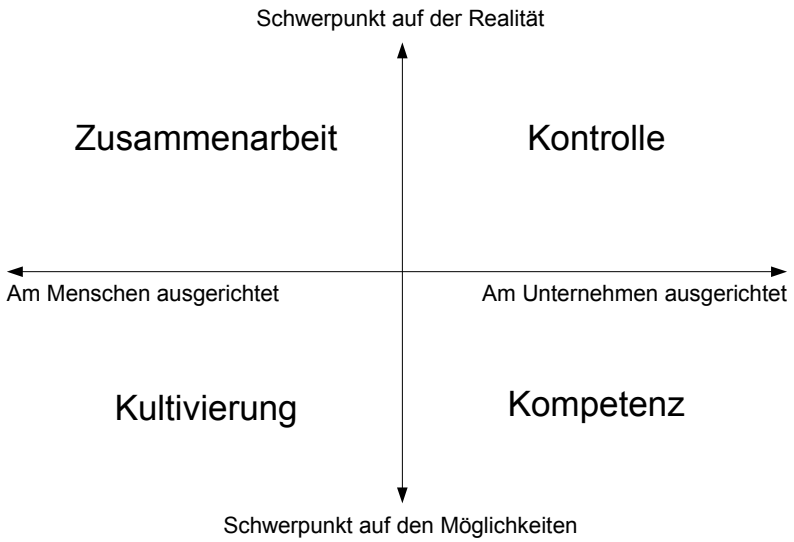

Abbildung 5: Das „Schneider culture model" in Anlehnung an [Sch99]

[10] Projekte, welche mit Wasserfall-ähnlichen Prinzipien durchgeführten wurden sind 17% [Sta10, S. 15] bis 28% [Amb10, S. 4] weniger erfolgreich als agile. Fallbeispiele befinden sich im Teil IX dieser Arbeit.

Nach Schneider wird keine der vier Kulturen in Unternehmen in Reinform vorkommen. Zwar wird es immer eine Kernkultur geben, allerdings werden dort immer Aspekte anderer Kulturen integriert sein. Verschiedene Umfragen ([Max12] und [Spa10]) belegen, dass Unternehmen, die mit Scrum arbeiten, sich als Kultur der Zusammenarbeit und Kultur der Kultivierung sehen. Beides sehr am Menschen ausgerichtete Kulturen. Weiterhin ist wichtig, dass die vier Kernkulturen zueinander wertneutral sind. Jede kann in ihrem Umfeld sehr erfolgreich sein.

Die Kulturen und deren Ausprägungen werden in den folgenden Kapiteln noch im Detail beschrieben. Doch warum ist Unternehmenskultur wichtig bei der Skalierung von Scrum? Unternehmenskultur ist prinzipiell ein wichtiges Thema sowohl bei der Ausrichtung des Unternehmens als auch bei der Mitarbeiterführung. Dies gilt sowohl bei kleinen, internen Projekten als auch bei großen Projekten mit verteilten Teams. Doch je größer ein Projekt ist, desto wahrscheinlicher ist es, dass mit mehreren Abteilungen und Unternehmen zusammengearbeitet wird die, wenn das Thema nicht entsprechend adressiert wird, nach den unterschiedlichsten Unternehmenskulturen arbeiten. In der jährlich durchgeführten „State of Agile Survey" von Version One werden unter anderem die Hauptgründe aufgeführt, warum agile Projekte scheitern[11]. 6% der befragten gaben an, dass ihr Projekt an einem „Mangel an Kultureller Verständigung" gescheitert ist oder, auf Rang 3 mit 11%, an einem „Mangel an Verständnis wie sie mit organisatorischem Wandel umgehen sollen" [Ver11, S. 5]. Unternehmenskultur ist somit ein Thema, welches im Kontext der Herausforderungen bei der Skalierung von Scrum adressiert werden sollte.

12.1 Zusammenarbeit

„*Wir sind erfolgreich durch Zusammenarbeit*"

Ein Unternehmen, das im Kern nach der „Collaboration" [Sch99, S. 44] Kultur lebt setzt seinen Fokus auf Vielfalt, Vertrauen, Partnerschaften, die Arbeit in Teams, ein gesundes Miteinander und Gleichberechtigung. Demnach sind Synergien in dieser Unternehmenskultur sehr wichtig. Es ähnelt damit dem Archetypus der Familie und/oder dem einer Sportmannschaft [Sch99] und folgt dem Agilen Manifest indem es dem Grundsatz „*Individuen und Interaktionen mehr als Prozesse und Werkzeuge*" [Bec01] intuitiv folgt.

Bei einer 2010 durchgeführte Umfrage zum Thema „Welche Kultur hätte ein ideales agiles Unternehmen?" [Spa10] befindet sich die Kultur der Zusammenarbeit mit 47% auf Rang 1.

Im Kontext von Scrum fehlen einem Unternehmen mit dieser Kernkultur eventuell noch ein paar wenige Aspekte der Unternehmenskultur der Kultivierung wie z.B. eine Vision, die Verbundenheit zu Zielen und den Mut manche Dinge sich einfach entwickeln zu lassen [Max12, S. 12]. Nach den Erfahrungen der IBM funktioniert ist eine solche Kultur mit agilen Teams am effektivsten [Amb09b, S. 15] und entspricht am ehesten der „Lean Development Governance" [AK07].

[11] Wobei auf Rang 1 mit 16% angegeben wurde, dass noch keines der agile durchgeführten Projekte gescheitert ist.

Die Lean Development Governance ist eine Weiterentwicklung der von Mary und Tom Poppendieck vorgestellten sieben Prinzipien des Lean Software Developments [PP03] und beschreibt, wie eine Unternehmensführung im Kontext des Lean Developments aussehen könnte.

12.2 Kultivierung

> „*Wir sind erfolgreich durch das Wachstum unserer Mitarbeiter und die Erreichung unserer Vision*"

Ein Unternehmen, dass im Kern nach der „Cultivation" [Sch99, S. 81] Kultur lebt setzt seinen Fokus auf Hingabe, Werte, die Zweckmäßigkeit von Aufgaben, Kreativität und hat viel Mut Dinge sich entwickeln zu lassen. Im Grundsatz existiert diese Kultur um das Wachstum seiner Kunden zu fördern und das Potential seiner Kunden voll auszuschöpfen. Sie strebt an bestimmte Ideale zu verfolgen und weitreichende Visionen zu erreichen. Nach [Sch99] ähnelt diese Kultur dem Archetypus eines religiösen Systems welches nach Wachstum und Selbstverwirklichung strebt. Ph.D William Schneider hat die Unternehmen Celestial Seasonings, Herman Miller, Esprit de Corp, und 3M im Jahr 1999 exemplarisch als Unternehmen aufgeführt, die nach der Kultur der Kultivierung leben[12].

Nach der Umfrage von Michael Spayd [Spa10] würden 41% der befragten eine Kultur der Kultivierung als eine ideale Kultur eines agilen Unternehmen sehen. Um Scrum in einem Unternehmen, dass nach dieser Kultur lebt, einzuführen und weiter zu entwickeln sind nur ein paar weitere Aspekte nötig. Diese sind in der benachbarten Kultur der Kommunikation zu Hause und lauten Teamwork, partnerschaftliches Arbeiten und Diversität [Max12, S. 14].

12.3 Kompetenz

> „*Wir sind erfolgreich da wir die Besten sind*"

Expertise und Effizienz sind die Ausprägungen einer „Competence" [Sch99, S. 63] Kultur. Mitarbeiter finden ihren Antrieb in einem Unternehmen mit dieser Kernkultur in neuen Herausforderungen, Gewinnen, dem Wettbewerb, Auszeichnungen und Anerkennung. Der Archetypus eines derartig geführten Unternehmens gleicht dem einer Universität - das erreichen von persönlichen Auszeichnungen sowie das Abliefern eines perfekten Produkts ist der oberste Ansporn [Sch99].

Nach [Spa10] würde man mit 9% ein Unternehmen mit dieser Kernkultur eher selten in einem idealen agilen Umfeld antreffen. Produkte oder Dienstleistungen, die in einem Unternehmen dieser Art entwickelt werden sind meistens unabhängig voneinander und basieren auf den Ideen und der Arbeit einzelner Mitarbeiter. Innerhalb des Unternehmens herrscht ein hoher Wettbewerb untereinander und damit wenig Zusammenhalt in Teams.

[12] Abrufbar unter http://www.parshift.com/Speakers/Speak016.htm

Die in einer Organisation vorhandene Professionalität und Expertise[13] kann für die Entwicklung von Scrum genutzt werden jedoch nicht die vorhandene Fokussierung auf Individuen statt auf Teams (siehe [Bec01] und [Max12, S. 16]). Eine Kulturänderung ist demnach erforderlich.

12.4 Kontrolle

„*Wir sind erfolgreich durch Erzielung und Erhaltung von Kontrolle*"

Standards und Prozesse und die damit verbundene Vorhersagbarkeit sind für eine „Control" [Sch99, S. 28] Kultur sehr wichtig. Mitarbeiter von Unternehmen, die nach dieser Kultur arbeiten sind häufig außerordentlich sachlich, überkontrollierend und somit unpersönlich. Der Archetypus entspricht nach Schneider dem eines militärisch geführten Systems.

Da sie ebenfalls einen sehr hohen Wert auf Hierarchien und Einzelleistungen legen ist es nicht verwunderlich, dass nach [Spa10] lediglich 3% der befragten eine Unternehmenskultur der Kontrolle als ein ideales agiles Unternehmen sehen würden. Die Einführung und Entwicklung von Scrum in einem nach dieser Kernkultur geführten Unternehmens ist äußert schwierig und kann nur über einen sehr langen Zeitraum Stück für Stück erfolgen [Max12, S. 18]. Nach den Erfahrungen der IBM funktioniert eine solche „command and control" - Struktur mit agilen Teams „nicht sehr gut" [Amb09b, S. 15]. Eine Kulturänderung ist demnach erforderlich.

12.5 Herausforderung Unternehmenskultur

Unternehmen oder Abteilungen, welche unterschiedlichen Kulturformen folgen und zusammenarbeiten sollen werden nach William Schneider häufig Konflikten gegenüberstehen. Besonders Kulturen, die nach Abbildung 5 diagonal zueinander stehen sind besonders schwierig in Einklang zu bringen.

Je nachdem welche Unternehmenskulturen vorliegen ist es unterschiedlich schwierig Scrum nachhaltig in das Unternehmen zu etablieren und zu skalieren. Dies kann mitunter ein mehrjähriges unterfangen sein (bis zu 10 Jahre und mehr nach [Kot96] und dem Management Experten Peter Drucker [Dru92]), besonders wenn das Unternehmen nach der Kontroll- oder Kompetenzkultur lebt. Einfacher ist es bei Unternehmen, die nach der Kultur der Kultivierung und Zusammenarbeit miteinander arbeiten.

Diese Grenzen des zusammenwachsen stellen Walsham, Sahay und Krishna in Ihrem Artikel in [KSW04, S. 65] deutlich heraus:

> *„Große Unterschiede in Normen und Werten können nicht harmonisiert werden, da sie von tief sitzenden Unterschieden in den kulturellen Hintergründen, der Ausbildung und dem Arbeitsleben abgeleitet werden. Beispiele beinhalten die Haltung gegenüber Hierarchien und Macht und unterschiedlichen Geschäftspraktiken."*

[13] Eine Comptence Kultur lebt nach [Sah12] den Leitsatz „Continuous attention to technical excellence and good design enhances agility"

Die Herausforderung bezüglich der vorherrschenden Unternehmenskulturen besteht demnach darin, diese zu identifizieren[14], zu entwickeln und Partnerunternehmen zu finden, die nach Möglichkeit nach der gleichen Unternehmenskultur arbeiten, denn:

> "*Culture — no matter how defined — is singularly persistent. In fact, changing behavior works only if it can be based on the existing 'culture'.*" [Dru92, S. 192]

Wird Scrum in seiner empfohlen Größe (3-9 Entwickler), in einem virtuellem Scrum Studio [SS12, S. 55ff] oder als Scrum PRN[15] betrieben, so bewegt sich Scrum innerhalb einer Abteilung und eine Organisationsentwicklung ist nicht nötig. Ob eine Kulturentwicklung nötig ist hängt hierbei ausschließlich von der vorliegenden Kultur in der Abteilung ab und der Aufwand von der dort vorherrschenden Kultur. In einem skalierten Umfeld, in dem mehrere Abteilungen, Unternehmensbereiche oder Organisationen involviert sind ist ein Veränderung oder Entwicklung der Kernkultur des Unternehmens nötig und stellt demnach eine Herausforderung bei der Skalierung von Scrum dar.

13 Projekt- und Prozessmanagement

Unter Prozessmanagement[16] ist die Planung, Gestaltung, Lenkung, Koordination, Realisierung und das Controlling von Aktivitäten und Maßnahmen innerhalb der Prozessausführung zu verstehen, welche funktionsübergreifend für eine definierte Aufgabenstellung zur Erreichung vorgegebener unternehmerischer Zielvorgaben sowie zur ständigen Prozessverbesserung erforderlich sind [Bin97, S. 1-9]. Im Kontext der Softwareentwicklung mit Scrum sind typische, definierte Prozesse z.B. die Pflege des Product Backlogs, die Planung der Aufgaben für den nächsten Sprint im Sprint Planning Meeting oder, im Sinne der ständigen Prozessverbesserung, das Sprint Review Meeting.

Die Scrum-Prozesse sind für ein Scrum Team somit in sich geschlossen und über die Scrum Meetings, Artefakte und Rollen definiert. Kommen weitere Teams oder Teams an anderen Standorten hinzu so, werden weitere Anforderungen an das Projekt- und Prozessmanagement gestellt. Das in Kapitel 10 erwähnte „System and Software Consortium" (SSCI) hat in einer dreijährigen Umfrage unzureichendes Projekt- und Prozessmanagement als ein großes Hindernis in der verteilten Entwicklung von Software ermittelt [ND05, Table 2]. Die Gründe hierfür sieht das SSCI in den unterschiedlichen Reifegrade der in komplexen Projekten involvierten virtuellen Teams oder Organisationen. Dieser Umstand macht

[14] William Schneider hat in „The Reengineering Alternative: A Plan for Making Your Current Culture Work" einen Fragebogen [Sch99, S. 20] zur Evaluierung der Unternehmenskultur beigefügt

[15] Scrum PRN bedeudet „pro re nata" was so viel heißt wie "wie es die Umstände erfordern". Die Notation PRN bedeutet demnach "nimm das was gebraucht wird".

[16] Die Begriffe Prozessmanagement, Projektmanagement, Geschäftsprozessmanagement oder Business Process Management (BPM) werden je nach gewählter Literatur und Land gleichermaßen oder unterschiedlich genutzt. In dieser Arbeit wird der Begriff „Prozessmanagement" dem „Business Process Management" gleichgestellt.

es schwierig einen gemeinsamen Standard für das abwickeln der Standardprozesse zu finden und stellt hierbei neue Herausforderungen an die Synchronisation der Teams, den einfachen und sicheren Austausch von Informationen sowie das kollaborative Entwickeln und integrieren in ein gemeinsames Produkt.

13.1 Kollaborationswerkzeuge

Das SSCI sieht die Hauptursache in mangelndem Projekt- und Prozessmanagement darin, dass unzureichende Kollaborationswerkzeuge genutzt werden [ND05, S. 70]. Diese legen den Fokus zu sehr auf die „hard skills" wie Workflow Management, Budgetierung, Zeitplanung und das Dokumentieren von Anforderungen. Die folgenden „soft skills" fallen nach dem SSCI in den meisten Kollaborationswerkzeugen hierbei zu kurz aus:

- Definieren des Geschäftswertes
- Definieren einer gemeinsamen Produktvision
- Der Aufbau von Teams
- Das Auflösen von Problemen
- Das mildern von Risiken

Diese Mängel können nach [LR08, Car99, LS08] nur durch möglichst reichhaltige Kommunikationsmittel wie Videotelefonie oder Gespräche von Angesicht zu Angesicht ausgeglichen werden. Dies entspräche auch einem Prinzip des agilen Manifests [Bec01]:

> *„Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteam zu übermitteln, ist im Gespräch von Angesicht zu Angesicht."*

Während seither der Einsatz von reichhaltigen Kommunikationsmitteln sehr teuer und örtlich begrenzt war so ist es heutzutage möglich auf eine große Bandbreite von Echtzeit Kollaborationswerkzeugen zurückzugreifen, die auf jedem Desktop PC installiert werden können. Wichtige Kriterien für die Auswahl geeigneter Werkzeuge können sein:

- Sicherheitsbestimmungen des Projekts und der involvierten Unternehmen
- Kompatibilität mit allen genutzten Plattformen und Datenformaten
- Die vorherrschende Verfügbarkeit an Rechenleistung und Datenübertragungsraten

Nach dieser Vorauswahl ist es wichtig den Benutzerkreis zu involvieren bevor das Kollaborationswerkzeug installiert wird. Die Implementation einer Plattform nach dem Prinzip „if you build it they will come" führt, nach den Erfahrungen des SSCI, zu einer Ablehnung des Werkzeugs durch den Nutzerkreis. Die Herausforderung des Projekt- und Prozessmanagements lautet hierbei geeignete Werkzeuge zu finden und, in Abstimmung mit dem Nutzerkreis, im gesamten Projektumfeld zu implementieren.

13.2 Synchronisation von Teams

Ein Scrum Team liefert nach jedem Sprint ein Produktinkrement das, zusammen mit den bisherigen Inkrementen, ein potentiell auslieferbares Produkt ergibt. Idealerweise nutzen die Entwickler für das Zusammenführen ihres Codes ein Versionskontrollsystem und die vorhandenen Unit- und Modultests prüfen den eingereichten Code auf Kompatibilität mit dem restlichen Programmcode. Doch wie und wann synchronisieren sich mehrere Scrum Teams? Die erste Frage, die sich stellt, bevor überhaupt eine Zeile Code geschrieben wurde lautet: Wie erfolgt die Aufteilung der Aufgaben auf die verschiedenen Teams? Im Buch von Craig Larman und Bas Vodde [LV10] werden mehrere Arten der Team- und Aufgabenverteilung vorgestellt. Die Herausforderung an das Projekt- und Prozessmanagement besteht hierbei darin, die für das Projekt am besten geeignete Team- und Aufgabenverteilung zu evaluieren und zu implementieren.

Ist die Teambildung abgeschlossen und wurden die ersten Anforderungen abgearbeitet so stellt sich die Frage wie das erstellte individuelle Arbeitsergebnis mit den Arbeitsergebnissen der anderen Mitarbeitern und Teams zusammengeführt wird. Wurde eine Konsens für eine gemeinsame Definition of Done (siehe Kapitel 7.5) gefunden und etabliert so muss nun geklärt werden wie und wann das Zusammenführen der Arbeitsergebnisse erfolgt.

Das Zusammenführen der Produktinkremente wird langfristig einfacher vonstatten gehen wenn die Inkremente möglichst viele Gemeinsamkeiten und somit Schnittstellen aufweisen. Natürlich wird kurzfristig das zusammenführen komplett unterschiedlicher Inkremente kein Problem darstellen, da sie wenige oder gar keine gemeinsame Schnittstellen aufweisen und eine Integration somit unproblematisch ist aber wenn dann zu einem späterem Zeitpunkt das „Puzzle geschlossen wird" ist der großer Knall praktisch vorprogrammiert. Demnach ist ein weiterer Knackpunkt bei mehreren Teams die Ausrichtung und Anordnung („Alignment") der Teams hinsichtlich des aktuellen Fokus der Anforderungen und der Sprintvision.

Ein weiteres Risiko bei der Entwicklung von Software mit mehreren Teams sind Abhängigkeiten der Teams untereinander, die zu Verzögerungen in der Produktentwicklung führen. Erfolgt das erwähnte Alignment der Teams so kann dieses Risiko gemindert aber nicht gänzlich aufgehoben werden. Die Herausforderung des Projekt- und Prozessmanagements besteht hierbei darin geeignete Strukturen (siehe auch Kapitel 14) zu bilden, die Abhängigkeiten von verschiedenen Teams möglichst frühzeitig aber auch ad hoc erkennen und auflösen können.

13.3 Herausforderungen an das Projekt- und Prozessmanagement

Mehrere oder verteilte Teams lassen sich nicht mehr ausschließlich durch eine iterative und empirische Vorgehensweise ideal steuern. Die Praxis (siehe Anhang dieser Arbeit) zeigt, dass hier weitere Vorüberlegungen nötig sind um das Potential mehrerer Teams voll auszuschöpfen. Zu den Herausforderungen des Projekt- und Prozessmanagements hinsichtlich der Teambildung und Koordination gehören zusammengefasst:

- Die Evaluierung geeigneter Teamstrukturen und der Zuschnitt der Anforderungen auf diese
- Die Evaluierung geeigneter Integrationsstrategien um die einzelnen Produktinkremente zusammenzuführen
- Das Kommunizieren gemeinsamer Sprintvisionen um den aktuellen Fokus in der Produkterstellung zu erhalten
- Das Alignment der Teams um Abhängigkeiten frühzeitig zu erkennen und aufzulösen

Auch an die Kollaborationswerkzeuge der Teams werden neue Herausforderungen gestellt. Diese gilt es durch das Projektmanagement unter der Berücksichtigung des Unternehmensumfelds zu evaluieren und in Absprache mit den Benutzern Projektweit zu implementieren.

14 Unternehmensstruktur

Eine Unternehmensstruktur beschreibt die festgelegten vertikalen und horizontalen Regeln und Strukturen, nach denen die Abteilungen und deren Mitarbeiter miteinander arbeiten. Sie gibt die Beziehungen und Hierarchien zwischen den Abteilungen und damit ein grobes Rahmenwerk für die Erfüllung der Aufgaben vor.

In „Organization and Environment" [Law86] beschrieben Lawrence und Lorsch 1986, dass es „nur einen besten Weg geben kann, eine Organisation zu strukturieren" und dies unter allen denkbaren gegebenen Umständen. Dieser Ansatz fokussiert sich auf Bürokratie, strenge Regelungen bei der Entscheidungsfindung, eine klar definierte Hierarchie sowie exakte Tätigkeitsbeschreibungen für jeden Mitarbeiter. Sicherlich hat diese Art der Unternehmensstruktur auch seine Vorteile - vor allem im damaligen Industriezeitalter.

Seit 1930 entwickelte sich jedoch ein weiterer Ansatz, der den individuellen Mitarbeiter und die Erfüllung seiner psychologischen sowie sozialen Gebräuche adressiert. Nach diesem Ansatz sollten Entscheidungen durch das Management nach den Anforderungen der Angestellten getroffen werden und in allen Ebenen kollaborativ erfolgen. Diese „Contingency Theory" wurde 1996 durch Lex Donaldson in „For Positivist - Organization Theory" [Don96] festgehalten. Sie basiert, ähnlich den Grundgedanken der agilen Softwareentwicklung [Bec01], auf dem Konzept der Unvorhersehbarkeit. Je höher der Grad der Unvorhersehbarkeit ist desto weniger formal, zentralisiert und spezialisiert sollte die Unternehmensstruktur sein [Don96, S. 58].

14.1 Ziele einer Unternehmensstruktur

Das Ziel einer Unternehmensstruktur ist es die menschliche Arbeitsteilung aufgrund von unterschiedlichen Anforderungen und der anfallenden Arbeitsmenge auf eine möglichst sinnvolle Art und Weise zu Organisieren. Eine gute Arbeitsteilung zeichnet sich dadurch aus, dass durch die entstehenden Schnittstellen

möglichst keine Nachteile in der Effizienz und der Qualität der Arbeit entstehen. Weitere Faktoren, die bei der Arbeitsteilung eine Rolle spielen sind die Flexibilität sowie die Steuerbarkeit der entsprechenden Teilbereiche.

Allerdings gibt es keine *ideale* Unternehmensstruktur [Don96]. Davon abgesehen, dass sich eine Unternehmensstruktur in einem ständigen Wandel mit dem Wachstum des Unternehmens befindet so ist sie immer ein Kompromiss. Auf der einen Seite strebt eine Unternehmensstruktur die Effizienz eines Mitarbeiters an auf der anderen seine Flexibilität. Idealerweise sollte das gesamte Unternehmen von einer zentralen Einheit aus steuerbar sein, andererseits möchte man aber auch nicht die schnelle Entscheidungsfähigkeit vor Ort verlieren. Eine Organisationsstruktur ist demnach immer höchstens untere den gegebenen Umständen *bestmöglich geeignet* für den jeweiligen Zweck.

14.2 Ausprägungen von Unternehmensstrukturen

Es gibt viele Arten von Unternehmensstrukturen: Die Funktionale Struktur, die Produktlinienstruktur, die geografische Struktur, die Matrix-Struktur, die Holding-Struktur und viele weitere. Diese im einzelnen zu evaluieren würde den Rahmen dieser Arbeit sprengen - vor allem, weil die unterschiedlichen Strukturen auch in Mischformen in der Praxis vorhanden sind. Auch hierdurch wird deutlich, dass es *eine ideale* Lösung für eine Organisationsstruktur gar nicht geben kann. Die Schnittstelle zu skalierten Projekten mit Scrum findet sich in einer Ausprägung aller Unternehmensstrukturen: Dem Grad der Zentralisierung beziehungsweise Dezentralisierung.

Bei kleinen Unternehmen obliegt die Entscheidungsfindung sowie die Kontrolle und Planung von Aufgaben meist einzelnen, wenigen Geschäftsführern. Bei mittelständischen Unternehmen würde ein derartig hochgradig zentralisiertes Unterfangen das Wachstum der Organisation negativ beeinflussen. Ob für ein großes Unternehmen ein eher zentraler oder ein eher dezentraler Ansatz gewählt wird hängt von verschiedenen Faktoren [Rec11] ab[17]:

- Größe des Unternehmens (siehe Kapitel 6)
- Geographische Ausdehnung des Unternehmens (siehe Kapitel 10)
- Geschwindigkeit, mit der Entscheidungen getroffen werden müssen
- Arbeitsbelastung der Entscheidungsträger
- Die vorliegende Unternehmenskultur (siehe Kapitel 12)
- Status der Planungs-, Kontroll-, und Informationssysteme (siehe Kapitel 13.1)

Die Vor- und Nachteile von einer zentralen Unternehmensstruktur gegenüber einer dezentralen stellt Dagmar Recklies in [Rec11, S. 3] gegenüber.

[17] Hinsichtlich der Relevanz zu skalierten Projekten mit Scrum gekürzte Liste

	Vorteile	Nachteile

Zentralisierung	• Konsistenz in der Unternehmensstrategie für alle Geschäftseinheiten
	• Einfachere Koordination der einzelnen Aktivitäten und ihrer gegenseitigen Wechselwirkungen
	• Einfachere Kontrolle
	• Veränderungen in der strategischen Grundausrichtung können besser unterstütz werden
		• U.U. zu langsame Reaktion auf Veränderungen, die nur einzelne Geschäftseinheiten und nicht das Unternehmen als Ganzes betreffen
		• Gefahr der Entwicklung einer übergroßen Unternehmenszentrale die
		  • Stark von Management-Informationssystemen abhängt,
		  • Zu wenig Kundennähe hat,
		  • Vielfältigen Interessen und komplexen Zusammenhängen gegenübersteht
		• Das Unternehmen entwickelt keine Manager mit echten strategischen Fähigkeiten. Stattessen agieren Experten für verschiedene Funktionen, die schwer koordiniert werden können.

Dezentralisierung	• Schnelle Änderung von Wettbewerbsstrategien und geschäftsbereichsbezogenen Strategien • Höhere Motivation durch mehr Eigenverantwortlichkeit • Bessere strategische Steuerung von sehr komplexen Unternehmensgruppen	• Probleme bei der Zuordnung von Kompetenzen zur Zentrale und den einzelnen Geschäftseinheiten • Effizienzverluste durch Duplikation verschiedener Funktionen • Probleme, die Entscheidungskompetenten einzelner Manager genau mit ihren Verantwortlichkeiten abzustimmen

Abbildung 6: Vor- und Nachteile einer (De)-zentralen Unternehmensstruktur nach [Rec11, S. 3]

Es ist zu erkennen, dass sowohl ein zentral als auch ein dezentral geführtes Unternehmen verschiedene Vor- und Nachteile mit sich bringt, die auch bei der Skalierung von Projekten mit Scrum eine Rolle spielen:

- Die Koordination einzelner Aktivitäten ist eine Herausforderung bei der Skalierung von Scrum (siehe Kapitel 13.3) und wird als ein Vorteil bei der zentralen Unternehmensstruktur aufgeführt.
- Eine dezentrale Unternehmensstruktur hat, wie auch die Mitglieder eines Scrum Teams, eine hohe Motivation durch Eigenverantwortlichkeit.
- Das Reagieren auf Veränderung wird im Agilen Manifest [Bec01] groß geschrieben, sowie auch bei einer dezentral geführten Organisation.
- Eine hohe Distanz zum Kunden wird einem zentral geführtem Unternehmen zugeschrieben - Kundennähe wird aber von den Prinzipien des Agilen Manifests gefordert.

Eine klare Empfehlung für eine bestimmte Unternehmensstruktur in Reinform kann daher nicht gegeben werden. Vielmehr muss sich die Lösung in einer Mischform dieser befinden, die es zu evaluieren gilt.

14.3 Herausforderung Unternehmensstruktur

In den vorangegangenen Kapiteln wurden zahlreiche Herausforderungen bei der Skalierung von Scrum beschrieben. Die Evaluierung und Lösungsfindung zu einer Herausforderung geschieht immer im Kontext einer bestehenden Unternehmensstruktur. Eine Organisationsstruktur darf aber einer idealen Lösungsfindung nicht im Wege stehen.

> *„Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen."*[18]

Agilität darf also nicht bei der Unternehmensstruktur aufhören sondern macht es erforderlich, dass ad hoc neue Teams und Gremien gebildet werden können um die Mitarbeiter und somit die Produktentwicklung ideal zu unterstützen. Dies darf natürlich nicht Blind geschehen sondern immer gemäß der Scrum Prinzipie „Inspect & Adapt" [SS11, S. 4].

Welche Unternehmensstruktur für eine Organisation am besten geeignet ist, wenn sie Projekte mit mehreren und/oder verteilten Teams angehen will, kann nicht pauschalisiert werden. Es muss vielmehr im einzelnen evaluiert werden, welche Unternehmensstruktur die *bestmöglich geeignetste* für den aktuellen Zweck ist.

[18] Ein Prinzip des Agilen Manifests [Bec01]

Teil V

Lösungsmöglichkeiten für die Skalierung von Scrum

Im Teil III dieser Arbeit wurden Herausforderungen aufgeführt, die es bei der Skalierung der Scrum Artefakte, Rollen und Meetings zu adressieren gilt. Hierbei kann es nie eine allgemeingültige Lösung geben, da diese immer abhängig vom Kontext sind, in denen sich ein Projekt bewegt. Die in einem Projekt vorherrschenden Bedingungen wie Teamgröße, Teamverteilung (siehe Kapitel 6) sowie eventuell weitere Komplexitätstreiber [Amb09b, S. 22-24] schließen diese allgemeingültige Lösung aus. In den folgenden Kapiteln werden jedoch Vorschläge unterbreitet, wie die in Teil III dieser Arbeit angesprochen Probleme bei der Skalierung von Scrum gelöst werden könnten und was die Alternativen sind.

15 Scrum Artefakte

15.1 Product Backlog

> „*Einfachheit - die Kunst, die Menge nicht getaner Arbeit zu maximieren - ist essenziell.*"
>
> Das Agile Manifest [Bec01]

15.1.1 Ein Product Backlog

Gibt es in einem skalierten Scrum Projekt weiterhin nur ein Product Backlog, so hat dies die Prämisse, dass die Product Backlog Items (wie in kleinen Projekten) priorisiert sind und dass mindestens so viele Product Backlog Items mit hoher Priorität komplett ausgearbeitet sind, wie alle Entwicklungsteams in einem Sprint erledigen könnten. Aufgrund der strikten Timebox eines Sprints kann es nach dem Schätzen der Anforderungen vorkommen, dass Product Backlog Items mit einer niedrigeren Priorität vorgezogen werden, da eventuell ein Product Backlog Item mit einer hohen Priorität nicht mehr in den Sprint passen würde. Bei vielen Teams steigt dieses Risiko an, wodurch es erforderlich wird, dass der oder die Product Owner deutlich mehr Product Backlog Items ausformulieren müssen als bisher.

Ein weiteres Problem ist die Verteilung der Product Backlog Items selbst. Welches Team hat beim Sprint Planning Meeting 1 das Vorrecht auf ein bestimmtes Product Backlog Item? Hier gibt es mehrere Möglichkeiten wie dies geregelt werden könnte:

1. Die Teams haben „Hüte" auf, die sie für die Erledigung bestimmter Aufgaben privilegieren. So kann es zum Beispiel innerhalb einer Anwendung für eine Versicherung ein Team geben, dass sich exzellent mit Hausratsversicherungen auskennt und somit privilegiert für derartige Aufgaben ist.

Im Sinne der Interdisziplinarität von Teams bedeutet dies allerdings nicht, dass sich dieses Team immer um derartige Product Backlog Items zwangsläufig bewerben muss.

2. In „Software in 30 Days" [SS12, S. 55ff] wird zur Einführung in Scrum das Scrum Studio beschrieben, welches zuerst mit einem Team beginnt und dann weitere Teams rekrutiert. Wenn Scrum nach diesem Prinzip eingeführt wurde so könnte man festhalten, dass wenn sich mehrere Scrum Teams auf ein Product Backlog Item bewerben das jeweils ältere Scrum Team das Vorrecht hat.

Generell sollte die Aufteilung von Product Backlog Items die Aufgabe der Entwicklungsteams sein. Sie sollten selbstständig miteinander „verhandeln" und so die hoch priorisierten Anforderungen aus dem Product Backlog untereinander aufteilen können. Die aufgezählten Punkte des „Vorrechts auf ein Product Backlog Item" sollten demnach bestenfalls Rückfallstrategien sein, falls sich die Teams überhaupt nicht einigen können. Ein Manko dieser Art der Aufteilung ist, dass bei vielen und verteilten Teams die Verhandlung über Product Backlog Items mitunter sehr lange dauern kann.

Hinzu kommt das Thema Abhängigkeiten: Wenn die Verhandlungen ungeschickt verlaufen so können die Anforderungen so aufgeteilt worden sein, dass mehr als zwei Teams mit der Auflösung einer Abhängigkeit betraut wurden. Dies erzeugt wiederum einen hohen Koordinationsaufwand während der Sprints.

Des Weiteren sollte die Anzahl an Product Backlog Items in einem Product Backlog die Grenze von 100 bis 150 Einträgen nicht überschreiten. Diese Grenze rührt ebenfalls von der in Kapitel 11.1.1 erwähnten Dunbar-Zahl [Dun93, 681-735] her. Dies kann entweder durch das Zusammenfassen von Product Backlog Items zu sogenannten „Epics" erfolgen oder durch die Einführung von gefilterten Sichten auf das Product Backlog. Mike Cohn hat dies in seinem Buch „Agile Softwareentwicklung - Mit Scrum zum Erfolg" [Coh10] wie folgt illustriert:

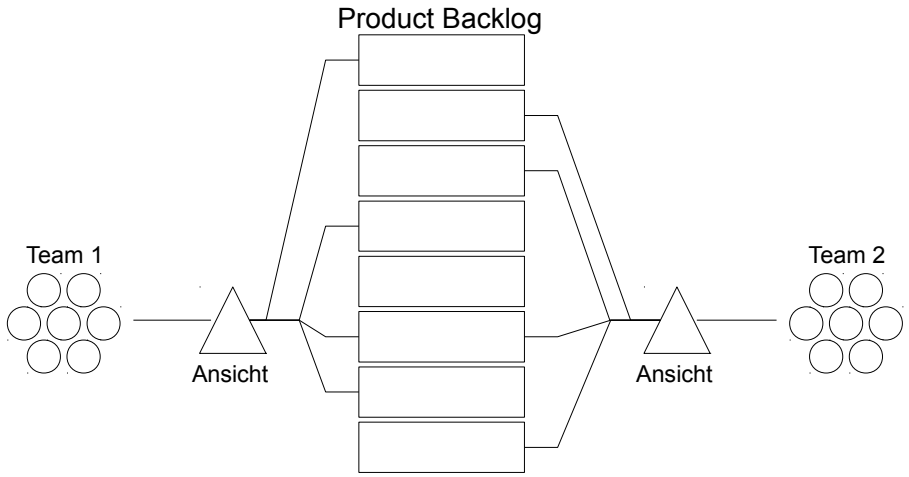

Abbildung 7: Ein Product Backlog mit verschiedenen Ansichten. In Anlehnung an [Coh10, Abbildung 17.2]

Die Abbildung zeigt ein Product Backlog mit vielen Einträgen, die jedoch für die Teams nur als Untermenge sichtbar sind.

15.1.2 Epic Backlog

Bei einem Produkt mit einer voraussichtlich sehr langen Entwicklungszeit gibt es voraussichtlich auch sehr viele Anforderungen, die zum Start des Projekts nicht in der Form ausformuliert werden können, so dass sie den Qualitätsanspruch von guten Product Backlog Items genügen. Wenn ein Feature oder eine Komponente frühesten in einem Jahr von den Entwicklungsteams angegangen werden soll, macht es auch keinen Sinn, das zum Start des Projekts sämtliche User Stories und Abnahmekriterien komplett ausformuliert werden. Derartige langfristigen Ideen, Wünsche oder Anforderungen (sogenannte Epics) an das Produkt könnten in ein zusätzliches Backlog übernommen werden - dem Epic Backlog.

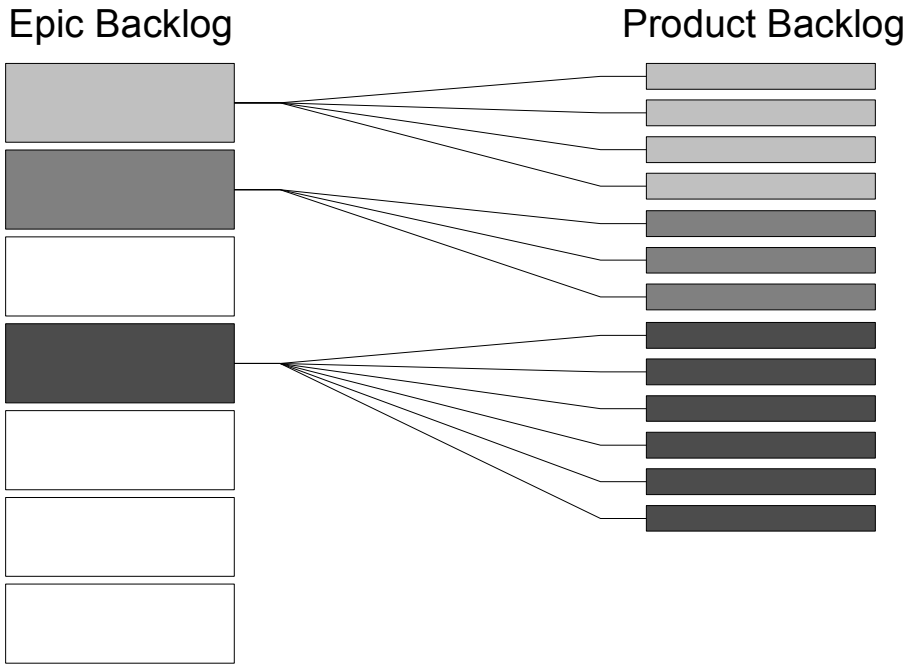

Abbildung 8: Ein Epic Backlog, dass auf das allgemeine Product Backlog heruntergebrochen wurde.

Die Vorteile eines solchen Epic Backlogs sind eine längerfristige Releaseplanung sowie die Möglichkeit eines höheren Abstraktionslevel für Product Backlog Items, die erst in ferner Zukunft implementiert werden sollen. Der Vorteil der Priorisierung aus dem Product Backlog kann ebenfalls auf das Epic Backlog angewandt werden.

Die Gefahr bei einem übergeordnetem Epic Backlog besteht darin, dass der Scrum Prozess etwas von seiner Transparenz einbüßt und ein wenig in das klassische, nicht agile, Projektmanagement abdriftet. So kann ein Epic Backlog von

der Geschäftsleitung (oder anderen Stakeholdern) als die erste Phase des Wasserfallmodells gesehen werden: Der langfristigen Planung inklusive Projektkalkulation, Budgetierung und dem setzen von datierten Meilensteinen. Wenn man dann noch das Epic Backlog von den Entwicklern abschottet so hat man nicht nur viel von der in Scrum geforderten Transparenz verloren sondern auch dem agilen Scrum Prozess in die ersten Ansätze eines Wasserfallmodells eingebettet.

Da der Scrum Guide nur ein *Product Backlog* vorsieht ist der Terminus bei dieser Art Backlogs aufzuteilen sehr wichtig. Es sollte stets von *einem* Product Backlog und einem *übergeordnetem* Epic Backlog die Rede sein. Auf keinen Fall sollte das Epic Backlog als „Epic *Product* Backlog" betitelt werden.

15.1.3 Team Backlogs

Die zweite Möglichkeit Product Backlog Items aufzuteilen ist durch den Product Owner. Die aufgeteilten Product Backlog Items münden also in weitere Backlogs, die den jeweiligen Teams zugeordnet werden. Der Vorteil hierbei ist ein kurzfristiger Zeitvorsprung, da die im vorangegangen Kapitel aufgeführten „Verhandlungen" über Product Backlog Items entfallen und somit wesentlich weniger Personen in diesen Prozess involviert sind.

Ein weiterer Vorteil von mehreren Team Backlogs ist, dass ein gewisses Alignment für die Product Backlog Items erfolgen kann. Abhängigkeiten können so von vorneherein aufgelöst werden, wenn sie einem einzigen Team zur Lösung vorgesetzt werden. Voraussetzung hierfür ist der Einsatz von einem oder mehreren Product Ownern (oder ein anders geartetes Gremium), die diese zumeist sehr technischen Abhängigkeiten erkennen und koordinieren können.

Ein Nachteil dieser Art der Aufteilung von Product Backlog Items ist (wie auch bei einem Epic Backlog), dass die Scrum Prinzipie „Transparenz" darunter leiden könnte. Die Entwicklungsteams bekommen beschnittene Product Backlogs vorgestellt und verlieren so leicht den Blick für das Ganze. Die Bereitstellung der Backlogs für alle Teams über ein Koordinationstool kann hier helfen, ist jedoch sicherlich nicht das gleiche wie eine persönliche Vorstellung.

Des Weiteren kann es Probleme mit der Priorisierung der unterschiedlichen Backlogs zueinander geben. So kann es zum Beispiel vorkommen, dass der am höchsten Priorisierte Eintrag eines Team Backlogs einen niedrigeren Wert aufweist als der eines anderen Team Backlogs.

15.2 Sprint Backlog

Wie in Kapitel 7.2 beschrieben ändert sich das Sprint Backlog eines Teams bei der Skalierung von Scrum nicht. Die Herausforderung besteht in der Synchronisation der Teams welche in Kapitel 13.2 beschrieben wurde. Mögliche Lösungsmöglichkeiten hierzu befinden sich in Kapitel 15.4.

15.3 Produktinkrement

> „*Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.*"
>
> Das Agile Manifest [Bec01]

15.3.1 Kontinuierliche Integration

Bei der kontinuierlichen Integration werden alle Produkt Inkremente kontinuierlich während der Sprints zusammengeführt und sind somit ein Bestandteil der Definition of Done jedes Entwicklungsteams. Nachteilig an der kontinuierlichen Integration ist, dass es zunächst mühsam erscheint sich permanent mit anderen Abteilungen bezüglich der Integration von Team übergreifenden Product Backlog Items abzustimmen. Die Vorteile überwiegen jedoch deutlich:

- Durch die permanente Integration sind Product Backlog Items, die als „Done" deklariert sind auch wirklich „Done". Sie sind entwickelt, getestet und vollständig in das Produkt integriert.
- Auftretende Abhängigkeiten werden sofort angegangen und nicht auf einen unbestimmten späteren Zeitpunkt datiert, was es schwieriger machen würde ein absehbares Release Datum für das Produkt zu bestimmen.
- Durch die ad hoc Auflösung von Integrationsproblemen müssen die Entwickler ihren Fokus weniger häufig wechseln und weniger Probleme „im Hinterkopf behalten"

15.3.2 Merge Sprints

Findet das zusammenführen der Produkt Inkremente der einzelnen Teams nicht permanent und iterativ statt sondern erst gegen Ende des Projekts in explizit dafür vorgesehen Sprints so kann man von „Merge Sprints" sprechen.

Das Zusammenführen der Produkt Inkremente in dieser Form ist während der Projektlaufzeit bequem, da sich jedes Team auf seine eigenen Features oder Komponenten konzentrieren kann und sich nicht permanent mit anderen Entwicklungsteams abstimmen muss um Abhängigkeiten aufzulösen. Die Geschwindigkeit mit der Product Backlog Items erledigt werden wird demnach bis unmittelbar vor Beginn der Merge Sprints höher sein als bei den anderen Möglichkeiten des Zusammenführens von Produktinkrementen. Um ein Product Backlog Item hierbei als „erledigt" kennzeichnen zu können ist es natürlich erforderlich, das die vollständige Integration eines Product Backlog Items *nicht* zur Definiton of Done gehört.

Dieses System birgt jedoch die Gefahr, dass es beim Zusammenführen der Produkt Inkremente in den Merge Sprints zum „großen Knall" kommt und mehr Merge Sprints benötigt werden als ursprünglich vorgesehen. Des Weiteren teilt diese Art des Zusammenführens von Produktinkrementen den Scrum Prozess in Phasen auf, die denen eines Wasserfallmodells ähneln. Dadurch, dass die Definition of Done um den Punkt „Inkrement ist vollständig implementiert" gekürzt wird ist keine regelmäßige Auslieferung der Produkts möglich. Merge Sprints widersprechen damit den Prinzipien des Agilen Manifests (siehe oben).

15.3.3 Hardening Sprints

Eine Kombination aus kontinuierlicher Integration und den Merge Sprints stellen die Hardening Sprints dar. Hierbei wird angestrebt die einzelnen Produkt Inkremente wie bei der kontinuierlichen Integration auch permanent zu integrieren. Man möchte aber auch die Qualität des Produktes weiter anheben, indem man in den Sprints gegen Ende der Produktentwicklung keine weiteren Features mehr entwickelt, sondern ausschließlich Bugs behebt und weitere Tests an der Software durchführt.

Das vermeintliche Anheben der Qualität in extra dafür ausgelegten Sprints birgt aber auch einige Gefahren:

- Die Entwickler wiegen sich während den regulären Sprints in Sicherheit und verschieben das erlangen von Qualität in ihren Inkrementen auf die Hardening Sprints.
- Es ist schwer schätzbar wie viel Zeit für die Hardening Sprints benötigt wird und es kann zu Verzögerungen bei Produktauslieferung „auf den letzten Metern" kommen

Auch diese Art Produktinkremente zusammenzuführen lässt den Scrum Prozess ein Stück in Richtung klassisches Projektmanagement abdriften und widerspricht demnach ebenfalls dem agilen Manifest.

15.4 Sprints und Sprint Ziel

> „*Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.*"
>
> Das Agile Manifest [Bec01]

15.4.1 Release-Kickoff

Der Cloud Computing Anbieter salesforce.com setzt bei seiner Skalierung von Scrum auf Release-Kickoff's und sogenannte Release-Open-Space's (siehe auch Kapitel 17.2.1). Die Entwicklungsteams sind zunächst angehalten einen groben Plan dessen zu erstellen was sie in den nächsten drei bis vier Monaten liefern möchten. Daraufhin stellen die Product Owner der Teams bei einem Release Kickoff diesen Plan den anderen Teams vor. Wenige Tage später findet das Release-Open-Space statt. Die Entwicklungsteams senden zu jedem Release-Open-Space mindestens einen Entwickler. Das Release-Open-Space beginnt damit, dass die Teilnehmer wichtige Themen und Abhängigkeiten nennen die es zu klären gilt. Daraufhin bilden die Teilnehmer des Release-Open-Space temporäre Gruppen indem sie sich 45 Minuten den vorgestellten Themen widmen. Im Anschluss an diese 45 Minuten stellt ein ausgesandter jeder Gruppe kurz das Arbeitsergebnis der eigenen Gruppe den anderen Gruppen gesammelt vor. Dieser dritte Teil des Release-Open-Space sollte laut salesforce.com nicht länger als 30 Minuten andauern. Salesforce wiederholt die Teile zwei und drei des Release-Open-Space so lange wie noch Interesse an den verbleibenden Themen besteht.

15.4.2 Asynchrone Sprints

Müssen bestimmte Personen (z.B. Chief Product Owner, Chefarchitekten, usw.) bei allen Planning und Review Meetings anwesend sein so empfiehlt es sich die Sprints aller Teams asynchron zueinander starten zu lassen. Der Vorteil von hintereinander startenden Sprints ist, dass es für die folgenden Teams leicht ersichtlich ist welche Product Backlog Items die vorangegangen Teams bis zum Ende des gestarteten Sprints abschließen möchten.

Diese Art Sprints zu planen birgt aber auch einige Nachteile. Das gravierendste Manko ist wohl, dass es für die anderen Teams schwierig ist ein Team, welches bereits mit der Arbeit begonnen hat, noch dazu zu bewegen ein Product Backlog Item zu tauschen um eine Abhängigkeit aufzulösen. Dieser Umstand macht eine gute Vorausplanung der Sprints erforderlich.

Wird der Start der Sprints um den minimalen Zeitraum von einem Tag versetzt[19], so ist diese Art der Synchronisation von Teams bis zu einer Größe von neun Teams möglich. Ansonsten würde laut Mike Cohn [Coh10, S. 376] für die Planung eine zu große Verzögerung durch Meetings entstehen. Viele zeitversetzt startende Sprints sind auch für die Personen unangenehm, welche an allen Meetings anwesend sein sollten da sie in diesem Fall einen regelrechten Meeting-Marathon abhalten müssten.

Des Weiteren ist es bei asynchronen Sprints schwierig, dem Endkunden während der Projektlaufzeit ein komplettes Produkt zur Verfügung zu stellen, da nie alle Teams gleichzeitig mit einem vollständig integriertem Produktinkrement fertig sind.

15.4.3 Zeitlich abgestimmte asynchrone Sprints

Ein Nachteil von asynchronen Sprints ist, wie erwähnt, dass es schwierig ist dem Endkunden ein Produktinkrement zur Verfügung zu stellen um Feedback einzuholen. Diesem Umstand kann man mit zeitlich abgestimmten asynchronen Sprints entgegen wirken. Bei drei Teams kann man so beispielsweise dem Team A zweiwöchige Sprints, Team B dreiwöchige Sprints und Team C vierwöchige Sprints zuteilen. In diesem Fall gibt es ein aktuelles und vorzeigbares Produkt alle 12 Wochen, da zu diesem Zeitpunkt jedes Team mit einem Sprint fertig ist.

Der Nachteil von zeitlich abgestimmten asynchronen Sprints ist (ergänzend zu den Nachteilen von komplett asynchronen Sprints), dass die Planungs- und Review Meetings der Teams zyklisch auf einen gemeinsamen Tag fallen.

[19] Da die Timebox der Sprint Planning Meetings 1 und 2 mindestens einen Tag füllt

15.5 Definition of Done

> *„Funktionierende Software ist das wichtigste Fortschrittsmaß."*
> Das Agile Manifest [Bec01]

In Kapitel 7.5 wurde beschrieben, dass die Regeln aus dem Scrum Guide zu ungleichen Definiton of Done's der Teams führen. Um die Qualität des Produkts auf ein einheitliches, mögliches hohes Maß zu bekommen ist eine Annäherung der Definition of Done's aller Teams erforderlich. Dies kann mit einer gestaffelten Definition of Done erfolgen. Für den Einsatz einer gestaffelten Definition of Done (DoD) sind mehrere Release-Stufen des Produkts erforderlich. Diese können zum Beispiel ein internes Release sowie ein Release in die Produktion sein. Vor diesen Releases gibt es die Abarbeitung einer User Story sowie die Summe aller erledigten User Stories pro Sprint als jeweils eine Release-Stufe. Die gestaffelte Definition of Done sieht nun für jede Release-Stufe eine eigene DoD vor. Diese könnten exemplarisch wie folgt gestaffelt sein:

1. Eine Definition of Done für das erledigen einer User Story. Diese DoD gleicht dem einer DoD aus kleinen Projekten mit einem Team ohne verknüpfte User Stories.

2. Die DoD aus 1 sowie eine zusätzliche DoD für die Summe aller User Stories, die im Sprint erledigt wurden. Diese zusätzliche DoD kann z.B. das durchführen von Tests hinsichtlich der Kompatibilität der Software zu den User Stories untereinander sein.

3. Die DoD aus 2 sowie eine zusätzliche DoD für das interne Release der Software. Diese sollte z.B. zusätzliche Tests hinsichtlich der Kompatibilität der Produkt Inkremente aller Teams zueinander beinhalten. Diese dritte Stufe der DoD wird (im Gegensatz zur ersten und zweiten Stufe) von allen Teams gemeinschaftlich erarbeitet und unterzeichnet.

4. Die DoD aus 3 sowie eine zusätzliche DoD für das Release der Software zum Endkunden. Die in diesem Beispiel letzte Stufe der DoD wird ebenfalls von allen Teams gemeinschaftlich erarbeitet und unterzeichnet. Sie könnte jetzt z.B. zusätzlich den Lasttest der gesamten Anwendung beinhalten.

Um den gemeinschaftlich erstellten DoD's zu genügen und diesen mit möglichst wenig offenen Punkten gegenüberzutreten, müssen die einzelnen Teams bereits auf „lokaler Ebene" auf die gemeinsamen DoD's hinarbeiten. Hierdurch näheren sich die Teams bis zum endgültigem Release der Software einem gemeinsamen Level an Qualität an.

16 Scrum Rollen

16.1 Product Owner

> „*Heisse Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.*"
>
> Das Agile Manifest [Bec01]

16.1.1 Kaskadierte Product Owner

Das Konzept eines sogenannten Super Product Owner (oder „Chief Product Owner") verfolgt den Gedanken, dass wie im „kleinen Scrum" auch eine Person die finale Entscheidungsgewalt über die Einträge und Priorität der Product Backlog Items im Product Backlog besitzt. Da eine Person alleine die Fülle an Anforderungen und Wünsche aller Stakeholder nicht mehr vollständig erfassen und priorisieren kann wird die Hilfe weiterer Product Owner nötig. Dies können entweder die Product Owner der Scrum Teams sein oder weitere, explizit für diese Aufgaben zuständige dedizierte Product Owner. Diese Kaskadierung der Product Owner kann nach bedarf um weitere Ebenen ergänzt werden. Wird beispielsweise eine Multimedia Suite entwickelt so kann die Aufteilung der Product Owner wie folgt illustriert in drei Ebenen erfolgen:

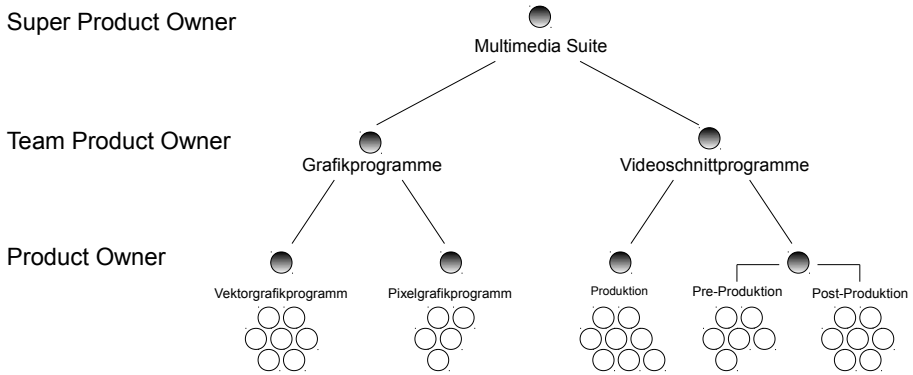

Abbildung 9: Skalierung der Product Owner. In Anlehnung an [Coh10, Abbildung 17.1]

Es gibt Product Owner (gefärbte Kreise), die als reguläre Product Owner für ein bis zwei Teams (weiße Kreise) fungieren. Die Ausrichtung der beiden Sparten Grafik- und Videoschnittprogramme erfolgt über die Team Product Owner. Die mittel- und langfristige Produktstrategie obliegt dem Super Product Owner. Der Super Product Owner ist weiterhin für die umfassende Vision des Produkts und die Vermittlung dieser verantwortlich.

Da die Arbeit eines Product Owners bereits im kleinen ein ganztags Job ist, ist es offensichtlich, dass ein Super- oder Team Product Owner keine praktischen Aufgaben für die Entwicklungsteams übernehmen kann. Dies bedeutet

aber nicht, dass die Product Owner ohne Entwicklungsteam feste Grenzen in ihrer Arbeit haben. Sie sollten auch weiterhin angehalten sein nach Möglichkeit für die Entwickler zur Verfügung zu stehen - denn letzten Endes „*sind alle Product Owner für das gesamte Produkt verantwortlich und müssen dieses Gefühl der gemeinsamen Verantwortung in ihren Teams wecken.*" [Coh10, S. 359]

Die Vorteile in der Kaskadierung der Product Owner liegen in der klaren Verantwortlichkeit für Product Backlog Items sowie in dem aus der Informatik bekannten reduktionistischen Lösungsansatz „Teile und herrsche" (*divide and conquer*). Hierbei werden die großen Probleme so lange in kleinere zerlegt, bis man sie lösen („beherrschen") kann. Die Analogie von „großen Problemen" liegt bei skalierten Scrum Projekten in einem großen Product Backlog welches so auf kleinere Backlogs heruntergebrochen wird. Der Nachteil ist, dass ein Product Owner auf unterster Ebene keinen direkten Einblick in das Backlog eines Kollegen einer anderen Sparte hat. Im Beispiel aus Abbildung 9 z.B. der Product Owner des Vektorgrafikprogramms zu dem der Post-Produktion aus der Videoschnittsparte. Die Scrum Prinzipie „Transparenz" leidet demnach darunter.

16.1.2 Product Owner Gremium

Kommt man vom klassischen Projektmanagement so kann man versucht sein, dem bisher als Lenkungsausschuss betitelten Gremium an Projektleitern einen neuen Namen zu geben und „Scrum-konform" in „Product Owner Gremium" umzubenennen. Ein Gremium an Product Ownern, in dem alle Product Owner gleichberechtigt sind, widerspricht jedoch der folgenden Passage aus dem Scrum Guide [SS11]:

> „*The Product Owner is one person, not a committee. The Product Owner may represent the desires of a committee in the Product Backlog, but those wanting to change a backlog item´s priority must convince the Product Owner. For the Product Owner to succeed, the entire organization must respect his or her decisions.*"

In einem Gremium an Entscheidungsträgern, ganz gleich wie geartet, muss es stets einen Super Product Owner (siehe vorheriges Kapitel) geben, der die finale Entscheidungsgewalt über die Einträge und die Priorisierung der Product Backlog Einträge besitzt. Vor diesem Hintergrund werden die Vor- und Nachteile eines Product Owner Gremiums ohne einen Super Product Owner in dieser Arbeit nicht weiter verfolgt.

16.1.3 Product Owner Gremium mit Super Product Owner

Ein reines Product Owner Gremium ohne eine klare Führungsposition bezüglich der Verantwortlichkeit über das Product Backlog ist aus den im vorherigen Kapitel genannten Gründen keine erstrebenswerte Lösung. Eine Kombination eines Product Owner Gremiums mit einem Super Product Owner (siehe Kapitel 16.1.1) ist jedoch durchaus denkbar. Die vielschichtigen Aufgaben eines einzelnen Product Owner könnten so auf das Gremium verteilt werden und in einem gemeinsamen Meeting zusammengetragen werden. Der Super Product Owner

profitiert von den Vorteilen eines Teams und behält weiterhin die finale Entscheidungsgewalt über den Inhalt des Product Backlogs sowie der Priorisierung der Product Backlog Items.

Problematisch wird die Skalierung mit Hilfe eines Gremiums, wenn die Mitglieder des Gremiums sich aufgrund geographischer Gegebenheiten nicht regelmäßig persönlich treffen können. Werden die Meetings dann über minderwertige Kommunikationsmittel abgehalten, so können wichtige Informationen verloren gehen, die dem Wert des Produkts schaden können.

16.2 Scrum Master

> „*Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.*"
>
> Das Agile Manifest [Bec01]

Im Idealfall sollte in einem Projekt mit vielen und verteilten Teams jedes einzelne seinen eigenen Scrum Master besitzen. Nur so kann sichergestellt werden, dass jedes Team zu jederzeit die optimale Unterstützung erhält sofern es auf Probleme bei der Entwicklung stößt, die es nicht eigenständig lösen kann.

Um Probleme zu adressieren, die mehrere Teams betreffen könnten oder eine Kooperation mehrerer Scrum Master erfordern ist eine regelmäßiges Treffen der Scrum Master nötig. Dieses Treffen sollte täglich stattfinden, so dass Probleme immer sofort angegangen werden können. Der erste Teil des Scrum of Scrum Masters Meeting kann dem eines Daily Scrums folgen indem jeder Scrum Master drei Fragen beantwortet:

- Welche Probleme wurden seit dem letzten Scrum of Scrum Masters Meeting gelöst?
- Welche Probleme bestehen weiterhin und was wird unternommen um diese zu lösen?
- Gibt es neue Probleme denen mein Team gegenübersteht? Wenn ja, könnte dies Auswirkungen auf die Arbeit anderer Teams haben?

Nach diesem kurzen ersten Teil kann dazu übergegangen werden die angesprochenen Probleme zu bearbeiten. Dabei sollten zuerst die brennendsten oder die Themen behandelt werden, die viele Teams betreffen. Zur Priorisierung kann ein Scrum Master Backlog verwendet werden welches entweder gemeinschaftlich oder von einem Super Scrum Master gepflegt wird.

16.3 Entwicklungsteam

> *„Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.“*
>
> Das Agile Manifest [Bec01]

16.3.1 Horizontale Aufteilung

Bei der horizontalen Aufteilung von Teams spricht man von sogenannten „Layer-”, „Schichten-” oder „Komponententeams”. Hierbei ist jedes Team für eine bestimmte Abstraktionsschicht innerhalb der Anwendung verantwortlich. In einem Webprojekt können dies z.B. die Grafische Benutzeroberfläche, die Geschäftslogik und die Datenbank sein, welche jeweils durch ein Komponententeam bearbeitet werden. Ein Vorteil dieser Art der Aufteilung von Teams besteht in der klaren Verantwortlichkeit für bestimmte Aufgaben. Die Teams bestehen des Weiteren ausschließlich aus Spezialisten auf ihrem Fachgebiet was auch für eine gewisse Integrität und Konsistenz im Code der jeweiligen Komponente sorgt.

Leider birgt der Einsatz von Komponententeams auch viele Nachteile. Da jedes Team für sich gekapselt arbeiten kann wird es dies aus Gründen der Bequemlichkeit auch tun und die Kommunikation zwischen den Teams wird darunter leiden. Eine Arbeitshaltung „streng nach Vertrag” kann die Folge sein, was wiederum der Projektmanagement-Philosophie eines Wasserfallmodells gleicht. Darüber hinaus ist es in einem homogenem Umfeld an Komponententeams nur sehr schwierig möglich ein integriertes und über alle Schichten der Anwendung hinweg getestetes Feature als vollständiges Produktinkrement auszuliefern.

Der Einsatz von Komponententeam empfiehlt sich daher nur entweder in Mischform mit Featureteams oder temporär. Näheres hierzu befindet sich in Kapitel 16.3.3.

16.3.2 Vertikale Aufteilung

Bei der vertikalen Aufteilung von Teams spricht man von „Value-” oder „Featureteams”. Hierbei sind die Teams für eine komplette Endanwenderanforderung zuständig und somit für eine ganzheitliche Integration in das Produkt. Sie bearbeiten hierbei sämtlich Schichten der Anwendung und müssen somit aus interdisziplinären Entwicklern bestehen. Der Vorteil von interdisziplinären Entwicklern ist zugleich ein Nachteil der Featureteams: Es gibt keine Spezialisten. Dies kann zu einer inkonsistenten Architektur und zu einer mangelhaften Integrität des Produkts führen.

Dennoch überwiegen die weiteren Vorteile, Teams auf diese Art aufzuteilen:

- Dadurch, dass sich die Entwickler in allen Anwendungsschichten bewegen, können bessere Design- und Architekturentscheidungen getroffen werden. Das Schätzen von Anforderungen in den Planungsmeetings wird zusätzlich vereinfacht.
- Die Übergabe von Anforderungen an andere Teams und der hierdurch entstehende Waste (siehe Kapitel 7.1) wird reduziert.

- Die Kommunikation innerhalb des Teams wird erhöht, da sich Wissen besser verteilt.
- Der Planungsaufwand ist geringer als bei Komponententeams.
- Wie der Name „Featureteam" schon sagt konzentrieren sich Featureteams auf das erledigen von Aufgaben die direkt dem Kunden dienen.

In Summe sollte eine Reihe von homogenen Featureteams wenig Probleme haben regelmäßig ein vollständig integriertes und getestetes Produktinkrement auszuliefern.

16.3.3 Gemischte Aufteilung

Eine weitere Möglichkeit der Aufteilung von Teams ist eine Mischform aus den zuvor genannten. Der Einsatz von Komponententeams zusätzlich zu den Featureteams empfiehlt sich z.B. wenn bestimmte Feature- oder Framework Komponenten von mehreren Featureteams benötigt werden. In diesem Fall können Komponententeams gebildet werden, die ihre Mitglieder aus den Featureteams abziehen um die entsprechenden Bausteine für die anfragenden Featureteams zu entwickeln. Die Featureteams können hierbei den „Product Owner" für die Komponententeams darstellen und sind somit für die Pflege des Product Backlogs für die Komponententeams sowie die Abnahme der geforderten Features verantwortlich. Sie stehen wie ein regulärer Product Owner den Komponententeams für Fragen zur Verfügung und geben entsprechendes Feedback bei den Sprint Review Meetings der Komponententeams.

Weiterhin besteht die Möglichkeit Komponententeams nur temporär einzurichten und nach Abschluss der Entwicklungsarbeit die abgezogenen Mitarbeiter wieder in die Featureteams zu entlassen. Handelt es sich ausschließlich um einen temporären Abzug so hat dies zwei gravierende Vorteile:

1. Die Mitarbeiter aus den Featureteams bringen reichlich Domänenwissen in die Komponententeams mit ein. Hierdurch ist ein Gesamtüberblick über die Anwendung vorhanden und ein großer Nachteil von Komponententeams aufgehoben.
2. Die abgezogenen Mitarbeiter wissen genau zu welchem Zweck das Komponententeam gebildet wurde und arbeiten sich praktisch selber zu. Die Motivation der Mitarbeiter ist dementsprechend hoch eine qualitativ hochwertige Komponente zu erstellen, die von ihm, seinen Team und weiteren Teams genutzt werden kann.

Das Conway'sche Gesetz [Con68] besagt, dass ...

> „*Any organization that designs a system (defined more broadly here than just information systems) will inevitably produce a design whose structure is a copy of the organization's communication structure.*"

Dies bedeutet auch, das häufig Teamstrukturen gebildet werden, die der Kommunikationsstruktur des Unternehmens gleichen jedoch nicht zwangsläufig optimal sind. Der eingangs zu diesem Abschnitt erwähnten Auszug aus dem Agilen Manifest besagt zudem, dass es selbstorganisierte Teams sind, die die beste Software liefern. Die Teamaufteilung sollte demnach in Kooperation mit den Verantwortlichen, erfahrenen Scrum Mastern sowie mit den Entwicklern selbst erfolgen um eine möglichst ideale Balance zwischen Feature- und Komponententeams zu finden.

16.3.4 Mitglieder in mehreren Teams

Hat man sich für die Mischform von horizontalen Komponententeams und vertikalen Featureteams entschieden, so gibt es nun die Möglichkeit bestimmte Entwickler in mehreren Teams gleichzeitig mitwirken zu lassen.

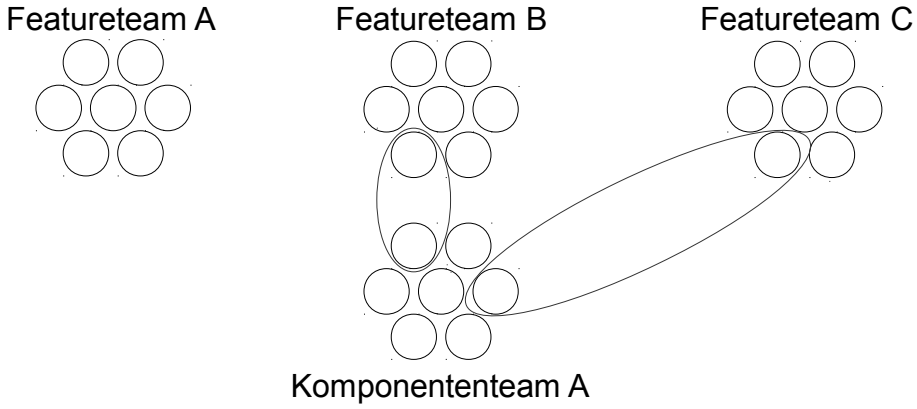

Abbildung 10: Entwickler sind Mitglied in mehreren Entwicklungsteams. In Anlehnung an [Coh10, Abbildung 17.4]

Bei einer homogenen Teamverteilung (nur horizontal oder nur vertikal) kann dieses Prinzip zwar ebenfalls angewandt werden, ist jedoch laut Cohn [Coh10, S. 367] nicht so effektiv. Nachteilig wäre z.B. der häufige Fokuswechsel der betroffenen Entwickler und das sie mit mehr Personen als der idealen Anzahl (siehe Kapitel 8.3) interagieren müssen.

Die ausgewählten Entwickler für zwei Teams sollten an beiden Seiten einer Abhängigkeit in der Entwicklung involviert sein um diese möglichst effektiv auflösen zu können. Selbstverständlich können diese „zweigeteilten Entwickler" nun nicht mehr als vollwertige Arbeitskraft eines Teams gezählt werden, was entsprechend im Sprint Planning Meeting 1 berücksichtigt werden muss. Dies ist in der Praxis jedoch nur schwer durchsetzbar. Ist ein Entwickler erst mal in mehr als einem Team, so möchte auch jedes Team von der neu gewonnenen Arbeitskraft profitieren. Es werden dem Entwickler zusätzlich zu seiner Aufgabe, die Abhängigkeiten zwischen dem Komponenten und dem Featureteam aufzulösen weitere kleinere Aufgaben zugesprochen oder der Entwickler anderweitig in das Tagegeschäft der Teams involviert. Der zweigeteilte Entwickler wird so mit mehr

Aufgaben betraut, wie es nach den Erkenntnissen von Clark und Wheelright in „Managing New Product and Process Development" [Whe10] sinnvoll ist:

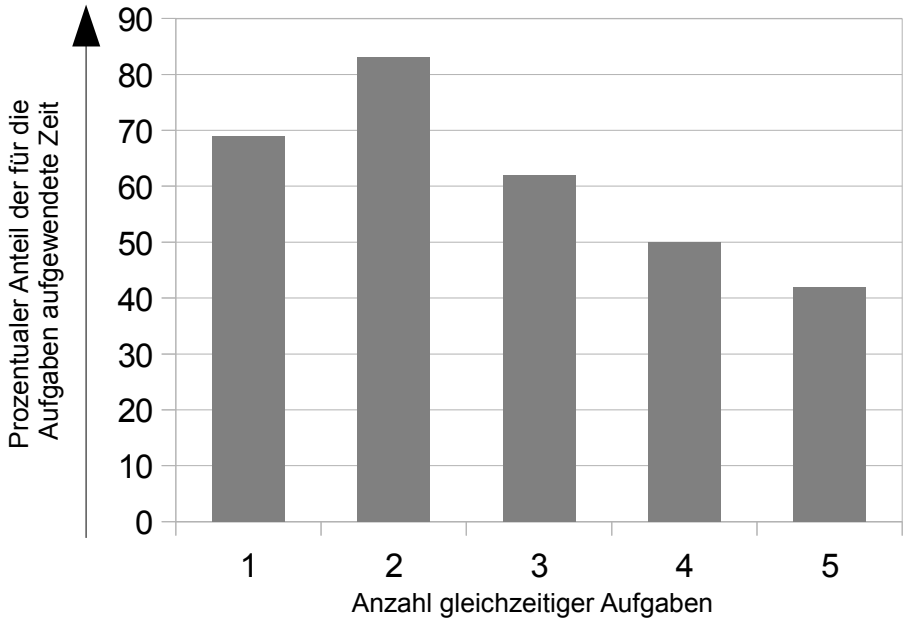

Abbildung 11: Weniger als drei gleichzeitige Aufgaben werden am effektivsten abgearbeitet [Whe10, S. 242]

Die Abbildung zeigt, dass von einer Person eine oder maximal zwei gleichzeitige Aufgaben mit dem prozentual geringstem Zeitaufwand bewältigt werden können. Ab einer Anzahl von drei gleichzeitigen Aufgaben sinkt die Effektivität stetig ab. Der Einsatz von Mitgliedern in mehreren Teams sollte daher nur mit Bedacht gewählt werden.

16.3.5 Integrationsteams

Die gravierendsten Lücken bei großen Projekten befinden sich laut Sosa, Eppinger und Roles [MSR07, S. 133-142] an den Schnittstellen zwischen den Teams. Um diese Schnittstellen zu identifizieren und Probleme mit Abhängigkeiten diesbezüglich zu lösen, kann man zusätzlich zu den regulären Entwicklungsteams ein oder mehrere Integrationsteams einführen. Diese übernehmen die Aufgabe nicht erkannte oder vernachlässigte Schnittstellen aufzuspüren und entweder Verantwortlichkeiten für diese zuzuteilen oder diese selbst aufzulösen.

Je nach Größe des Projekts kann man sich für drei Arten von Integrationsteams entscheiden:

1. Bis zu einer Größe von zwölf Teams empfiehlt Mike Cohn [Coh10, S. 368] den Einsatz von virtuellen Integrationsteams. Wie bei den im vorangegangen Kapitel erwähnten Mitgliedern, die sich in mehreren Teams befinden, so bestehen auch virtuelle Integrationsteams aus Mitgliedern die sich zeitgleich in mehreren Teams befinden.

2. Die zweite Möglichkeit ist der Einsatz von festen Integrationsteams. Diese empfehlen sich ab einer Projektgröße von 12 Teams und besitzen dedizierte Mitglieder über die gesamte Projektlaufzeit [Coh10, S. 368].

3. Die dritte Möglichkeit des Einsatzes von Integrationsteams ist unabhängig von der Anzahl an vorhandenen Teams und besteht aus einer Mischform der zuvor genannten beiden Möglichkeiten. Das Integrationsteam besteht zunächst aus einer fixen Anzahl an dedizierten Mitarbeitern welche sich um Aufgaben bei der Projektinitiierung kümmert. Dies kann z.B. das Aufsetzen von Servern, Versionskontrollsystemen, Wikis, sowie das Aufsetzen von Tests für Schnittstellen etc. sein. Ist ein Großteil der einmalig stattfindenden Aufgaben abgeschlossen, so kann sich das Integrationsteam nach Bedarf in ein virtuelles Integrationsteam umwandeln bei dem die ehemaligen Vollzeitmitglieder des Integrationsteams jetzt zusätzlich in den Entwicklungsteams mitarbeiten.

Unabhängig davon für welche Art von Integrationsteams man sich entscheidet, so ist es für die Mitglieder der Integrationsteams zwingend erforderlich an möglichst vielen Meetings aller Entwicklungsteams teilzunehmen. Nur hierdurch erhalten die Mitglieder der Integrationsteams einen Überblick über das Gesamtsystem und können bisher nicht erkannte oder vernachlässigte Schnittstellen identifizieren.

17 Scrum Meetings

17.1 Sprint Planning Meeting 1

> „*Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteam zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.*"
>
> Das Agile Manifest [Bec01]

17.1.1 Laufende Vorausplanung

Die simpelste, ad hoc Möglichkeit Teams zu synchronisieren besteht darin, dass es den Entwicklungsteams überlassen wird sich im Sprint Planning Meeting 1 Gedanken über den Inhalt zukünftiger Sprints zu machen. Hierbei wird zunächst wie gewohnt der aktuelle Sprint geplant, die User Stories in Tasks unterteilt und mit einem Schätzwert (Story Points, Stunden, etc.) versehen. Im Anschluss daran macht sich das Entwicklungsteam Gedanken was es in den folgenden beiden Sprints angehen möchte. Dieser zusätzliche Teil im Sprint Planning Meeting sollte im Idealfall (das Product Backlog ist synchronisiert und das Team kennt seine

Entwicklungsgeschwindigkeit) nicht länger als 10 Minuten andauern [Coh10, S. 364].

Diese Vorausplanung muss dann in einem teamübergreifenden Meeting von jedem Team allen anderen Teams vorgestellt werden. So kann sichergestellt werden, dass eine Komponente, welche Team A von Team B in Sprint *n* benötigt wird von Team B in Sprint *n-1* bearbeitet wird. Diese Vorstellung als ein weiteres Meeting nimmt zusätzlich Zeit in Anspruch wird jedoch mit einer besseren Koordination der Sprint Backlogs belohnt.

17.1.2 Großraumverfahren

Beim Großraumverfahren befinden sich alle Teams in einem Raum, der groß genug ist, alle Teams gleichzeitig aufzunehmen. Zunächst wird durch den Product Owner die Produktvision vorgestellt und was in den nächsten ein bis drei Sprints erreicht werden sollte. Daraufhin ziehen sich die Teams in Ecken des Raumes zurück und halten das reguläre Sprint Planning Meeting 1 für sich ab. Fällt bei diesen Sprint Planning Meetings auf, dass eine Abhängigkeit zu einem der anderen Teams besteht, so wird ein Teammitglied ausgesandt um zu dem betroffenen Team zu gehen und das Thema anzusprechen.

Ein weiterer Vorteil dieser Methode ist, dass ein Super Product Owner oder Chefarchitekt abwechselnd allen Teams für Fragen zur Verfügung stehen kann. Hierzu werden die benötigten Personen von den anfragenden Teams einfach herbeigerufen oder abgeholt.

Nachteilig bei einem Verfahren bei dem sich viele Leute in einem Raum befinden ist sicherlich der aufkommende Lärmpegel. Um diesen etwas einzudämmen können z.B. Signalschilder eingeführt werden die besagen, dass man eine Frage an einen Product Owner, einen Architekten oder Team X hat. Die betroffenen Personen können so nach ihrem aktuellen Gesprächen einfach zum nächsten Team wechseln ohne das bei einer Anfrage der Geräuschpegel angehoben wird.

Voraussetzung für dieses Verfahren ist, dass eine Product Owner Hierarchie (ganz gleich wie geartet) die Produktvision ganzheitlich durchdrungen hat und diese entsprechend in die Teams tragen kann.

17.2 Sprint Planning Meeting 2

17.2.1 Sprint-Open-Space

Bei einem „Sprint-Open-Space" treffen sich die Teams, um nach ihrer initialen Planung durch Diskussionsgruppen, Abhängigkeiten und weitere wichtige Themen zu besprechen und Lösungsmöglichkeiten zu erarbeiten. Ein Sprint-Open-Space folgt den Prinzipien der Open-Space[20] Vorgehensweise bei Besprechungen. Hierbei schreiben die Teilnehmer zunächst wichtige Themen für den Sprint auf große Blätter und hängen diese im Raum auf. Daraufhin bilden sich zu den Themen Gruppen, die über das Thema diskutieren. Im Anschluss an die

[20] Detaillierte Informationen zur Open-Space Methode sind abrufbar unter http://www.openspaceworld.org/german/index.html

Diskussionen findet eine Nachbesprechung statt, bei der die Ergebnisse der Diskussion allen Teilnehmern des Sprint-Open-Space vorgestellt wird. Diese letzten beiden Teile des Sprint-Open-Space sollten Timeboxed sein und so lange iteriert werden, wie Interesse an den Themen besteht. Es empfiehlt sich dennoch auch das komplette Meeting zeitlich zu limitieren, um nicht in endlose Diskussionen zu verfallen.

Ein Vorteile der Sprint-Open-Space Methode ist demnach, dass sich viele Menschen engagiert mit unterschiedlichen Themen befassen und zu vielseitigen und häufig nachhaltigen Lösungen kommen. Des Weiteren wirken Open-Space's immer gemeinschaftsbildend. Dadurch, dass bei den unterschiedlichen Themen meist verschiedene Leute zusammenkommen, lernt man sich auch teamübergreifend näher kennen.

Jedoch birgt die Open-Space-Methode im Umfeld von Scrum auch ein Risiko: Wenn die erarbeiteten Lösungsmöglichkeiten zu oft durch höhere Instanzen (z.B. aufgrund von mangelnden Ressourcen oder Unternehmensrichtlinien) verworfen werden, kann dies zu einer nachhaltigen Störung der Veranstaltung führen. Die Teilnehmer werden demotiviert und äußern sich nur noch zurückhaltend zu den angebotenen Themen, was zu weniger konstruktiven und qualitativ minderwertigen Lösungsvorschlägen führt.

17.2.2 Last responsible moment

Der Zeitpunkt um Lösungsmöglichkeiten für schwierige Themen oder dem Auflösen von Abhängigkeiten zu finden kann auch auf die Daily Scrum of Scrums (siehe folgendes Kapitel 17.3) verlegt werden. Dieses Vorgehensweise folgt dem Prinzip des „*last responsible moment*" [PP03, S. 57] aus der Lean Production. Aufkommende Schwierigkeiten werden hierbei im Sprint Planning Meeting 2 Teamintern angesprochen und notiert, jedoch erst im Scrum of Scrums den betroffenen Teams mitgeteilt, sobald das Problem akut wird. Auf diese Weise wird sichergestellt, dass Lösungen gefunden werden, die auf dem aktuellsten Stand der Software basieren. Zudem können sich die Teams bis zuletzt den Aufgaben widmen, die sie ohne Hilfe fremder Teams bearbeiten, können was zu weniger Kontextwechseln während der Arbeit führt.

Nachteilig bei dieser Art Lösungsmöglichkeiten für schwierige Themen oder dem Auflösen von Abhängigkeiten zu finden ist, dass die Teams nicht wissen wie viel Zeit sie für das Auflösen von Abhängigkeiten einplanen müssen. Durch die Möglichkeit, Teams ad hoc um Hilfe bei derartigen Aufgaben zu bitten, muss also ein Puffer in die Zeitplanung aller Teams eingebaut werden. Dieser sollte hinreichend groß dimensioniert werden (um die für den Sprint zugesagten zu erledigenden Aufgaben noch bewältigen zu können) und auf Erfahrungswerten aus den bisherigen Sprints basieren.

17.3 Daily Scrum

> „*Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten*"
>
> Das Agile Manifest [Bec01]

17.3.1 Scrum of Scrums (kurz)

Das Auflösen von Abhängigkeiten von verteilten interdisziplinären Entwicklungsteam kann durch das in [Pic07, S. 125ff] beschriebene Scrum of Scrums erfolgen. Beim Scrum of Scrums (oder „Meta-Scrum" oder „Integration Daily Scrum" [SI08, S. 162]) trifft sich jeweils ein Teammitglied jedes Teams unmittelbar nach dem Daily Scrum mit allen ausgesandten aus den anderen Entwicklungsteams. Das Scrum of Scrums hat den gleichen Inhalt, Umfang und Ziele wie das Daily Scrum (siehe Kapitel 17.3 und [SS11, S. 10]) jedoch werden die Fragen auf Team-Ebene beantwortet:

- Was hat mein Team seit dem letzten Scrum of Scrums getan?
- Was möchte mein Team bis zum nächsten Scrum of Scrums erledigen?
- Welche Hindernisse liegen meinem Team im Weg oder verlangsamen die Arbeit meines Teams?

Eine weitere Frage ergänzt die aus dem Daily Scrum bekannten Drei und adressiert vor allem das Thema Synchronisation von Teams (siehe Kapitel 15.4):

- Wird mein Team einem anderen Team ein Hindernis in den Weg legen oder wurde eine neue Abhängigkeit identifiziert?

Wie auch im Daily Scrum sollen Probleme hier nicht gelöst[21] sondern lediglich angesprochen werden, da dies sonst die Timebox (15 Minuten) des Scrum of Scrums sprengen würde. Die Auflösung erfolgt individuell mit den involvierten Mitgliedern aus den unterschiedlichen Entwicklungsteams.

Beim Scrum of Scrums sollte beachtet werden, dass immer unterschiedliche Mitglieder aus den Entwicklungsteams zum Scrum of Scrums ausgesandt werden. Dies erhält die Gleichstellung aller Entwickler im Team und fördert die Kommunikation aller Teammitglieder aller Teams untereinander (jeder lernt jeden mal kennen).

Jedoch gilt auch beim Scrum of Scrums die Miller Regel [Mil56] (siehe Kapitel 16.3) und somit verliert auch das Scrum of Scrums ab einer Höchstgrenze von neun Teams an Effektivität. Um das Scrum of Scrum noch weiter zu skalieren gibt es die Möglichkeit ein weiteres Scrum of Scrums einzuführen. Sozusagen ein „Scrum of Scrums of Scrums" oder „Meta-Meta-Scrum". Die folgende Abbildung illustriert dieses Art der Skalierung des Daily Scrums:

[21] Wenn das Problem mit ein, zwei Sätzen gelöst werden kann ist dies selbstverständlich gestattet

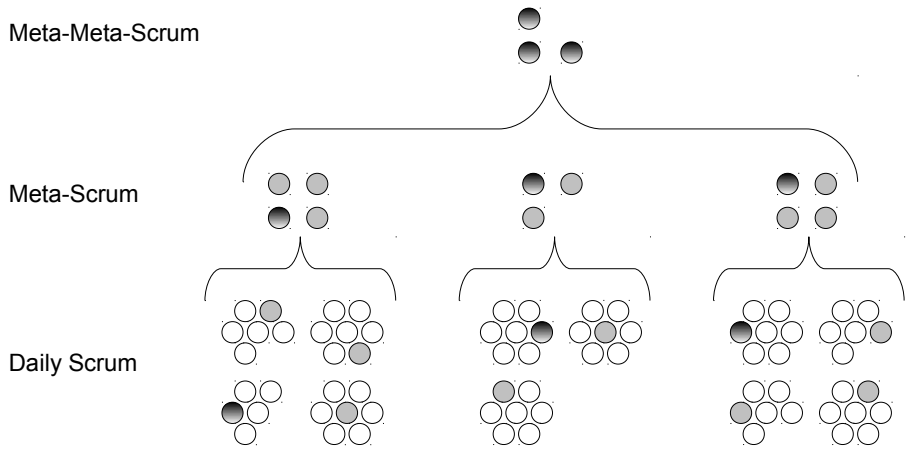

Abbildung 12: Skalierung des Daily Scrums. In Anlehnung an [Coh10, Abbildung 17.5]

17.3.2 Scrum of Scrums (lang)

Neben der kurzen Version des Scrum of Scrums gibt es noch eine von Mike Cohn [Coh10, S. 371ff] herausgearbeitete lange Version des Scrum of Scrums. Das lange Scrum of Scrums sollte nach Mike Cohn nicht täglich sondern etwa zwei bis drei mal die Woche stattfinden und hat eine Timebox von maximal einer Stunde. Im Gegensatz zum kurzen Scrum of Scrums können in dieser Timebox also Probleme nicht nur angesprochen sondern nach Möglichkeit auch gelöst werden. Da auch diese Timebox bei vielen Teams manchmal nicht ausreicht, empfiehlt es sich ein „Scrum of Scrums Problem Backlog" einzuführen. Dieses Backlog enthält Probleme (oder Abhängigkeiten) die noch nicht gelöst sind oder zumindest beobachtet oder zugeteilt werden sollten.

Das lange Scrum of Scrums unterteilt sich somit in zwei Teile. Im ersten Teil wird das kurze Scrum of Scrums mit einer 15 Minütigen Timebox abgehalten in der die bekannten Fragen beantwortet werden. Identifizierte Probleme werden in das Scrum of Scrums Problem Backlog übernommen. Es werden keine Probleme gelöst oder Namen einzelner Teammitglieder genannt. Das weglassen von Namen hilft laut Cohn dabei das Abstraktionslevel auf Teamebene und somit die Redezeit jedes Teilnehmers möglichst gering zu halten. Der zweite Teil des langen Scrum of Scrums widmet sich dann der Besprechung der festgestellten (und optional priorisierten) Probleme aus dem Scrum of Scrums Problem Backlog.

Wie auch das kurze Scrum of Scrums besitzt das lange Scrum of Scrums keine festen Mitglieder sondern rekrutiert seinen Stab aus wechselnden Mitgliedern aus den Entwicklungsteams.

17.4 Sprint Review Meeting

> *„Individuen und Interaktionen mehr als Prozesse und Werkzeuge."*
> Das Agile Manifest [Bec01]

17.4.1 Sprint Review Messe

Bei einer Sprint Review Messe treffen sich alle Teams und Stakeholder zu einem gemeinsamen Sprint Review Meeting. Dieses findet in einem großen Raum statt bei dem jedes Team seinen eigenen Messestand besitzt. Jeder Messestand sollte mit Notebooks, Beamer sowie Schreibmaterial ausgestattet sein. Auf allen Notebooks ist das aktuelle Produkt Inkrement installiert und ein Notebook an den Beamer angeschlossen. Zu Beginn der Veranstaltung versammeln sich alle anwesenden und die Messe wird durch einen Scrum Master oder Super Product Owner anmoderiert. Er wiederholt zunächst das gemeinsame Sprintziel (falls vorhanden) für den vergangenen Sprint und erklärt (falls nötig) das weitere Vorgehen auf der Messe. Daraufhin übergibt er das Wort an die Product Owner der einzelnen Teams. Diese stellen kurz vor was ihr Team seit der letzten Sprint Review Messe erledigt hat und was die aktuellen Brennpunkte in der Entwicklung sind bei denen Feedback von den Teilnehmern gewünscht wird. Nach dieser Vorstellungsrunde wird die Messe eröffnet. Die Teams begeben sich zu ihren Ständen und stehen den Besuchern für Fragen zur Verfügung. Sie helfen den Interessenten dabei die neuen Features auf den Notebooks auszuprobieren und holen so Feedback zu ihrem Produktinkrement ein.

Dadurch, dass niemand gezwungen wird bestimmte Stände zu besuchen kann man davon ausgehen, dass die Besucher eines Standes starkes Interesse daran haben Feedback zu dem Inkrement des Teams zu geben. Die so entstehenden Gespräche sind demnach sehr wertvoll für das Team und für den Wert des Produkts.

17.4.2 Zeitlich abgestimmte Sprint Review Messe

Nachteilig an der zuvor genannten Methode ist, dass die Teams untereinander nur erschwert Feedback einholen können da sie an ihre Messestände „gebunden" sind. Um dies zu umgehen kann man den offenen Teil der Messe zeitlich in zwei Teile unterteilen: Einen für die Stakeholder und einen für die Teams. Im ersten Teil würden die Teams wie gehabt an ihre Stände gebunden sein um so Feedback von den Interessenten einzuholen. Im zweiten Teil steht es den Entwicklern dann frei sich auf der Messe untereinander auszutauschen und lediglich ein Bruchteil des Teams bleibt abwechselnd am Stand zurück.

Eine weitere Möglichkeit Feedback aller Teilnehmer und Teams der Messe einzuholen gleicht der des Speed-Datings. Beim Speed-Dating bleibt immer eine Gruppe der Teilnehmer (meist die Teilnehmerinnen) sitzen und der zweite Teil (die Männer) wechselt in periodischen Abständen zur nächsten Teilnehmerin. Gewechselt wird immer gesammelt und als Zeichen hierfür dient ein lautes Signal. Dieser Vorgang wird wiederholt bis jede Teilnehmerin jeden Teilnehmer kennengelernt hat. Dieses Prinzip kann auf die Sprint Review Messe angewandt,

werden indem immer ein Teil der Teams periodisch den Messestand zum jeweils nächsten Team wechselt.

Um die Teammitglieder aktuell zu halten, die stets am eigenen Stand verweilt haben, ist es bei beiden Möglichkeiten erforderlich eine kurze Teaminterne Nachbesprechung der Messe abzuhalten.

17.5 Sprint Retrospektive Meeting

> „*In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.*"
>
> Das Agile Manifest [Bec01]

17.5.1 Herzschlag Retrospektiven

Bei mehreren und verteilten Teams gibt es zwei Arten von Prozessen, die es kontinuierlich zu verbessern gilt:

1. Die Prozesse auf Teamebene
2. Die gemeinsam verwendeten Prozesse aller Teams

Zur Verbesserung der Prozesse auf Teamebene kann, wie bei der Anwendung von Scrum mit einem Team, auf ein lokales Sprint Retrospektive Meeting zurückgegriffen werden. Dieses sollte ebenfalls an jedem Ende eines Sprints stattfinden und so dabei helfen die Effektivität des Teams auf lokaler Ebene zu verbessern.

Um die gemeinsam verwendeten Prozesse zu verbessern ist es erforderlich, dass sich alle Teams an regelmäßigen Sprint Retrospektive Meetings beteiligen. Die gemeinsamen Sprint Retrospektive Meetings sollten zumindest zu jedem Sprintende aber auch zu wichtigen Meilensteinen wie z.B. dem Release des Produkts intern oder in Produktion (siehe Kapitel 15.5) erfolgen. Das verfolgen eines gemeinsamen Ziels, den Prozess zu verbessern und die Regelmäßigkeit der Retrospektiven gaben diesen Meetings den Namen „Herzschlag Retrospektiven" [Eck09, S. 146].

Das treffen aller Mitarbeiter aller Entwicklungsteams zu regelmäßigen Sprint Retrospektive Meetings an einem Standort ist jedoch bei vielen oder international verteilten Teams nicht mehr ohne weiteres möglich. Eine Möglichkeit um ein gemeinsames Sprint Retrospektive Meeting dennoch regelmäßig stattfinden zu lassen ist, ähnlich wie beim Scrum of Scrums (siehe Kapitel 17.3.1), nur jeweils einen Vertreter der Teams an der gemeinsamen Retrospektive teilhaben lassen. Der Nachteil beim aussenden eines Vertreters ist, dass das lokale und das gemeinsame Sprint Retrospektive Meeting nie an einem Tag stattfinden kann, sofern die Teams an unterschiedlichen Standorten arbeiten. Die Reisezeit und die Dauer des Meetings selbst würden es schwierig machen beide Meetings an einem Tag abzuhalten. Um die gemeinsamen Sprint Retrospektive Meetings dennoch am gleichen Tag wie die lokalen Team Retrospektive Meetings abzuhalten kann auf virtuelle Retrospektiven zurückgegriffen (siehe folgendes Kapitel 17.5.2) werden.

17.5.2 Virtuelle Retrospektiven

Über ein möglichst reichhaltiges Kommunikationsmedium (z.B. Videotelefonie) durchgeführte virtuelle Sprint Retrospektive Meetings haben den Vorteil, dass die Teilnehmer nicht mehr vor Ort sein müssen, um Verbesserungsvorschläge zu diskutieren. Nachteilig ist, dass selbst das reichhaltigste Kommunikationsmedium nie das persönliche Gespräche von Angesicht zu Angesicht ersetzen kann. Daher sollten virtuelle Retrospektive regelmäßig durch persönliche Retrospektiven untermauert werden. Um dies zu erreichen könnten z.B. die gemeinsamen Sprint Retrospektiven immer an unterschiedlichen Standorten stattfinden. So könnten die Vertreter der Teams (welche ebenfalls wechseln sollten) in regelmäßigen Abständen auch persönlich an den Sprint Retrospektive Meetings ihres eigenen Teams teilhaben.

Die zweite Möglichkeit, die ausgesandten der Teams auf dem aktuellen Stand zu halten, ist das Ausklammern der Stellvertreter aus den lokalen Retrospektiven der Teams [Eck09, S. 148]. Diese müssten dann z.B. über ein minderes Kommunikationsmedium wie E-Mail oder einem Wiki über den Ausgang des Sprint Retrospektive Meetings unterrichtet werden.

Der Vorteil dieser Art Retrospektiven abzuhalten liegt in der gesparten Reisezeit sowie dass die Retrospektiven wie bisher im kleinen an einem Tag abgehalten werden können. Ein Nachteil ist die notwendige Ausgrenzung von Stellvertretern aus den Teams, die dadurch nicht im vollen Umfang von den Erkenntnissen der lokalen Retrospektiven profitieren können.

17.5.3 Meta Retrospektiven

Wie auch bei den Scrum of Scrums (siehe Kapitel 17.3.1) ist es bei den Retrospektiven denkbar, diese in mehreren Stufen durchzuführen. Die bekannte Retrospektive auf Teamebene könnte in diesem Fall um weitere Retrospektiven beispielsweise wie folgt erweitert werden:

- Retrospektive aller Teams der gleichen Komponente (siehe „Komponententeams" in Kapitel 16.3.1)
- Retrospektive aller Scrum Master oder Product Owner
- Retrospektive aller Teams am gleichem Standort

Bei Meta Retrospektiven gilt es zu beachten, dass auch die nicht geladenen Teams über die für sie relevanten Punkte aus den Meetings informiert werden. Des Weiteren sollte darauf geachtet werden, nicht zu viele Staffelungen dieser Meta Retrospektiven durchzuführen um das wiederholte behandeln der gleichen Themen zu vermeiden. Diese Art Retrospektiven durchzuführen ist die gründlichste der drei vorgestellten, da sie alle Ebenen und Rollen des Scrum Prozesses involviert. Der Zeitaufwand für das Abhalten der Meetings ist jedoch entsprechend hoch.

Teil VI

Bewertung der vorgestellten Lösungsmöglichkeiten

Im folgenden befindet sich eine Zusammenfassung der vorgestellten Lösungsmöglichkeiten. Die Matrix enthält Stichpunktartig die Vor- und Nachteile der Lösungsmöglichkeiten und unterteilt diese zur qualitativen Unterteilung in drei Kategorien: „Anzustrebende Lösung", „Akzeptable Lösung" und „Sollte vermieden werden".

	Anzustrebende Lösung	**Akzeptable Lösung**	**Sollte vemieden werden**
Product Backlog			
Ein Product Backlog	+ Priorisiert + Verhandlung über PBI - Viele Einträge im Product Backlog		
Epic Backlog			- Es sollte nur ein Product Backlog geben - Mangelnde Transparenz - Ähnlich Wasserfall
Team Backlogs		+ Alignment erfolgt vorab - Mangelnde Transparenz	
Produktinkrement			
Kontinuierliche Integration	+ Done ist wirklich Done + Weniger Waste und höherer Fokus - Aufwändiger während der Sprints		
Merge Sprints			- Keine kontinuierliche Auslieferung - DoD ungenügend
Hardening Sprints		+ Sprints zur Qualitätssteigerung - Nur schwer schätzbar - Qualität vor den Hardening	
Sprints und Sprint Ziel			
Release-Kickoff	+ Abhängigkeiten werden sofort identifiziert + Löungsmöglichkeiten für Abhängigkeiten werden sofort erarbeitet		
Asynchrone Sprints			+ Planung für folgende Teams einfacher + Einzelne PO's und SM's können an mehreren Team Meetings teilnehmen - Abhängigkeiten schwer auflösbar - Nie vollständiges Produkt
Zeitlich abgestimmte asynchrone Sprints		+ Planung für folgende Teams einfacher + Einzelne PO's und SM's können an mehreren Team Meetings teilnehmen - Abhängigkeiten schwer auflösbar	
Definition of Done			
Gestaffelte Definition of Done	+ Teameigene DoD + Annäherung an gemeinsames Qualitätslevel aller Teams		

Abbildung 13: Zusammenfassung der vorgestellten Lösungsmöglichkeiten zu den Scrum Artefakten zur qualitativen Bewertung.

	Anzustrebende Lösung	Akzeptable Lösung	Sollte vemieden werden
Product Owner			
Kaskadierte Product Owner		+ Klare Verantwortlichkeit für PBI - Teilweise mangelnde	
Product Owner Gremium			- Widerspricht dem Scrum Guide
Product Owner Gremium mit SPO	+ Klare Verantwortlichkeit für PBI + Verteilte Arbeitslast		
Scrum Master			
Scrum of Scrum Masters	+ Gute Abstimmung möglich		
Entwicklungsteam			
Horizontale Aufteilung			- Wenig Kommunikation der Team untereinander - Kein vollständigen Features
Vertikale Aufteilung	+ Bessere Design- und Architekturentscheidungen + Wenig Waste durch Übergaben + Bessere Wissensverteilung + Weniger Planungsaufwand		
Gemischte Aufteilung		+ Hohe Motivation bei temoprären Komponententeams - Aufteilung kann der Unternehmensstruktur gleichen und nicht optimal sein	
Mitglieder in mehreren Teams		+ Abhängigkeiten können effektiv aufgelöst werden - Nur in heterogen aufgeteilten Teams sinnvoll - Gefahr, dass geteilte Entwickler zu viele Aufgaben bekommen	
Integrationsteams		+ Wachsen und schrumpfen nach Bedarf - Viele Meetings nötig	

Abbildung 14: Zusammenfassung der vorgestellten Lösungsmöglichkeiten zu den Scrum Rollen zur qualitativen Bewertung.

	Anzustrebende Lösung	Akzeptable Lösung	Sollte vemieden werden
Sprint Planning Meeting 1			
Laufende Vorausplanung		+ Bessere Koordintion der Sprint Backlogs - Zusätzlicher Zeitaufwand	
Großraumverfahren	+ Abhängigkeiten werden sofort identifiziert und angesprochen + PO muss nur an einem PM 1 teilnehmen		
Sprint Planning Meeting 2			
Sprint-Open-Space	+ Vielseitige und nachhaltige Lösungen + Gemeinschaftsbildend		
Last responsible moment			- Keine Synchronisation - Kann Zeitprobleme verursachen
Daily Scrum			
Scrum of Scrums (kurz)		+ Koordination und Synchronisation - Probleme werden nicht gelöst sondern nur angesprochen	
Scrum of Scrums (lang)		+ Koordination und Synchronisation - Größerer Zeitaufwand	
Sprint Review Meeting			
Sprint Review Messe		+ Sehr wertvolles Feedbacken durch Stakeholder - Teams untereinander können nur schwer Feedback geben	
Zeitlich abgestimmte Sprint Review Messe	+ Sehr wertvolles Feedbacken durch Stakeholder + Feedback durch andere Teams möglich		
Sprint Restrospektive Meeting			
Herzschlag Retrospektiven	+ Verbessert lokale Team- als auch gemeinsam verwendete Prozesse - Über zwei Tage verteilt - Nicht alle können bei den gemeinsamen Meetings teilnemen		
Virtuelle Retrospektiven			+ Retrospektiven an einem Tag - Größtenteils nur virtuell - Stellvertreter sind bei den lokalen Retrospektiven außen vor
Meta Retrospektiven		+ Gründlichste Art die Prozesse aller Teams und Rollen zu verbessern - Einzelne Personen müssen an vielen Retrospektiven teilnehmen - Gefahr zu viele Staffelungen an Retrospektiven einzuführen	

Abbildung 15: Zusammenfassung der vorgestellten Lösungsmöglichkeiten zu den Scrum Meetings zur qualitativen Bewertung.

Aus der Komposition der anzustrebenden Lösungen lässt sich folgendes Schaubild ableiten, dass ein skaliertes Framework zur Entwicklung komplexer Produkte in komplexen Umgebungen zeigt. Es gilt zu beachten, dass dies nur eine exemplarische Lösungsmöglichkeit von vielen ist. Wie Eingangs in Kapitel 4 erläutert sollte dieses Muster-Framework nicht blind übernommen werden sondern an die Gegebenheiten des Unternehmens angepasst werden.

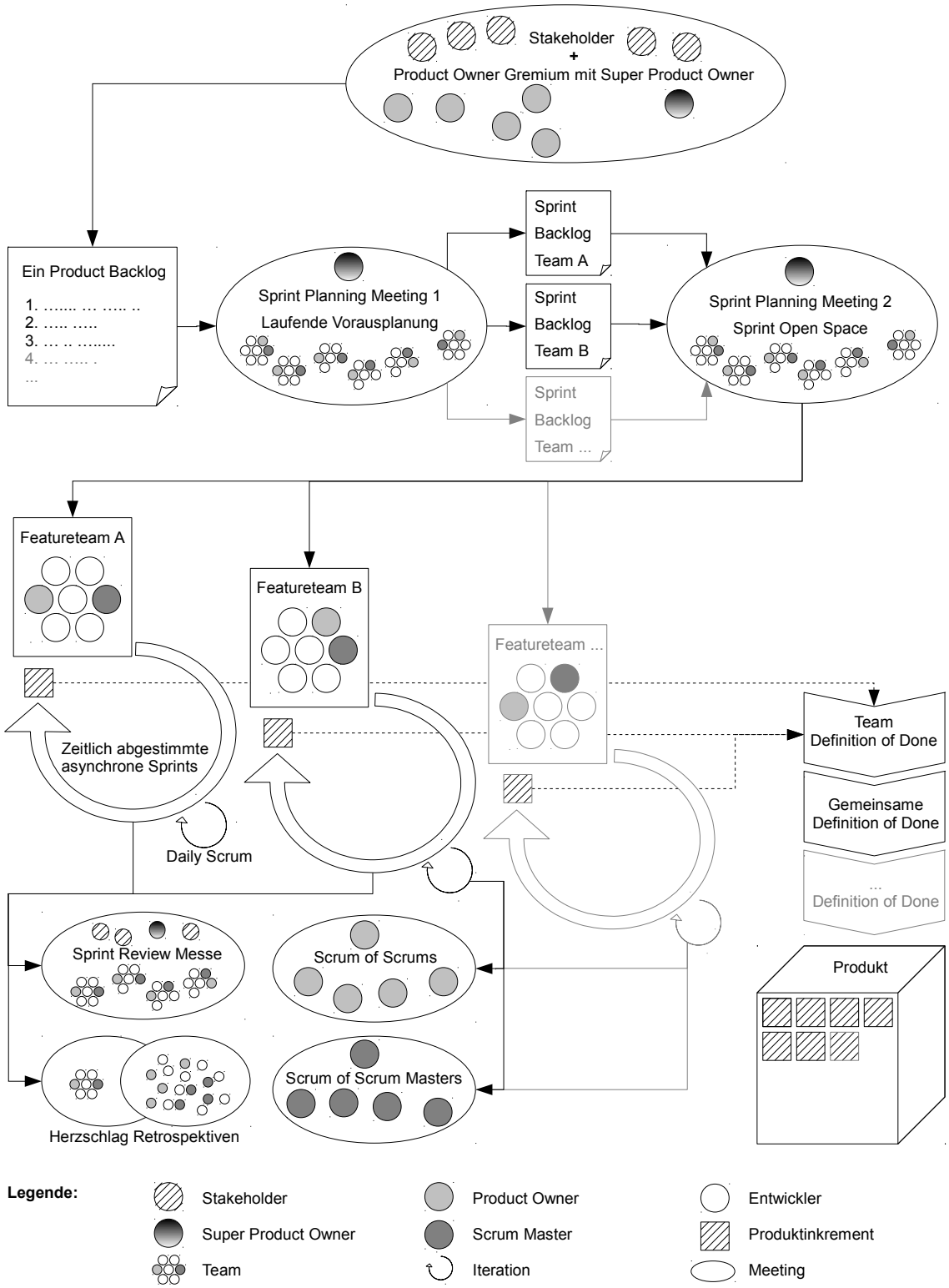

Abbildung 16: Exemplarische Lösungsmöglichkeit für die Skalierung von Scrum

Teil VII

Fazit und Ausblick

Scrum ist ein Framework für die Entwicklung komplexer Produkte und kann die Herausforderungen, die komplexe und umfangreiche Produkte mit sich bringen, meistern. Es gibt jedoch keine allgemeingültige Lösung dafür, wie Scrum im skalierten Umfeld funktioniert. Jedes Unternehmen muss eigenständig über die Scrum Prinzipie „Inspect & Adapt" herausfinden, was die in ihrem Kontext am besten geeignete Lösung ist. Die wichtigsten Faktoren, die es bei der Skalierung von Scrum im Unternehmen zu beachten gilt, wurden in Teil III dieser Arbeit evaluiert. Natürlich kann es noch weitere Faktoren geben, die unter bestimmten Umständen von Bedeutung sind und die es zu beachten gilt.

Viele erfolgreiche Unternehmen haben Scrum für die Entwicklung ihrer umfangreichen Produkte genutzt und sind dabei auf verschiedene Lösungsmöglichkeiten für die Skalierung der Scrum Artefakte, Rollen und Meetings gestoßen. Diese und weitere Lösungsmöglichkeiten wurden in dieser Arbeit evaluiert, hinsichtlich der Prinzipien von Scrum und dem agilen Manifest manifestiert und um eigene Ideen ergänzt. Im letzten Teil der Arbeit wurden die Lösungsmöglichkeiten zusammengefasst und bewertet. Aus der Komposition der anzustrebenden Lösungsmöglichkeiten wurde ein Muster-Framework abgeleitet, das zeigt wie Scrum bei vielen Teams im skalierten Umfeld funktionieren kann. Dass Scrum auch im Praxiseinsatz in großen Unternehmen besteht, untermauert der erprobte Einsatz von Scrum in Unternehmen wie Google, IBM, Yahoo und Toyota.

Meiner persönliche Meinung nach sind die Erfolgsfaktoren bei der Skalierung von Scrum die regelmäßige und qualitativ hochwertige Kommunikation sowie das kontinuierliche Feedback aller Projektbeteiligten zueinander. Diese beiden Schlüsselfaktoren finden sich in allen Success Stories der besagten Unternehmen, sind bei sämtlichen Interviews immer wieder zur Sprache gekommen und werden bei nahezu allen Lösungsmöglichkeiten zur Skalierung von Scrum angestrebt.

Dabei besteht qualitativ hochwertige Kommunikation zum einen darin, die Kommunikationswege möglichst direkt (von Angesicht zu Angesicht) zu gestalten als auch darin, eine Umgebung (in Form von kommunikationsfördernden Meetings und örtlichen Gegebenheiten) zu schaffen um diese ermöglichen. Das stetige Einholen von Feedback ist ein Erfolgsfaktor von Projekten, die mit Scrum durchgeführt werden. Im skalierten Umfeld muss der Anteil an Feedback, der eingefordert wird, mindestens proportional zur Projektgröße mitwachsen. Das Feedback sollte von technischer Seite aus in Form von Tests erfolgen als auch von Teams untereinander über die angestrebten Lösungsmöglichkeiten (z.B. zur Auflösung von Abhängigkeiten). Von den Stakeholdern sollte das Feedback über die Produktinkremente der einzelnen Teams sowie über das Gesamtprodukt eingeholt werden. Somit muss auch das Einholen von Feedback in kurzen und regelmäßigen Abständen von allen Stakeholdern und Projektbeteiligten gefordert und gefördert werden.

Ausblick

Querschnittliche Faktoren, die es bei der Skalierung von Scrum über die Scrum Artefakte, Rollen und Meetings hinaus zu beachten gilt, wurden in Teil IV festgehalten. Die evaluierten Herausforderungen lauteten Kommunikation, Softwarearchitektur, Unternehmenskultur, Projekt- und Prozessmanagement sowie Unternehmensstruktur. Die Lösung jeder dieser Herausforderungen im skalierten Scrum Umfeld ist sehr individuell und komplex, so dass sie über den Rahmen dieser Abschlussarbeit hinausgingen. Zeitgleich zur Entstehung dieser Arbeit wurde eine weitere Masterarbeit zum Thema „Software-Architektur in skalierten Scrum Entwicklungen" [Ros13] geschrieben, die sich ausschließlich dieser Herausforderung bei der Skalierung von Scrum widmet. So ist in folgenden Arbeiten noch Raum für die Lösung der übrigen querschnittlichen Faktoren bei der Skalierung von Scrum für große Projekte.

Auch im fünften Teil der Arbeit gibt es noch Raum für Ergänzungen bezüglich der Lösungsmöglichkeiten zur Skalierung der Scrum Artefakte, Rollen und Meetings. Je mehr Interviews man führt und je mehr Erfahrung man mit der Materie sammelt, desto mehr Möglichkeiten wird man entdecken. Eine Option, die vorgestellten Lösungsmöglichkeiten in der Praxis zu evaluieren und hinsichtlich ihrer Praxistauglichkeit zu bewerten, ergab sich bislang noch nicht. Wenn man beachtet, dass die Effektivität jeder vorgestellten Lösungsmöglichkeit abhängig ist vom Kontext, in dem sie eingesetzt wird, so könnte sich eine weitere Arbeit mit der Praxistauglichkeit verschiedener Lösungsmöglichkeiten zur Skalierung von Scrum befassen.

Teil VIII

Anhang 1 - Analyse der Vorgehensweisen bei der Skalierung von Scrum

A Success Story Adobe Systems Inc.

Adobe Systems Inc. entwickelt seit Anfang der 1990er Jahre Adobe Premiere mit verschiedenen klassischen Projektmanagement Methoden für Windows und Mac. Mit dem Release von Final Cut Pro für Mac OS X 1998 wurde die Entwicklung für den Mac eingestellt. 2005 entschloss sich das Adobe Premiere Pro Team den Markt für Mac OS X zurückzuerobern. Die Entwicklung erfolgte traditionell mit einem Wasserfall-ähnlichem Ansatz. Das folgende Release, „Adobe Premiere CS3", war qualitativ unterirdisch. 25% der zweijährigen Entwicklungszeit wurde damit verbracht Bugs zu fixen und die Qualität des Produktes hinreichend sicherzustellen [Gre12]. Dies geschah in einer Fülle von Überstunden und wurde mit schlechten Kundenrezessionen und Mitarbeitern, die aufgrund von Burnout-Syndromen in Krankenhäuser eingeliefert werden mussten, quittiert. Während dieser Zeit gab es bereits in einer anderen Abteilung bei Adobe ein Team welches erfolgreich mit Scrum arbeitete. 2008 entschloss sich Peter Green (Agile Adoption Leader und Certified Scrum Trainer) Scrum ebenfalls im Adobe Premiere Team einzuführen und die folgende Version „CS4" mit Scrum zu entwickeln. Peter Green entschloss sich das Entwicklungsteam in drei Cross-Funktionale Teams aufzuteilen. Für alle drei Teams gab es einen Scrum Master. Die Projektgröße machte es erforderlich den Product Owner (PO) bei seinen Aufgaben zu unterstützen. Die Unterstützung erhielt der Product Owner durch den Scrum Master (SM), einen Engineering Manager (EM), einen Quality Engineering Manager (QEM) sowie einen User Experience Manager (UXM). Alle vier wurden unter dem Titel „Product Owner Council" zusammengefasst, wobei der Product Owner immer das letzte Wort bei seinen Aufgaben hat. Die Mitglieder des Product Owner Council unterstützen den Product Owner in seinen Aufgaben die Feature Requests zu überprüfen, große Anforderungen in kleinere zu unterteilen, sowie die Product Backlog Items zu priorisieren. Jeder im Team stimmte diesen Bedingungen, in Form von „Working Agreements", und der Art Scrum auf diese Weise zu nutzen, zu.

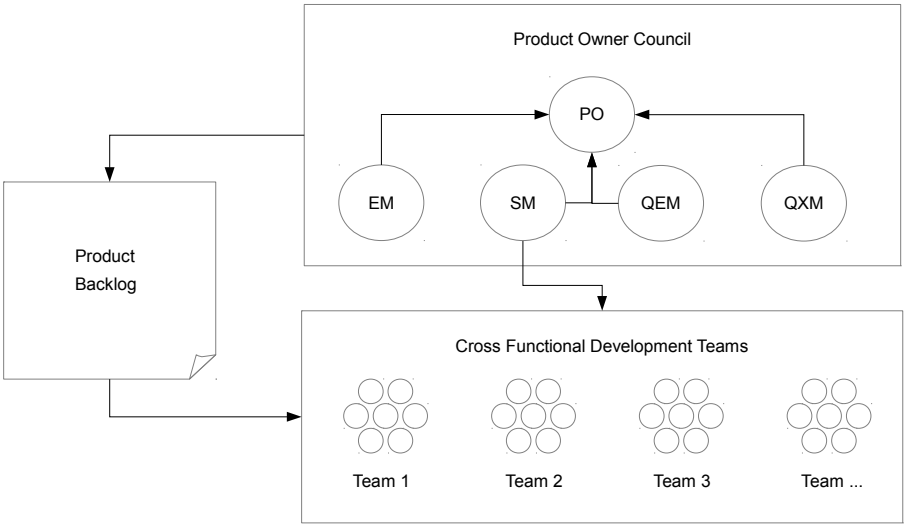

Abbildung 17: Organigramm des Adobe Premiere Scrum-Teams bei Adobe

Bereits nach kurzer Zeit stellten sich drei wesentliche Probleme ein, die den Scrum Master Peter Green vor neue Herausforderungen stellte:

1. Es gab Defizite bei der Kommunikation mit entfernten Teammitgliedern
2. Die Entwicklungsteams konnten Product Backlog Items teilweise nicht eigenständig und vollständig in ein Produktinkrement umsetzen
3. Es gab Probleme mit Abhängigkeiten zu anderen Abteilungen, die nicht nach einem agilen Ansatz arbeiteten

Diese werden in den folgenden Unterabschnitten näher erläutert.

A.1 Kommunikation mit entfernten Teammitgliedern

Die Entwicklungsteams von Adobe Premiere arbeiteten an unterschiedlichen Standorten und zum Teil auch von zu Hause aus. Bei den Scrum Meetings wurde auf Videokonferenz Systeme zurückgegriffen, um ein möglichst hohes Level der Kommunikation sicherzustellen. In der Praxis saßen so einige Teammitglieder in einem Raum während einige wenige über das Videokonferenzsystem zugeschaltet waren. Dies sorgte für ein Ungleichgewicht im Bezug auf die Reichhaltig der Kommunikationsmittel der Teammitglieder und somit für eine Art „Redepartner erster und zweiter Klasse", wie es Peter Green [Gre12, S. 2, Abs. A] darstellt. Er ist hierbei auf ein Hindernis aufgrund der Virtuellen Distanz (siehe Kapitel 10) der Teammitglieder zueinander gestoßen. Die Lösung von Peter Green und seinem Entwicklungsteam bestand darin, das Kommunikationslevel auf einen gemeinsamen, kleinsten Nenner zu bringen. Dies wurde erreicht indem jeder im Team das gleiche Kommunikationstool nutzte, um in den Scrum Meetings miteinander zu Kommunizieren.

A.2 Umwandlung von Produkt Backlog Items

Aus der Historie heraus wurden die Mitarbeiter von Adobe Systems Inc. darauf geschult große Probleme horizontal zu unterteilen. Die so unterteilten Probleme wurden dann von verschiedenen Abteilungen, die jeweils auf ihre Architekturschicht spezialisiert waren, gelöst. Der Scrum Ansatz die Probleme gemeinsam und ganzheitlich innerhalb eines Teams zu lösen widerspricht dieser historischen Vorgehensweise der horizontalen Unterteilung von Problemen. Nach Rücksprache und Schulung durch das erwähnte Team, das bereits mit Scrum arbeitete, wurde entschieden, dass die Product Backlog Items vom Product Owner Council nun vertikal unterteilt werden und stets einen Wert („Value") repräsentieren müssen. Ein Typisches Beispiel war in diesem Zuge die reine Darstellung des Anwendungsfensters der 64bit Version des Programms. Dieses Product Backlog Item umfasste alle Schichten des Systems und brachte einen Mehrwert für das Gesamtprodukt.

A.3 Zusammenarbeit mit nicht agilen Teams

Die „Adobe Creative Suite" Familie[22] besteht aus einer Sammlung von Design-, Grafik-, Audio- und Produktionsprogrammen. Jedes der zahlreichen Unterprogrammen wird in einer anderen Abteilung mit unterschiedlichen Ansätzen agil oder mit klassischen Methoden entwickelt. Viele der Programme nutzen gemeinsame Komponenten um die Kompatibilität der Programme untereinander sicherzustellen und redundante Entwicklungsarbeit zu minimieren - so auch Adobe Premiere. Das Adobe Premiere Team wusste um diesen Umstand und hat zwei Sprints am Ende der Entwicklungszeit dafür eingeplant, die gemeinsamen Komponenten von Entwicklungsteams zu integrieren, die nicht nach einem agilen Ansatz arbeiteten und somit erst gegen Ende der Entwicklungszeit ein fertiges Produkt abliefern konnten.

Sind Planänderungen an das Entwicklungsteam von Adobe Premiere herangetragen worden, welche Änderungen an einer gemeinsam genutzten Komponente erforderten so wurde ein traditioneller „Engineering Change Order"-Prozess angestoßen, der alle betroffenen Teams mit einbezog. Dies hatte zur Folge, dass die ungestörte Arbeit der Entwickler nicht gewährleistet werden konnte. Das Maß an Flexibilität in der Entwicklung glich immer dem des unflexibelsten Teams. Das Management von Adobe strebt in diesem Fall eine Lösung an, in der alle Teams die gemeinsame Komponenten entwickeln einen agilen Ansatz verwenden und so zumindest monatlich ein fertiges und qualitativ hochwertigen Release für die anderen Teams bereitstellen können.

B Success Story Primavera

Primavera entwickelt seit 1983 Projektmanagement Software um Firmen dabei zu unterstützen ihre Projekte und Ressourcen zu planen und Projekte erfolgreich umzusetzen. Mit dem Wachstum von Primavera und den immer komplizierteren und umfangreicheren Kundenanforderungen wurden die Release Zyklen ihrer

[22] Webseite der Produktfamilie: http://www.adobe.com/de/products/creativesuite.html

Software immer länger. Für den Release 3.5 ihrer Software im Jahr 2002 machte das Entwicklungsteam viele Überstunden und arbeitete auch am Wochenende. Dennoch wurde die Software erst drei Wochen nach dem geplanten Release ausgeliefert, vom Management als „unvollständig" angesehen, Bonus Zahlungen ausgelassen und die Moral des Teams war somit verflogen. Bob Schatz, Vice President of Development bei Primavera, wusste dass nicht das Entwicklungsteam das Problem war, sondern die Art wie Software bei Primavera entwickelt wurde. Anforderungen wurden in einem großen Dokument vom Marketing an die Projektmanager der Entwickler überreicht. Die Aufgaben wurden dort in kleine Teile unterteilt und den Analysten, Dokumentierern, Programmierern und Testern per „Command-and-Control" zugewiesen. Ein klassisches Wasserfall Modell. Ein Jahr nach dem Desaster mit Release 3.5 erfuhr Bob Schatz vom agilen Prozess und Scrum und entschloss dies bei Primavera einzuführen. Er lud Ken Schwaber zu Primavera ein um das Management von dieser neuen Art des Projektmanagements zu überzeugen und Scrum eine Chance zu geben. Das Management stimmte zu und somit wurde das Release 4.0 ihrer Software mit Scrum entwickelt. Im folgenden sind die im Whitepaper [OM04] von Primavera und Object Mentor Inc.[23] aufgeführten Erfolgsfaktoren und wie Primavera Scrum für sich skaliert hat dokumentiert.

B.1 Die Büroräume

Das Entwicklungsteam von Primavera besteht aus 90 Mitarbeitern, dass für Scrum in zehn Cross-Funktionale Teams unterteilt wurde. Die Büroräume wurden entsprechend angepasst und die ehemals in Kabinen unterteilten Büros wurden zu offenen Arbeitslandschaften für die Teams. Am Eingang zum Büro jedes Teams hängen die Vision des aktuellen Sprints sowie die Scrum-typischen Metriken um den Sprint fortschritt anzuzeigen. Die Mitarbeiter innerhalb der Teams haben sich in den Büroräumen so angeordnet, dass direkte Kommunikation von Angesicht zu Angesicht möglich ist.

B.2 Das Sprint Review Meeting

Durch die Synchronisation der Sprints aller Teams fällt auch das Sprint Review Meeting aller Teams auf einen gemeinsamen Termin. Bei Primavera treffen sich alle Entwickler, Product Owner, Stakeholder und Führungskräfte an diesem Tag zu einer „Sprint Review Messe". Diese findet in einem großen Raum statt, in dem jedes Team seine eigene „Demonstration Area" besitzt, in der sie ihr Produkt Inkrement der letzten 30 Tage präsentieren. Während die Entwicklungsteams an ihren Demonstration Areas verweilen, haben sämtliche Gäste der Messe die Möglichkeit den Entwicklern jedes Teams jeweils 15 Minuten lang Fragen zum aktuellen Release zu stellen, Feedback zu geben und mit den Entwicklern zu diskutieren. Nach dieser Timebox von 15 Minuten ertönt ein Signal und die Gäste wechseln zur nächsten Demonstration Area. Hochgerechnet bedeutet dies, dass das Sprint Review Meeting für alle Beteiligten bei zehn Teams 2,5 Stunden andauert. Dies unterschreitet die reguläre Dauer des Sprint Review Meetings

[23] Object Mentor Inc. begleitete Primavera bei der Einführung von eXtreme Programming

zwar, wird jedoch durch den Mehraufwand den Raum entsprechend vorzubereiten, aufgefüllt.

B.3 Sicherstellen von Software Qualität

Auch nach der Einführung von Scrum behielten die Entwickler die alte Praktik bei Tests und Bugfixing erst kurz vor dem erreichen des Release Datums (bzw. gegen ende eines Sprints) durchzuführen. Die kurze Entwicklungszeit und die gesteigerte Entwicklungsgeschwindigkeit machten jedoch kontinuierliches Testen, kontinuierliches Bugfixing und kontinuierliche Verbesserungen am Entwicklungsprozess erforderlich. Bob Schatz entschloss sich zu diesem Zweck auf eXtreme Programming (kurz „XP") zurückzugreifen. XP sieht vor, dass Modul- und Akzeptanztest *vor* der Entwicklung des eigentlichen Codes geschrieben werden. Die Tests laufen automatisiert ab und sorgen dafür, dass jeder der etwas mit dem neuen Produktinkrement zu tun hat von Anfang an in die Entwicklung involviert ist. Die Modultests[24] werden hierbei von den Entwicklern geschrieben und die Akzeptanztests von der Marketing- und Qualitätssicherungsabteilung. Gleichzeitig werden die Tests als Spezifikation für die Entwicklung sowie als Kriterium für die Definition of Done herangezogen. Diese Art der Softwareentwicklung wird als „Test Driven Development" bezeichnet.

Ein weiterer Aspekt für die Sicherstellung der Softwarequalität ist das verstehen und verwenden von guten Entwurfsmustern („Design Patterns"). Diese Entwurfsmuster sind Lösungsschablonen für wiederkehrende Probleme in der Softwareentwicklung. Dazu gehört auch das schreiben von sauberem Code, aufräumen von veraltetem Code („Refactoring") sowie das regelmäßige Review des Codes im Team.

Teil IX

Anhang 2 - Gedächtnisprotokolle

Die in den Gedächtnisprotokollen festgehaltenen Gedanken sind persönliche Meinung und Erfahrungen der jeweiligen Beteiligten.

C Gespräch zum Thema Softwarearchitektur

Interview: Ruben Burr (Software Architektur Experte)

Ort: NovaTec Ingenieure für neue Informationstechnologien GmbH, Dieselstraße 18/2, Echterdingen

Zeit: 16.11.2012, 8:00 - 9:00 Uhr

[24] Modultests, auch „Unit Tests" genannt, werden dazu verwendet die Funktionsfähigkeit einer einzelnen Methode oder Unterprogramms zu testen

C.1 Softwarearchitektur und die Rolle des Architekten

Das Gespräch beginnt mit einer kurzen Abklärung, was unter Softwarearchitektur verstanden wird, bzw. wie die Rolle eines Architekten in der klassischen Softwareentwicklung und in Scrum definiert wird. Softwarearchitektur beschreibt für Burr die technischen und fachlichen Bausteine eines Systems sowie deren Abhängigkeiten untereinander.

Softwarearchitektur hat mehrere Ziele, wie zum Beispiel die Vereinheitlichung und Standardisierung von Softwareteilen und beinhaltet dabei sowie technische als auch fachliche Bestandteile. Da nach Burr häufig bis zu 80% der Kosten in der Wartung von Software und nur 20% bei der Entwicklung anfallen, sollte der Fokus von Softwarearchitektur auf der Wartbarkeit von Software liegen.

Der Softwarearchitekt im klassischen Softwareentwicklungsprozess besitzt die Entscheidungsgewalt über die Architektur und gibt diese an die Entwickler weiter. In Scrum dagegen ist Architektur die Verantwortung des gesamten Entwicklungsteams. Nach dem Scrum Guide sollte es in einem Scrum Entwicklungsteam möglichst keine Spezialisten für bestimmte Entwicklungsaufgaben geben. Daher empfiehlt Burr, dass es bei großen Projekten innerhalb der Teams einen „Architekturkümmerer" geben sollte. Ein Architekturkümmerer muss ausreichend Fachwissen besitzen und seine Vorstellungen und Ideen überzeugend präsentieren können. Ausreichende Soft Skills (siehe [HS12]) sind hierbei sehr wichtig.

C.2 Skalierte Softwarearchitektur

Bei skalierten Projekten wird es wichtig, Architekturentscheidungen auch mit dem Product Owner oder einem anderen Weisungsbefugten abzuklären, da die Softwarearchitektur teilweise mit in den Bereich Enterprise Architecture mit einfließt sowie davon beeinflusst wird. Im Folgenden beschreibt Ruben Burr seine Ideen für ein Vorgehen bei skalierten Softwareprojekten mit Scrum und die Handhabung von Softwarearchitektur in diesem Kontext.

C.3 Trennung von Architektur und Design

Eine Trennung von Architektur und Design sieht Herr Burr darin, dass Architekturfragen generell die großen und weitreichenden Entscheidungen betrifft, während Design eher während der Entwicklung geschieht. Eine schärfere Trennung der Begriffe möchte er nicht vornehmen.

C.4 Team Aufteilung

Bei mehreren oder verteilten Teams ist es nach Burr wichtig, dass die Zusammensetzung der Teams beachtet wird. Jedes Team sollte über Teammitglieder mit Architekturkenntnissen verfügen. Bei größeren Projekten könnte auch ein Architektur-Board installiert werden, welches seine Mitglieder aus den Subteams rekrutiert. Dieses Architekturboard identifiziert Herausforderungen in der Softwarearchitektur, versucht Lösungen zu entwickeln und diese so in die Teams zu tragen.

Eine weitere Variante für sehr große Entwicklungsprojekte wäre auch, ein separates Architektur-Team aufzustellen, welches sich ausschließlich um Architekturfragen kümmert und keine Arbeit an der Implementierung verrichtet.

C.5 Vorüberlegungen für Softwarearchitekturen

Bei einer Podiumsdiskussion zum Thema Agilität und Architektur war von einem Experten für effektive Produktentwicklung, Matthias Bohlen, zu hören, dass er vor zwei Jahren noch daran festhielt, dass zu Beginn eines Projektes keine Architekturentscheidungen getroffen werden sollten. Architekturentscheidungen sollten nur während der Entwicklung getroffen werden. Mittlerweile ist Herr Bohlen von dieser Aussage abgerückt und hin zu einem Modell namens „Design The Big Things Up Front" . Auch Burr sieht es so, dass ein Refactoring in Architekturfragen sehr aufwendig und teuer ist und daher zu Beginn eines Projekts gewisse Vorentscheidungen zum Thema Architektur und Design getroffen werden sollten.

C.6 Erfahrungsbericht eines Projektverlaufs

Im Folgenden befindet sich Stichpunktartig der Projektverlauf eines Projekts in einem Namhaften Konzern, welches Burr zum Teil begleitet hat.

- Das Team besteht ständig aus ca. 10-15 Personen mit einer hohen Fluktuation und wurde bereits zwei mal abgebrochen und vom Endkunden neu gestartet.
- Die Organisationsstruktur ist Silo-artig. Ein fachlicher Verantwortlicher innerhalb einer Ebene hat keine Entscheidungsgewalt über einen technischen Mitarbeiter der gleichen Ebene. Man ist auf das Wohlwollen des technischen Mitarbeiters angewiesen und muss bei Problemen über mehrere Stufen eskalieren um einen Entscheidung durchsetzen zu können.
- Das Projekt wurde ursprünglich von einem Team gestartet, welches nach dem klassischen Wasserfallmodell vorgegangen ist. Im Zuge des Architekturentwurfes wurden über 300 Seiten Dokumentation erstellt. Drei Monate bevor ein Prototyp fertiggestellt sein sollte stellte das Team fest, dass es diesen Termin nicht einhalten kann. Das Projekt wurde gestoppt und unter anderer Zusammenstellung intern neu gestartet.
- Auf Grund des Zeitdrucks wurde auf Grundlage der bestehenden Funktionalitäten zunächst ein lauffähiger Prototyp erstellt. Auf Architekturelle Konsistenz wurde kein Wert gelegt, das System sollte möglichst viele umgesetzte Features aufweisen um einen erneuten Projektabbruch durch den Endkunden vorzubeugen.
- Nach einiger Zeit stellte das Team fest, dass die Entwicklungsgeschwindigkeit stark nachließ, da:
    - Ständig Bugfixes zu erledigen waren

  - Die Beziehungen der Klassen immer unklarer wurde
  - Die Struktur des Projekts keinerlei Konsistenz aufwies

  Das Team stand somit einer großen Anzahl von technischen Schulden gegenüber, welches ein Weiterkommen stark erschwerte.

- Es wurde der Entschluss gefasst das komplette System neu aufzusetzen. Die Grundstruktur wurde neu gestaltet, die technischen und fachlichen Verantwortlichkeiten geklärt und die gängigsten Stereotypen für Funktionalitäten festgelegt. Dieser Vorgang wurde parallel zur Entwicklung am alten System vorgenommen. Es dauerte über ein halbes Jahr, bis das neue System auf einem vergleichbaren Stand war, wie das Alte bei der Entscheidung zur Neuentwicklung. Einige Funktionalitäten des alten Systems mussten am Ende nicht übernommen werden, da die Relevanz nicht so hoch war, wie zuvor gedacht.

Man sieht nach Burr an diesem Projekt, dass „Big Design Up Front" fehlschlug. Das Refactoring von Softwarearchitektur erwies sich als sehr teuer und aufwendig, Refactoring ist nach Burr ein Werkzeug für kleinere, lokale Aufgaben. Folglich sollten gewisse Vorüberlegungen für Softwarearchitekturen angestellt werden. Diese Architektur wird kontrolliert weiterentwickelt, der Architekt hat die Aufgabe die Entwicklung der Architektur mitzuverfolgen und zu steuern. Dazu gehört auch, dass nicht zu viel vorgearbeitet wird, um Überkonstruktion zu vermeiden.

C.7 Softwarearchitektur Qualität

Bei der NovaTec GmbH ist vorgeschrieben, dass vor dem Einchecken in das Repository ein Codereview von einem zweiten Entwickler vorgenommen wird.[25] Architektur-Reviews sind eine sinnvolle Ergänzung hierzu. Generell sollten Architektur-Entscheidungen nach Burr auch immer kritisch hinterfragt werden.

D Gespräch zum Thema Softwarearchitektur

Interview: Volker Koch (Software Architektur Experte)

Ort: NovaTec Ingenieure für neue Informationstechnologien GmbH, Dieselstraße 18/2, Echterdingen

Zeit: 23.11.2012, 10:00 -11:30 Uhr

D.1 Situationsbeschreibung und Herausforderungen

Volker Koch arbeitet an einem Projekt mit über 100 Entwicklern sowie zusätzlichen Testern und fachlichem Personal mit, welches seit dem Jahr 2000 entwickelt

[25] Vergleichbar mit dem „Vier-Augen"-Prinzip aus dem Extreme Programming.

wurde. Zu Beginn des Projektes wurde eine umfassende Architektur und Dokumentation entworfen, die jedoch im weiteren Verlauf des Projektes kaum noch beachtet oder eingehalten wurde.

Weiterhin wurde zu Beginn des Projektes ein Framework entwickelt, dessen Umgang allerdings nicht eindeutig erklärt wurde. So verlagerten einige Teams Funktionalitäten in die Benutzeroberfläche, während andere eine klare Schichtentrennung praktizierten. Es entstand Architekturheterogenität durch die verschiedenen Stile oder „Programmierphilosophien" in den Entwicklungsteams.

Für die Einhaltung von Architekturrichtlinien gab es bis einschließlich 2008 keine klare Verantwortlichkeit, die Entwicklung von Funktionalitäten stand im Vordergrund. Weiterhin wurden keine technischen Schnittstellen definiert, es konnte theoretisch aus jedem Teil der Software direkter Einfluss auf andere Bereiche genommen werden. Aus dieser Historie heraus entstanden zyklische Abhängigkeiten der Softwarekomponenten zueinander, wodurch ein kompletter Build unmöglich und das Zusammenfügen mehrerer partieller Builds notwendig wurde.

Daneben wurde ein Phasenmodell angewendet, welches für jeden Change Request einen sequentiellen Ablauf ähnlich dem Wasserfallmodell vorsieht. Der Workload war dadurch sehr unterschiedlich, Architekten wurden z.B. während der Entwurfsphase stark belastet. Das ChangeRequest Management benötigte auf Grund der langen Releasezyklen für die Umsetzung eines Features bis zu 1,5 Jahre. Ein Workaround für diese Problematik war das Einführen von „HotFixes", welche neben den Change Requests eingefügt wurden.

Ein weiteres Problem war, dass die verschiedenen Abteilungen häufig den Standort wechselten und somit keine Bindung zwischen den verschiedenen Bereichen entstehen konnte bzw. diese auch häufig nicht gewollt war.

Eine weitere Auswirkung der großen Projektgröße war ein hohes Maß an Bürokratie, welches sich zum Beispiel in der Zeiterfassung bemerkbar macht. So müssen Arbeiten an verschiedenen Teilen der Software häufig anderen Kostenstellen im Unternehmen zugeordnet werden. Die hohe fachliche Komplexität im Projekt macht zudem eine hohe Spezialisierung der einzelnen Bereiche notwendig, die Bildung von funktionsübergreifende Teams, die in allen Teilen der Software einsetzbar sind, ist in diesem Umfeld unmöglich. Dazu wurde durch die größe des Projektes der Aufwand für die Verwaltung der vielen Change Requests, Bugs und Hot Fixes sehr hoch. Einige Personen im Team waren ständig mit der Priorisierung der eingehenden Änderungswünsche beschäftigt.

Eine Ursache für die lange Umsetzungsdauer liegt laut Herrn Koch in der historisch gewachsenen Software sowie der mangelnden Pflege der Architektur. Nach Herrn Koch ist „Eine Software Architektur, die nicht geprüft wird, nicht existent" - was in diesem Projekt bis 2008 der Fall war.

Nach den in dem Projekt existenten Hierarchien ist es auch heute nicht möglich Architekturprobleme direkt zur Aufarbeitung in die Teams zu geben. Sie scheitern an der Akzeptanz beim Kunden oder am mangelnden Zeitbudget der Architekten. Architekten sind im Projekt in einem virtuellen Architekturteam organisiert und dazu angehalten 50% ihrer Zeit für die Architektur zu verwenden und 50% für die Arbeit in den Entwicklungsteams. Die letzte Entscheidung über die Entwicklung eines Features oder das Durchführen eines Refactorings

liegt beim Kunden, dem die Notwendigkeit von architekturellen Arbeiten häufig nicht vermittelt werden kann. Für Herrn Koch ist daher das Modell des virtuellen Architekturteams nicht geeignet. Seiner Meinung nach sollte ein festes Architekturteam mit eigenem Budget eingeführt werden, welches nicht dem „Feature-Druck" unterliegt.

D.2 Maßnahmen

Mit einem Neustart des Projektes im Jahr 2008 wurden einige Probleme erkannt und behoben. So wurde unter anderem:

- Die Einführung von Schnittstellen der Komponenten beschlossen, um die Abhängigkeiten der Komponenten untereinander zu verringern.
- Organisatorisch ein sogenanntes Architekturboard eingeführt, in dem neben fachlichen auch architekturelle Aktivitäten angesprochen werden sollen. Leider bezieht sich dieses Gremium nicht nur auf das hier besprochene Software-Projekt sondern auch auf andere Software-Projekte des Kunden, Legacy-Systeme etc..
- Es wurde ein regelmäßiges Architekturmeeting eingeführt, um Architekturfragen zu besprechen. Zwei Architekten dieses Meetings bringen 100% ihrer Zeit für Architekturthemen auf und befinden sich nicht in den Entwicklungsteams. Ein Chefarchitekt auf oberster Ebene kommuniziert Architekturthemen zum Architekturteam. Er bestimmt das „Wie" in Architekturfragen. Eine Rückmeldung des Architekturteams zum Chefarchitekten ist nur schwer möglich, da dieser schlecht erreichbar ist. In den Architekturmeetings werden exemplarisch folgende Themen behandelt:
    - Welche Schichtenarchitektur nutzen wir?
    - Wie prüfen wir die Architekturvorgaben?
    - Wie gestaltet sich die Aufteilung der Software in Komponenten?
    - Schnittstellendefinitionen
    - Zugriffswege von Klassen
    - Welche Persistenzschicht wird genutzt?

Die überarbeitete Organisation des Entwicklungsteams ist in der folgenden Abbildung dargestellt.

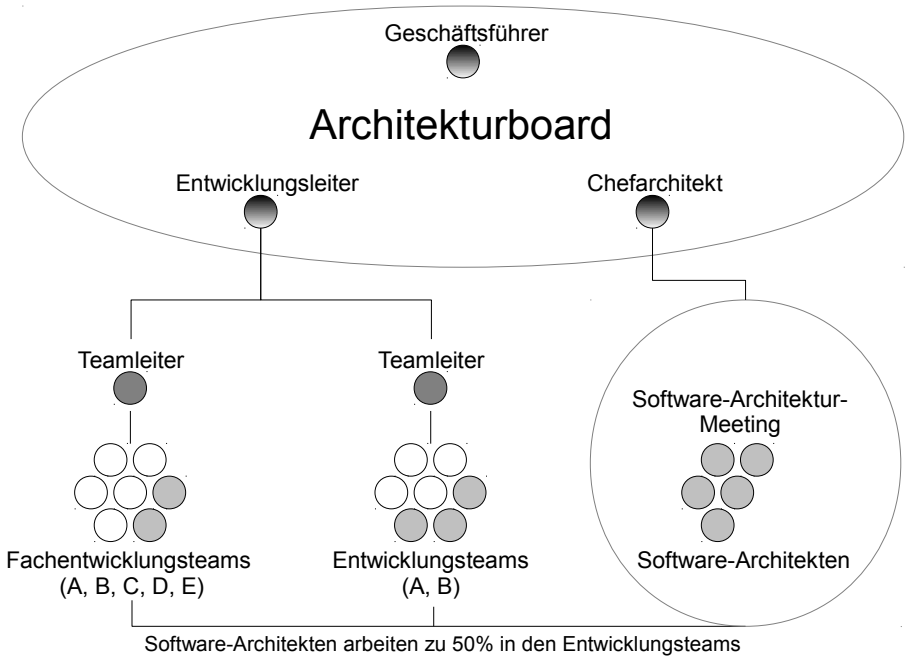

Abbildung 18: Organisation des Softwareprojektes

Ein hierbei festgestelltes Problem ist, dass es keine Ebene für das Management des Softwareproduktes an sich gibt. Über den Teamleitern und IT-Architekten für die Teilbereiche der Softwarelösung folgt direkt ein sehr hohes Management-Gremium, welches noch viele andere Aufgaben als nur die Verwaltung dieses einen Softwareproduktes hat. Folglich sind sowohl Entwicklungsleiter als auch Chefarchitekt häufig nur schwer zu erreichen, Entscheidungen können sehr lange dauern.

D.3 Lessons Learned

Einige Probleme im Projekt, wie die zyklischen Abhängigkeiten und unkontrollierten Zugriffe durch alle Ebenen und Bereiche der Software, konnten durch diese Maßnahmen behoben werden. Den fachlichen Entscheidern sind die architekturellen Aufgaben und dessen Notwendigkeit inzwischen bewusster. Leider gestaltet sich die Kommunikation zum Kunden diesbezüglich nach wie vor als schwierig, da der direkte Nutzen schwer zu vermitteln ist. Zudem ist die Entscheidungsgewalt der IT-Architekten nach wie vor auf die Umsetzung beschränkt, es kann kein Einfluss auf die Funktionalität genommen werden. Laut Herrn Koch wäre ein Standort für Entwicklungs- und Fachabteilungen sinnvoll, um den Problemen in den Absprachen entgegenzuwirken.

E Gespräch zum Thema Scrum und Prozessmanagement

Interview: Tobias Schäfer (Scrum Master, Software Developer)

Ort: NovaTec Ingenieure für neue Informationstechnologien GmbH, Dieselstraße 18/2, Echterdingen

Zeit: 17.12.2012, 10:30-11:15

E.1 Prozessmanagement und BPM

Aus Sicht von Tobias Schäfer sind Prozessmanagement, Business Process Management und weitere Management Begrifflichkeiten das Gleiche.

E.2 Scrum im Rahmen von BPM

Tobias Schäfer ist Scrum Master seines Entwicklungsteams und sorgt dafür, dass diese keinerlei Berührungspunkte mit den Geschäftsprozessen der Firma haben. Die Schnittstelle hierzu ist der Product Owner, welcher seine Aufgabe sehr gut macht und Input von den Kunden, Stakeholdern und der Marketingabteilung entgegennimmt und dem Team geordnet im digitalen Product Backlog[26] aufbereitet.

Als das Entwicklungsteam von Tobias Schäfer mit seiner Arbeit begann, hatten sie den Vor- und Nachteil, dass sie auf der „grünen Wiese" beginnen konnten. Somit gab es keinerlei Vorgaben hinsichtlich Entwicklungsumgebung, Versionskontrollsystem oder sonstige organisatorische Richtlinien. Die Vorteile sind die freie Verwirklichung und volle Entfaltung der einzelnen Entwickler. Der Nachteil ist nach Herr Schäfer, dass sämtliche Organisationsstrukturen um das Team herum sich speziell auf das Entwicklungsteam abstimmen müssen. So muss die IT-Abteilung eigens Server bereitstellen und spezielle Backup Strategien erstellen, die bisher nicht nötig waren. Kommen im Rahmen von weiteren Projekten zusätzliche Teams hinzu, so muss sich die IT-Abteilung auch auf deren Prozesse und Tools einstellen. Dies führt zweifelsohne zu einer immer komplexeren heterogenen IT-Landschaft mit all seinen Nachteilen. Auch ein Mitarbeiterwechsel von einem Entwicklungsteam in ein anderes ist schwieriger wenn kein Team an homogene Prozessstrukturen gebunden ist. Die Einarbeitungszeit steigt somit und die Flexibilität sinkt.

E.3 Erweiterungen von Scrum

Um Teamprobleme und eskalierte Probleme zu besprechen und einen Austausch zwischen dem Scrum Master (Tobias Schäfer), dem Product Owner sowie den Stakeholdern zu fördern wurde ein zweiwöchig stattfindendes Meeting eingeführt. Der so benannte „Lenkungsausschuss" dient damit der Produktentwicklung und als zusätzliches Meeting der Koordination zwischen Product Owner, Scrum Master und den Stakeholdern.

[26] Bei der NovaTec GmbH „Jira" mit „Greenhopper"

F Gespräch zum Thema international verteilte Teams

Interview: Stephanie Gasche (Management Consultant)

Ort: Persönliches Gespräch am Scrumtisch in Stuttgart sowie Skype Meeting zwischen Oldenburg und Leinfelden-Echterdingen

Zeit: 28.08.2012 (Scrumtisch Stuttgart) und 07.01.2013 (Skype Meeting)

F.1 Situationsbeschreibung

Im Rahmen ihrer Tätigkeit als Management Consultant bei der bor!sgloger training KG arbeitete Stephanie Gasche als Scrum Masterin in einem Projekt mit international verteilten Teams mit. 12 Entwickler sind verteilt über drei Standorte in Indien, Vietnam und Deutschland. Das größte Domänen Know How befindet sich bei den Entwicklern in Deutschland. Der Product Owner kommt aus England.

F.2 Herausforderung Zeitzonen

Durch die Verteilung der Entwickler über Länder mit unterschiedlichen Zeitzonen ist die Zeit, in der sich jeder Entwickler gleichzeitig am Arbeitsplatz befindet auf fünf Stunden reduziert. Da jeder der 12 Entwickler an allen Meetings komplett teilnehmen sollte machte es dieser Umstand erforderlich die Scrum Meetings über drei Tage zu verteilen. Das Sprint Review und Sprint Retrospective Meeting wurden an einem Tag abgehalten und das Sprint Planning Meeting 1 sowie das Sprint Planning Meeting 2 an jeweils einem weiteren Tag. Da dies in den Augen der Entwicklungsteams zu viel Zeit in Anspruch nahm, wurden die für Meetings zur Verfügung stehenden Tage gegen Ende des Projekts auf zwei reduziert. Das Daily Scrum wurde zeitlich ebenfalls so platziert, dass alle Entwickler aus allen Ländern teilnehmen konnten.

F.3 Herausforderung Kommunikation

Bei der Kommunikation gab es zwei wesentliche Aspekte, die die internationale Verständigung erschwerten. Der erste Aspekt war schlicht ein Problem mit der Verständlichkeit des Product Owners für die Entwickler aus Vietnam und zum Teil aus Indien. Der Product Owner war für die Vietnamesen zu schnell in der Aussprache und auch aufgrund des nordenglischen Dialekts nur schwer verständlich. Dies konnte durch Einzelgespräche über die Scrum Masterin Stephanie Gasche festgestellt und durch zusätzliche Englischkurse der Entwickler in Vietnam gelöst werden. Des weiteren wurde der Product Owner gebeten bei der Kommunikation in den Meetings langsam und deutlich zu sprechen.

Der zweite Aspekt, der die Kommunikation erschwerte waren unzureichende technische Hilfsmittel. Bei den Entwicklern in Indonesien mangelte es an hochwertigen Headsets und Webcams. Durch die Aufteilung der Büroräume in Cubes

gibt es in Indien zusätzlich Probleme mit der Geräuschkulisse bei den Meetings. Ein weiteres Manko waren teilweise Probleme mit der Telefonverbindung - speziell nach Vietnam. In Summe führten diese Hindernisse dazu, dass sich die Entwickler in Vietnam und Indien nur begrenzt an den Diskussionen beteiligten. Durch die Zusendung von qualitativ hochwertigerem Equipment nach Vietnam und Indien konnten zumindest ein paar der technischen Hindernisse beseitigt werden. Dennoch musste jedes Meeting durch die Scrum Masterin mit einer entsprechenden Vorlaufzeit vorbereitet werden um den Zeitverlust der durch das Aufsetzen der Verbindung, Einrichtung der Webkonferenzräume etc. verloren ging, zu minimieren.

F.4 Herausforderung Kulturelle Unterschiede

Im Entwicklungsteam in Vietnam gibt es einen Entwickler welcher zugleich als Teamleiter fungiert. Während den Meetings wurde festgestellt, dass die übrigen Entwickler sich nur sehr selten eigenständig zu Wort meldeten und die Kommunikation des Teams hauptsächlich über den Teamleiter stattfand. Die Hintergründe hierzu liegen vermutlich in der dortigen Kultur. Aufgedeckt wurde dies durch persönliche Einzelgespräche der Scrum Masterin mit den Entwicklern vor Ort in Vietnam.

Um dieses Problem zu adressieren wurde in jedem Meeting explizit jeder Entwickler jedes Landes gefragt, ob er etwas zum Thema beitragen kann. Dies geschah proaktiv durch die Scrum Masterin. Hierdurch wurden auch die eher zurückhaltenden Inder noch aktiver in die Diskussionen eingebunden.

F.5 Lessons Learned

Die Arbeit mit verteilten Teams macht es erforderlich regelmäßig und oft an allen Standorten mit verteilten Teams persönlich anwesend zu sein. Nur so kann man Probleme effektiv aufdecken und beheben sowie die Zusammenarbeit der Teams weiter ausbauen. Dies ist ebenfalls nötig um die persönliche Ebene bei der Kommunikation zu steigern und somit das Vertrauen aller Beteiligten untereinander weiter auszubauen.

Nach Meinung von Frau Gasche sollte unabhängig hiervon vor dem Projektstart mit verteilten Teams geprüft werden, ob sich die Arbeit mit international verteilten Teams überhaupt rechnet. Der gewonnene Kostenvorteil durch günstigere Arbeitskräfte kann schnell durch hohe Kosten bei der Kommunikation sowie die nötigen Reisen aller Beteiligten aufgezehrt werden. Des weiteren kann es in extremen Fällen vorkommen, dass die hinzugezogenen Unternehmen in anderen Ländern nur wenig bis gar kein Grundwissen in der Anwendungsdomäne besitzen. Dies sollte ebenfalls vorab geprüft werden.

Teil X

Literaturverzeichnis

Literatur

[AK07] Ambler, Scott W. ; Kroll, Per: Lean development governance. In: *IBM* (2007), 32. ftp://ftp.software.ibm.com/software/rational/web/whitepapers/Lean_Development_Governance.pdf

[All84] Allen, T.J.: *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information Within the Ramp Organization.* Mit Press, 1984 http://books.google.de/books?id=kAhhQgAACAAJ. – ISBN 9780262510271

[Amb09a] Ambler, Scott: *Agility at Scale Survey.* 2009

[Amb09b] Ambler, Scott: Tha Agile Scaling Model (ASM) - Adapting Agile Methods for Complex Environments. In: *IBM* (2009), 35. ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/raw14204usen/RAW14204USEN.PDF

[Amb10] Ambler, Scott: *IT Project Success Rate by Team Size and Paradigm.* 2010

[Amb11] Ambler, Scott: *IT Project Success Survey 2011.* 2011

[Bec01] Beck, M. ; Van Bennekum A. ; Cockburn A. ; Cunningham W. ; Fowler M. ; Grenning J. ; Highsmith J. ; Hunt A. ; Jeffries R. ; Kern J. ; Marick B.; Martin R. ; Mellor S. ; Schwaber K. ; Sutherland J. ; Thomas D. K. ; Beedle B. K. ; Beedle: *Das Agile Manifest.* Website. http://www.agilemanifesto.org. Version: 2001

[Ber08] Berger, Roland: Projekte mit Launch Management auf Kurs halten. In: *Roland Berger* (2008), 29. http://www.rolandberger.de/expertise/branchenexpertise/it_industry/2008-08-26-rbsc-pub-Projekte_mit_Launch_Management_auf_Kurs_halten.html

[Bin97] Binner, H.F.: *Integriertes Organisations- und Prozeßmanagement.* Hanser, 1997 (REFA-Fachbuchreihe Unternehmensent wicklung). http://books.google.de/books?id=qQGApxgWGz0C. – ISBN 9783446191747

[BM02] Bradner, Erin ; Mark, Gloria: *Why distance matters: Effects on cooperation, persuasion and deception.* University of California, Irvine, 2002 http://www.mobilecommunitydesign.com/research/BradnerMark_CSCW_2002.PDF

[BP97] Bright, D. ; Parkin, B.: *Human Resource Management: Concepts and Practices.* Business Education, 1997 http://books.google.de/books?id=vN_sAAAACAAJ. – ISBN 9780907679950

[Car99] Carmel, E.: *Global Software Teams: Collaborating Across Borders and Time Zones.* Prentice Hall, 1999 (High Performance Cluster Computing Series). http://books.google.de/books?id=_cxwQgAACAAJ. – ISBN 9780139242182

[Coc03] Cockburn, A.: *Agile Software-Entwicklung.* mitp, 2003 http://books.google.de/books?id=oWn8ygAACAAJ. – ISBN 9783826613463

[Coh10] Cohn, M.: *Agile Softwareentwicklung.* Addison Wesley Verlag, 2010 (Sonstige Bücher AW). http://books.google.de/books?id=AAECyPcN3UoC. – ISBN 9783827329875

[Con68] Conway, Melvin E.: *How Do Committees Invent?* F. D. Thompson Publications, Inc., 1968 http://www.melconway.com/research/committees.html

[CRY] *Crystal Family.* http://de.wikipedia.org/wiki/Crystal_Family

[Don96] Donaldson, L.: *For Positivist Organization Theory.* SAGE Publications, 1996 http://books.google.de/books?id=Ur6epBvVuEsC. – ISBN 9780761952275

[Dru92] Drucker, P.F.: *Managing for the future: the 1990s and beyond.* Dutton, 1992 (Truman Talley books). http://books.google.de/books?id=cR3y4blqGUMC. – ISBN 9780525934141

[Dun93] Dunbar, R.I.M.: Coevolution of neocortex size, group size and language in humans. In: *Behavorial and brain sciences* (1993). http://www.cogsci.ucsd.edu/~johnson/COGS184/3Dunbar93.pdf

[Eck09] Eckstein, J.: *Agile Softwareentwicklung mit verteilten Teams.* Dpunkt.Verlag GmbH, 2009 http://books.google.de/books?id=fWy1PwAACAAJ. – ISBN 9783898646307

[Eck12] Eckstein, J.: *Agile Softwareentwicklung in großen Projekten: Teams, Prozesse und Technologien- Strategien für den Wandel im Unternehmen.* Dpunkt.Verlag GmbH, 2012 http://books.google.de/books?id=OjempwAACAAJ. – ISBN 9783898647908

[Feh08] Fehskens, Leonard: Re-Thinking Architecture. In: *The Open Group* (2008), 66. https://collaboration.opengroup.org/conference-live/protected/

login.php?prot=Y&gpid=197&gpid=197&laction=dlogin&target=sendfile.php&qstring=gpid%3D197%26type%3Ddoc%26id%3D17752%26fn%3Db3_fehskens_Re-Thinking_Architecture.pdf

[Fow03] Fowler, Martin: Who Needs an Architect? In: *IEEE SOFTWARE* (2003), 3. http://www.in-ag.eu/uploads/media/whoNeedsArchitect.pdf

[Fow04] Fowler, Martin: Is Design Dead? In: *XP 2000 conference* (2004), 14. http://martinfowler.com/articles/designDead.html

[Fri10] Friedrichsen, Uwe: Wer braucht einen Architekten? Über Ziele und Aufgaben von Architektur und Architekten. In: *OBJEKTspektrum* (2010), 8. http://www.codecentric.de/files/2011/05/wer-braucht-einen-architekten.pdf

[Glo09] Gloger, B.: *Scrum: Produkte zuverlässig und schnell entwickeln.* Hanser Fachbuchverlag, 2009 http://books.google.de/books?id=SrQqDJhBr6IC. – ISBN 9783446419131

[Gre12] Green, Peter: Adobe Premiere Pro Scrum Adoption How an Agile approach enabled success in a hyper-competitive landscape. In: *blogs.adobe.com* (2012), August. http://goo.gl/sHAA1

[HS12] Hruschka, P. ; Starke, G.: *Knigge für Softwarearchitekten.* Software + Support, 2012 (Schnell + kompakt). http://books.google.de/books?id=zpqppwAACAAJ. – ISBN 9783868020809

[ISO] ISO, IEC, IEEE: *ISO/IEC/IEEE 42010:2011 - Systems and software engineering – Architecture description.* http://www.iso.org/iso/catalogue_detail.htm?csnumber=50508

[Kot96] Kotter, J.P.: *Leading Change.* Harvard Business School Press, 1996 http://books.google.de/books?id=ib9Xzb5eFGQC. – ISBN 9780875847474

[KSW04] Krishna, S. ; Sahay, Sundeep ; Walsham, Geoff: Managing cross-cultural issues in global software outsourcing. In: *Commun. ACM* 47 (2004), April, Nr. 4, 62–66. http://dx.doi.org/10.1145/975817.975818. – DOI 10.1145/975817.975818. – ISSN 0001–0782

[Law86] Lawrence: *Organization and Environment.* McGraw-Hill Companies, The, 1986 (Harvard Business School Classics). http://books.google.de/books?id=b-UbAQAAMAAJ

[Lef] Leffingwell, Dean: *Scaled Agile Framework.* http://scaledagileframework.com/

[LR08] Lojeski, K.S. ; Reilly, R.R.: *Uniting the Virtual Workforce: Transforming Leadership and Innovation in the Globally Integrated Enterprise.* John Wiley & Sons, 2008 (Microsoft Executive

Leadership Series).
http://books.google.de/books?id=iHby7ailOCEC. – ISBN 9780470193952

[LS08] Lipnack, J. ; Stamps, J.: *Virtual Teams: People Working Across Boundaries with Technology.* Wiley, 2008
http://books.google.de/books?id=FI3sZhchpmsC. – ISBN 9780470438954

[LV08] Larman, C. ; Vodde, B.: *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum.* Addison-Wesley, 2008 (Agile Software Development).
http://books.google.de/books?id=HbRo4kYnTnMC. – ISBN 9780321480965

[LV10] Larman, C. ; Vodde, B.: *Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development With Large-Scale Scrum.* Addison-Wesley, 2010 (Agile Software Development Series).
http://books.google.de/books?id=fqdTsH36TVYC. – ISBN 9780321636409

[Max12] Maximini, Dominik: Fremde Gezeiten - Wenn agile Ideen an Kulturklippen zerschellen. In: *NovaTec GmbH* (2012), S. 24

[Mil56] Miller, G.A.: *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information.* The Psychological Review, 1956

[MM85] McLean, J. ; Marshall, A. ; Francis Pinter, London (Hrsg.): *Exploring Organisation Culture as a Route to Organisational Change.* Hammond V. (ed), Current Research in Management, 1985

[MSR07] Manuel, Sosa ; Steven, Eppinger ; Rowles, Craig: Are your engineers talking to one another when they should? In: *Harvard Business Review* 85 (2007), 133-6, 138, 140-2 passim.
http://www.biomedsearch.com/nih/Are-your-engineers-talking-to/18159793.html

[ND05] Nidiffer, Kenneth ; Dolan, Dana: Evolving Distributed Project Management. In: *Systems and Software Consortium* (2005), 6.
http://dl.acm.org/citation.cfm?id=1092707.1092729

[OM04] Object Mentor, Inc.: Primavera White Paper. In: *Primavera* (2004), 10.
http://agileinfusion.com/pdf/PrimaveraWhitePaper.pdf

[OOS08] OOSE Innovative Informatik GmbH: Einfluss klassischer und agiler Techniken auf den Erfolg von IT-Projekten. In: *OOSE Innovative Informatik GmbH* (2008), S. 18

[Pic07] Pichler, R.: *Scrum - Agiles Projektmanagement erfolgreich einsetzen.* Dpunkt.Verlag GmbH, 2007
http://books.google.de/books?id=KhoxPQAACAAJ. – ISBN 9783898644785

[PP03] Poppendieck, M. ; Poppendieck, T.: *Lean Software Development: An Agile Toolkit*. Addison-Wesley, 2003 (The Agile Software Development Series). http://books.google.de/books?id=hQk4S7asBi4C. – ISBN 9780321150783

[Put11] Putnam, Doug: Team Size Can Be the Key to a Successful Software Project. In: *QSM* (2011), 3. http://www.qsm.com/process_improvement_01.html

[Rec11] Recklies, Dagmar: Unternehmensstrukturen - zentralisieren oder dezentralisieren? In: *Recklies Management Project GmbH* (2011), 4. http://www.themanagement.de/pdf/StrukturenII.PDF

[Ros13] Rosemeyer, Nils: *Software-Architektur in skalierten Scrum-Umgebungen*, Hochschule Reutlingen, Masterarbeit, 2013

[Sah12] Sahota, Michael: *An Agile Adoption and Transformation Survival Guide*. InfoQ - Enterprise Software Development Series, 2012 http://de.slideshare.net/michael.sahota/agile-2012-an-agile-adoption-and-transformation-survival-guide

[SB08] Schwaber, Ken ; Beedle, Mike: *Agile Software Development with Scrum*. Pearson Education International, 2008 (Series in Agile software development). http://books.google.de/books?id=lO3XLgAACAAJ. – ISBN 9780132074896

[Sch99] Schneider, W.E.: *The Reengineering Alternative: A Plan for Making Your Current Culture Work*. McGraw-Hill, 1999 http://books.google.de/books?id=XUGgxdaO9uYC. – ISBN 9780071359818

[Sch11] Schneider, Ulf: Scrum und Architektur: Konzeptionelle Integrität im Scrum-Prozess. In: *OBJEKTspektrum* (2011), 5. http://allesagil.net/2011/06/27/scrum-und-architektur/

[SI08] Schwaber, K. ; Irlbeck, T.: *Scrum im Unternehmen*. Microsoft GmbH, 2008 http://books.google.de/books?id=L_3fJwAACAAJ. – ISBN 9783866456433

[Spa10] Spayd, Michael: Agile & Culture - The Results. In: *Collective Edge Coaching* (2010). http://collectiveedgecoaching.com/2010/07/agile__culture/

[SS11] Schwaber, Ken ; Sutherland, Jeff: *Scrum Guide*. Scrum.org, 2011 http://www.scrum.org/Scrum-Guides

[SS12] Schwaber, K. ; Sutherland, J.: *Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust*. John Wiley & Sons, 2012 http://books.google.de/books?id=sdnAZOAuuDkC. – ISBN 9781118228548

[Sta94] Standish Group International: CHAOS RISING - A chaos executive commentary. In: *Standish Group International* (1994). http://blog.standishgroup.com/pmresearch

[Sta10] Standish Group International: CHAOS SUMMARY FOR 2010. In: *The Standish Group International, Inc* (2010), S. 27

[Tuc65] Tuckman, B.W.: *Developmental sequences in small groups.* Psychological Bulletin, 1965

[Ver11] Version One: *State of agile Survey.* 2011

[Whe10] Wheelwright, S.C.: *Managing New Product and Process Development: Text Cases.* Free Press, 2010 http://books.google.de/books?id=awmav051w24C. – ISBN 9781451602319