AutoUni – Schriftenreihe

Fabian Sobiech

Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum

AutoUni – Schriftenreihe

Band 95

Herausgegeben von/Edited by
Volkswagen Aktiengesellschaft
AutoUni

Die Volkswagen AutoUni bietet Wissenschaftlern und Promovierenden des Volkswagen Konzerns die Möglichkeit, ihre Forschungsergebnisse in Form von Monographien und Dissertationen im Rahmen der „AutoUni Schriftenreihe“ kostenfrei zu veröffentlichen. Die AutoUni ist eine international tätige wissenschaftliche Einrichtung des Konzerns, die durch Forschung und Lehre aktuelles mobilitätsbezogenes Wissen auf Hochschulniveau erzeugt und vermittelt.

Die neun Institute der AutoUni decken das Fachwissen der unterschiedlichen Geschäftsbereiche ab, welches für den Erfolg des Volkswagen Konzerns unabdingbar ist. Im Fokus steht dabei die Schaffung und Verankerung von neuem Wissen und die Förderung des Wissensaustausches. Zusätzlich zu der fachlichen Weiterbildung und Vertiefung von Kompetenzen der Konzernangehörigen, fördert und unterstützt die AutoUni als Partner die Doktorandinnen und Doktoranden von Volkswagen auf ihrem Weg zu einer erfolgreichen Promotion durch vielfältige Angebote – die Veröffentlichung der Dissertationen ist eines davon. Über die Veröffentlichung in der AutoUni Schriftenreihe werden die Resultate nicht nur für alle Konzernangehörigen, sondern auch für die Öffentlichkeit zugänglich.

The Volkswagen AutoUni offers scientists and PhD students of the Volkswagen Group the opportunity to publish their scientific results as monographs or doctor’s theses within the “AutoUni Schriftenreihe” free of cost. The AutoUni is an international scientifc educational institution of the Volkswagen Group Academy, which produces and disseminates current mobility-related knowledge through its research and tailor-made further education courses. The AutoUni‘s nine institutes cover the expertise of the different business units, which is indispensable for the success of the Volkswagen Group. The focus lies on the creation, anchorage and transfer of knew knowledge.

In addition to the professional expert training and the development of specialized skills and knowledge of the Volkswagen Group members, the AutoUni supports and accompanies the PhD students on their way to successful graduation through a variety of offerings. The publication of the doctor’s theses is one of such offers. The publication within the AutoUni Schriftenreihe makes the results accessible to all Volkswagen Group members as well as to the public.

Herausgegeben von/Edited by
Volkswagen Aktiengesellschaft
AutoUni
Brieffach 1231
D-38436 Wolfsburg
http://www.autouni.de

Fabian Sobiech

Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum

Springer

Fabian Sobiech
Wolfsburg, Deutschland

Zugl.: Dissertation TU Clausthal, 2016

Die Ergebnisse, Meinungen und Schlüsse der im Rahmen der AutoUni Schriftenreihe veröffentlichten Doktorarbeiten sind allein die der Doktorandinnen und Doktoranden.

AutoUni – Schriftenreihe
ISBN 978-3-658-16327-3 ISBN 978-3-658-16328-0 (eBook)
DOI 10.1007/978-3-658-16328-0

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer ist Teil von Springer Nature
Die eingetragene Gesellschaft ist Springer Fachmedien Wiesbaden GmbH
Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Für meine
Familie

Danksagung

In den letzten Jahren haben mich viele Menschen auf meinem Weg begleitet und ihren Beitrag zu dieser Arbeit geleistet. An dieser Stelle möchte ich mich bei all diesen Personen bedanken.

Ich bedanke mich bei meinem Doktorvater Prof. Dr. Andreas Rausch für die kontinuierliche Unterstützung und für das Stellen der richtigen Fragen zum richtigen Zeitpunkt. Durch diese Impulse konnte die Forschung in dieser Arbeit auch in kritischen Momenten immer wieder in die richtigen Bahnen gelenkt werden und beständige Fortschritte erzielen. Dies ist ein wesentlicher Baustein für das Gelingen dieser Arbeit.

Ich danke auch Herrn Prof. Dr. Jörg P. Müller und auch Herrn Prof. Dr. Urs Andelfinger für die kurzfristige Übernahme der weiteren Gutachten sowie Herrn Prof. Dr. Sven Hartmann für die Übernahme des Vorsitzes der Prüfungskommission.

Frau Dr. Beate Eilermann danke ich für die Unterstützung während der ganzen Erstellungsphase, insbesondere für die wissenschaftlichen Anregungen zu Beginn und zum Abschluss dieser Arbeit.

Außerdem gilt mein Dank Herrn Hans-Christian Heidecke, der die Entstehung dieser Arbeit initial ermöglicht hat. Auch möchte ich mich bei Herrn Jan Wipke und Herrn Francisco Bravo Gomez für das mir entgegengebrachte Vertrauen und für das Einräumen ausreichender Freiräume bedanken. In diesem Kontext möchte ich mich auch bei allen internen und externen Kollegen in der K-SIPB-5/3 bedanken. Mein besonderer Dank gilt in diesem Zusammenhang Herrn Andreas Bleil für die vielen wertvollen Diskussionen rund um Theorie und Praxis. Außerdem haben sich die Herren S. Gipperich, S. Truthe & A. Wurster einen besonderen Dank verdient. Jeder von ihnen hat auf seine Art und Weise zum Gelingen dieser Arbeit beigetragen.

Dem gesamten Lehrstuhl von Herrn Prof. Dr. Andreas Rausch mit seinen internen und externen Doktoranden danke ich, dass ich mich trotz meiner Stellung als externer Doktorand, als Teil der Gemeinschaft gefühlt habe. Besonders danke ich Herrn Dr. Stefan T. Rühl für die „Tüfteleien“ im Detail.

Abschließend danke ich meinen Eltern Hans-Jürgen und Monika für die Korrekturen und die fortwährende Unterstützung auf meinem gesamten Bildungsweg, meiner Freundin Martina Meissner für ihr Verständnis und die eingeräumten Freiräume sowie der restlichen Familie Meissner und meinen Freunden.

Vielen Dank!

Fabian Sobiech

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Zusammenfassung

Der Bereich der Softwareentwicklung innerhalb industrieller Unternehmen und Konzernen ist in einem stetigen Wandel begriffen. In den IT-Abteilungen innerhalb dieser Organisationen müssen sowohl neue Entwicklungen in der Technik als auch sich ändernde Prozesse innerhalb des Unternehmens berücksichtigt werden.

Durch die Einführung agiler Softwareentwicklungsmethoden sollen die Innovationszyklen verkürzt und die internen Kunden und Anwender mehr in den Mittelpunkt der Entwicklung gestellt werden. Ein wichtiger Erfolgsfaktor für interne IT-Projekte ist, die „richtigen“ Anforderungen zur richtigen Zeit umzusetzen. Deswegen liegt der Schwerpunkt dieser Forschungsarbeit auf der Verbesserung der Planung von agilen Iterationen, welche in Scrum „Sprints“ heißen, bei der Softwareentwicklung und den damit verbundenen Optimierungspotenzialen.

Es wird zum einen gezeigt, wie Nutzwerte für Anforderungen, im Umfeld agiler Softwareentwicklung, mit mehreren heterogenen Kunden bestimmt werden können. Hierfür werden bestehende Verfahren für die Nutzwertbestimmung verwendet und für die konkrete Problemstellung erweitert und angepasst.

Zum anderen wird eine Lösung erarbeitet, die es ermöglicht Synergiepotenziale zwischen Anforderungen abzubilden. Diese Potenziale können bei der gemeinsamen Betrachtung mehrerer Anforderungen entstehen und genutzt werden. Doppelentwicklungen ähnlicher Funktionalitäten und nachträgliche Anpassungen an Schnittstellen können dadurch verringert werden. Die hierfür notwendigen Änderungen bei der Dokumentation von Anforderungen und der Iterationsplanung werden ebenfalls dargestellt.

Aufbauend auf diesen Ergebnissen wird ein mathematisches Modell zur Beschreibung von validen Iteartionen erarbeitet. Dieses Modell wird anschließend in ein Optimierungsproblem überführt. Zur Lösung dieses neuen Problems werden unterschiedliche Algorithmen bzgl. ihrer Anwendbarkeit auf dieses Problem analysiert. Das Ziel ist es, die Projektverantwortlichen bei der Iterationsplanung zu unterstützen und die sich ergebenden Potenziale bestmöglich auszuschöpfen.

Die Ergebnisse des algorithmischen Vorgehens werden mit den Ergebnissen von Projektverantwortlichen verglichen. Die jeweiligen Auswertungen zeigen, dass durch ein algorithmischen Vorgehens zur Sprintplanung sowohl Zeiteinsparungen entstehen als auch eine Steigerung des Nutzwertes je Iteration erreicht wird.

1 Einleitung

Die agilen Vorgehensweisen, wie Scrum, werden für die Softwareentwicklung im industriellen Kontext immer bedeutender. Allerdings gibt es bei der Einführung und Umstellung auf agile Vorgehensweisen immer noch nur teilweise gelöste Probleme, wie z. B. die Skalierbarkeit dieser Methoden. Ursprünglich wurden meist kleinere Entwicklerteams mit 5-7 Entwicklern betrachtet. Die Übertragbarkeit auf industrielle IT-Großprojekte stand dabei nicht direkt im Fokus (Rising und Janoff 2000).

Das Anliegen dieser Arbeit ist nicht, das hier angesprochene Problem oder andere damit verwandte Probleme zu lösen. Das Ziel ist, die sich aus der Skalierung ergebenden Potenziale zu nutzen. Hierfür werden insbesondere Anforderungen unterschiedlicher Kunden analysiert, sich ergebende Synergiepotenziale systematisch abgebildet und in ein Modell zur optimierten Abarbeitung von Anforderungen überführt.

In dieser Einleitung wird zunächst das Thema weitergehend motiviert. Anschließend wird eine Übersicht über die Ziele und Ergebnisse dieser Arbeit gegeben. Abschließend wird der Aufbau dieser Arbeit schematisch skizziert.

1.1 Motivation

Die Automobilindustrie ist geprägt durch kontinuierliche Innovationen und Verbesserungen bestehender Funktionen von einer Modellgeneration zur nächsten. Neben den bestehenden, kürzer werdenden Innovationszyklen innerhalb von Modelllebenszyklen, in denen IT-Systeme und IT-gestützte Funktionen einen immer größeren Stellenwert einnehmen, steht die IT auch vor weiteren Herausforderungen, die im Einklang mit dem globalen Wachstum stehen (Mühleck, K. H. & Heidecke, H.-C. 2012; Rudow und Heidecke 2014).

Neben der Komplexität der im Fahrzeug eingesetzten Software steigen auch die Komplexität und die Anforderungen an IT-Systeme, welche die Prozesse innerhalb eines globalen Produktionsnetzwerkes unterstützen. Dies betrifft Systeme, die für die operative Steuerung einer Fabrik oder eines Produktionsnetzwerks benötigt werden. In diesem Zusammenhang ergeben sich auch weiterführende Anforderungen an Kennzahlen und Berichtssysteme. Diese Systeme, die ganz grob dem Bereich der Business Intelligence (BI) zugeordnet werden (Eilermann 2013), rücken ebenfalls immer mehr in den Fokus. Durch die Möglichkeit aktuellere und größere Datensätze aus der Produktion erhalten und verarbeiten zu können, steigen auch die Anforderungen an die Aktualität der Berichte. Somit ist es notwendig neuartige Auswertungen bereitzustellen, um unterschiedliche Daten aus unterschiedlichen Quellen miteinander zu verbinden. Diese verbundenen Daten bilden die Grundlage für neue Berichte und Analysemöglichkeiten.

In diesem Umfeld sind somit die steigenden Anforderungen an die Berichtslegung durch Kunden aus unterschiedlichen Standorten und Abteilungen umzusetzen. Neben diesen Anforderungen ist es innerhalb der IT ebenfalls notwendig neue Techniken und Technologien zu evaluieren, um mit den steigenden Anforderungen auf Seite der Kunden mithalten zu können und diese unter anderem auch mit neuen Technologien umzusetzen.

Für IT-Projekte mit begrenzten Ressourcen ist es deswegen problematisch die Anforderungen unterschiedlicher Kunden umzusetzen und zeitgleich auch genügend Kapazität für z. B. die Prüfung neuer Technologien aufzuwenden. Je nach Finanzierungsmodell der internen IT und

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0_1

der Bereitschaft der Kunden für Innovationen zu zahlen, ergeben sich teilweise kaum Freiräume für Innovation. Ein möglicher Ausweg aus dieser Situation könnte eine Optimierung entlang des Entwicklungsprozesses darstellen, sodass Anforderungen „effizienter“ umgesetzt werden und frei werdende Kapazität zielgerichtet für innovative Tätigkeiten verwendet werden kann.

Unter der Annahme, dass die eingesetzten Entwickler, die in einem oder mehreren Teams an einem System arbeiten, nicht ohne Weiteres Anforderungen schneller umsetzen können, ist es notwendig bestehende Anforderungen mit einem geringeren Aufwand umzusetzen. Um diese Möglichkeit zu erschließen, wird es an dieser Stelle nicht ausreichen einzelne Anforderungen separat zu betrachten. Vielmehr sollte versucht werden, durch die gemeinsame Betrachtung mehrerer Anforderungen Mehrfachentwicklungen und nachträgliche Anpassungen zu reduzieren.

In Situationen mit mehreren Kunden, die sich nicht miteinander abstimmen, wie es z. B. bei konzernweit eingesetzten Managementinformationssystemen der Fall sein kann, ergeben sich nutzbare Potenziale bei der gemeinsamen Betrachtung mehrerer Anforderungen. Diese Potenziale werden durch eine anschließende Konsolidierung unter der Ausnutzung von Synergieeffekten zwischen diesen Anforderungen nutzbar.

Diese Art der Konsolidierung und Nutzung von Synergieeffekten ist an dieser Stelle keine vollkommen neuartige Idee. Innerhalb großer IT-Projekte werden solche Betrachtungen bereits durchgeführt. Um diese Potenziale aber effizient und systematisch nutzen zu können, fehlt zurzeit die prozesstechnische Unterstützung innerhalb von agilen Softwareentwicklungsmethoden. Es fehlen Möglichkeiten, solche Sachverhalte mit bestehenden Methoden, z. B. innerhalb eines Produkt Backlogs, zu dokumentieren und anschließend in einen oder mehrere Sprint Backlogs zu überführen, sodass diese Potenziale auch tatsächlich genutzt werden können.

Diese Arbeit ist durch den Bedarf einer effizienten Abarbeitung von Anforderungen motiviert. Durch eine verbesserte Umsetzung sollen sich auch Freiräume für Innovation ergeben. Eine weitere Motivation ergibt sich durch die Beobachtung, dass Synergieeffekte nicht systematisch genutzt werden, um dieses Ziel zu erreichen, obwohl genau diese Potenziale einen vielversprechenden Ansatz für weitere Betrachtungen darstellen.

Eine weitere, teilweise von dieser Diskussion losgelöste Motivation ist, sich im Forschungsbereich agiler Softwareentwicklung nicht nur mit den Herausforderungen und Problemen z. B. bzgl. der Skalierbarkeit für Großprojekte oder mit dem Übergangsprozess von klassischer hin zu agiler Entwicklung zu beschäftigen (vgl. bspw. (Misra et al. 2009; Sureshchandra und Shrinivasavadhani; Nerur et al. 2005)), sondern mit dieser Arbeit einen ersten Schritt weg von der „Problemforschung“ hin zur „Innovations- und Möglichkeitsforschung“ zu leisten. Dieses kann ebenfalls durch agile Vorgehensmodelle ermöglicht werden.

1.2 Ziele und Ergebnisse

Die Ziele dieser Arbeit leiten sich aus den Beschreibungen der Motivation ab. Das Hauptziel dieser Arbeit ist es Gemeinsamkeiten bzw. Synergien zwischen Anforderungen unterschiedlicher Kunden an ein IT-System innerhalb der IT-Entwicklung nutzbar zu machen. Durch eine systematische Nutzung dieser Potenziale sollen Aufwände bei einer agilen Umsetzung reduziert und Umsetzungskosten gesenkt werden.

Um die zusätzlichen Aufwände für die Identifikation und Dokumentation dieser Potenziale zu leisten und diese auch in der Entwicklung nutzen zu können, ist es notwendig die Sprintplanung entsprechend anzupassen. Ein Ziel für eine angepasste Iterationsplanung ist ebenfalls die jeweiligen Projektverantwortlichen bei der Planung zu unterstützen. Somit ergeben sich die folgenden weiteren Ziele:

Nutzung von Gemeinsamkeiten bzw. **Synergien** zwischen Anforderungen innerhalb eines agilen Vorgehenmodells sowie die Bereitstellung entsprechender Möglichkeiten um diese **abzubilden**.

Durch die systematische Identifikation und Nutzung von Synergiepotenzialen ergeben sich Anpassungsbedarfe an die Iterationsplanung. Ein weiteres Ziel ist, die Projektverantwortlichen bei der **Iterationsplanung** (teil-)automatisiert zu **unterstützen**.

Diese Arbeit beinhaltet drei relevante Zwischenergebnisse, die zu einem ganzheitlichen Lösungskonzept führen. Die relevanten Ergebnisse dieser Arbeit sind:

Erweiterung einer Methode zur Quantifizierung des Nutzwertes von IT-Anforderungen

Hierfür werden unterschiedliche Studien mit Prozessbeteiligten durchgeführt. Es werden relevante Nutzwertdimensionen bestimmt und gewichtet. Abschließend werden die Ergebnisse in eine aggregierbare Form überführt. Durch diese Methode kann der Nutzwert von Anforderungen einheitlich und transparent bestimmt werden. Der Nutzwert von Anforderungen wird im Verlauf dieser Arbeit auch als Optimierungskriterium verwendet.

Erweiterung: Product Owner Stories als neues Backlog-Element

Die Erweiterung des Ansatzes von Epics, User Stories und Aufgaben (Tasks) um ein weiteres Element ist notwendig, um gegebene Limitationen bzgl. der Möglichkeiten zur Abbildung von Synergiepotenzialen zu überwinden. Neben der reinen Abbildung wird auch erläutert, wie sich dieses in die Scrum-Methodik integrieren lässt.

Ableitung eines Modells zur Beschreibung und Optimierung von Iterationen

Anschließend wird aus den bestehenden Ergebnissen und Beobachtungen aus dem industriellen Kontext ein mathematisches Modell zur Beschreibung von Iterationen abgeleitet. Dieses Modell erlaubt es Synergiepotenziale zwischen Anforderungen zu nutzen, eine theoretisch beliebige Anzahl an Entwicklerteams in eine Iteration einzubinden und deren jeweilige Fähigkeiten bei der Planung zu berücksichtigen. Darauf aufbauend wird ein Optimierungsproblem definiert, dessen Lösung zur Unterstützung der Projektverantwortlichen bei der Iterationsplanung verwendet werden kann. Für die Lösung dieser Problemstellung werden unterschiedliche Lösungsansätze untersucht und miteinander verglichen. Die Ergebnisse dieser Verfahren werden zum einen durch zufällige Probleminstanzen miteinander verglichen. Zum anderen werden Ergebnisse von Projektverantwortlichem vergleichend herangezogen. Abschließend wird das Verfahren auch auf reale Iterationen angewendet.

Die in dieser Arbeit präsentierten Ergebnisse, stehen größtenteils auch in Verbindung mit realen Problemstellungen aus dem Umfeld, in dem diese Forschung stattfand. Entsprechend sind einige Ergebnisse auf einen speziellen Kreis von Projektbeteiligten zugeschnitten. Eine Generalisierung, die Interpretation bestimmter Ergebnisse und deren Übertragung auf andere Domänen ist ebenfalls Bestandteil dieser Arbeit.

1.3 Aufbau der Arbeit

Im Folgenden werden der Aufbau der Arbeit sowie die Inhalte der Kapitel kurz vorgestellt.

Weiterführend zur Motivation aus Kapitel 1.1 wird in Kapitel 2 die Problemstellung weiter detailliert und in ihren industriellen Kontext eingebettet. Aufbauend auf einer Beschreibung des Umfeldes, in dem diese Arbeit entstanden ist, werden die konkreten Problemstellungen abgeleitet.

Anschließend wird in Kapitel 3 Scrum als Vorgehen für agiles Projektmanagement vorgestellt und erläutert. Dieses Rahmenwerk bildet dabei eine wichtige Grundlage für die Arbeit. Mit den in diesem Kapitel vermittelten Grundlagen soll auch die Einordnung der aktuellen Forschungsergebnisse erleichtert werden.

Darauf folgend wird in Kapitel 4 der aktuelle Stand der Forschung bzgl. der identifizierten Defizite, Potenziale und möglichen Lösungsrichtungen vorgestellt. Die Hauptpunkte sind zum einen die Anwendbarkeit von Scrum auf mehrere Teams innerhalb eines Projekts, zum anderen Methoden zur Nutzwertbestimmung sowie verwandte Problemstellungen und Lösungen.

Anschließend wird ein erster Lösungsansatz vorgestellt, welcher sich aus den identifizierten Fragestellungen und aus dem aktuellen Stand der Forschung ableitet.

In den Kapiteln 6 - 1 werden einzelne Aspekte der gesamtheitlichen Lösung hergeleitet und beschrieben. Eines der Hauptergebnisse ist dabei in Kapitel 1 dargestellt. Auf der Grundlage der vorherigen Untersuchungen und Ergebnisse wird ein mathematisches Modell für die Beschreibung eines „validen“ Sprints abgeleitet. In Kombination mit dem klassischen agilen Ziel der Nutzwertoptimierung mit jeder Iteration ergibt sich ein algorithmisch komplexes Problem. Für die Lösung des Problems werden unterschiedliche Lösungsansätze evaluiert und miteinander verglichen.

In Kapitel 1 wird anhand einer beispielhaften Sprintplanung gezeigt, dass die erarbeitete algorithmische Lösung im Vergleich zu IT-Projektleitern wesentlich bessere Ergebnisse liefert und eine Sprintplanung mit weniger Zeitaufwand durchgeführt werden kann. In dieser Erprobung werden auch die notwendigen Änderungen an der Sprintplanung mit berücksichtigt. Auch wird gezeigt, dass dieses Vorgehen auf reale Probleminstanzen anwendbar ist.

Abschließend werden die Ergebnisse in einem Fazit zusammengefasst. Neben der Zusammenfassung der Ergebnisse wird auch ein Ausblick für weitere Forschungen in diesem Themenkomplex gegeben.

2 Problemstellung

In diesem Kapitel wird, weiterführend zur Einleitung in Kapitel 1, das Umfeld, in dem diese Arbeit entstanden ist, genauer illustriert. Aufbauend auf einer Beschreibung des Projektumfelds in Kapitel 2.1, in dem ein standardisiertes Berichtssystem für die Produktion durch die interne IT entwickelt wird, wird in Kapitel 2.2 eine Beschreibung der bestehenden und sich im Einsatz befindlichen Projektmanagementmethoden für die Entwicklung von Softwareprodukten gegeben. In Kapitel 2.3 werden dann, abgeleitet aus der Beschreibung des Umfeldes sowie aus der Beschreibung der aktuell im Projektmanagement eingesetzten Methoden kontextbezogene Defizite anhand von Beispielen dargestellt und offene Forschungsfragen identifiziert.

2.1 Beschreibung des Umfeldes

Bei global agierenden Konzernen innerhalb der Automobilindustrie ist die operative Steuerung der Produktion sowie der an der Produktion beteiligten Werke und Gewerke nur über einen kontinuierlichen Informationsfluss zu gewährleisten (Neubauer und Rudow 2012). Eine wesentliche Voraussetzung für die operative Steuerung sind valide und aktuelle Informationen aus allen am Produktentstehungsprozess beteiligten Gewerken der unterschiedlichen Werke.

Für die IT-gestützte Datensammlung, deren Aufbereitung sowie die Darstellung für den Endanwender, sind IT-Lösungen aus dem Bereich der Business-Intelligence (BI) spezialisiert (Kemper, H.G., Baars, H. & Mehanna W. 2010). Durch die zunehmende Globalisierung der Produktion und die Integration neuer Produktionsstandorte in das bestehende Produktionsnetzwerk ergibt sich für die interne IT die Herausforderung, diese neuen Produktionsstandorte mit in die bestehende Berichtssystematik zu integrieren und für eine Standardisierung zu sorgen (Mühleck, K. H. & Heidecke, H.-C. 2012). Dabei ist es auch erforderlich, entsprechende Besonderheiten der unterschiedlichen Standorte und der unterschiedlichen organisatorischen Einheiten eines Standortes mit zu berücksichtigen, um Prozesse vor Ort bestmöglich durch ein Berichtssystem unterstützen zu können.

Neben den Anforderungen unterschiedlicher Produktionsstandorte ergeben sich auch eine Vielzahl an Anforderungen für Neu- bzw. Weiterentwicklungen innerhalb eines Standortes durch die voneinander abweichenden Bedürfnisse unterschiedlicher organisatorischer Einheiten sowie durch eine abweichende Granularität bei der Berichtslegung auf unterschiedlichen Führungsebenen. Ein Konzern in der Automobilindustrie kann bspw. in acht unterschiedliche Hierarchieebenen unterteilt werden. Diese Ebenen werden in Tabelle 1 dargestellt und kurz beschrieben.

Auf der Ebene der Schicht bestehen bspw. für einen Meister andere Anforderungen an die Granularität der Auswertungen über die „Leistung“ als auf der Management- oder Vorstandsebene. Hier liegen Informationen meist nur in stark aggregierter Form vor. Diese Unterschiede bei der Granularität lassen sich z. B. dadurch begründen, dass auf den Ebenen der Schicht bzw. Kostenstelle oder auch Unterabteilung der Fokus der Arbeit meist stark auf operative Tätigkeiten ausgerichtet ist, wohingegen auf den höheren Hierarchieebenen das Operative immer mehr durch eine strategische und planerische Komponente ersetzt wird.

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0_2

Tabelle 1: Hierarchieebenen eines Automobilkonzerns

Hierarchieebene	Beschreibung
1. Ebene (Konzernvorstand)	Kennzeichnung der Systemgrenze durch internationale Zusammenarbeit und Aufgabenteilung; Informationen liegen in stark aggregierter Form vor
2. Ebene (Markenvorstand)	Zusammenwirken verschiedener Unternehmen mit einer gewachsenen Arbeitsteilung; Informationen liegen in stark aggregierter Form vor
3. Ebene (Geschäftsfeldleitung)	Arbeitsprozesse innerhalb einer Organisation, geschäftsfeldsspezifisch verteilt; Informationen liegen in aggregierter und detaillierter Form vor
4. Ebene (Werkleitung)	Arbeitsprozess innerhalb eines Werkes; Informationen liegen in aggregierter und detaillierter Form vor
5. Ebene (Abteilung, Unterabteilung)	Arbeitsstätte, in der der Arbeitsprozess erfolgt; Informationen liegen detailliert für den Verantwortungsbereich vor
6. Ebene (Kostenstelle)	Abteilungsgruppen, die eine Arbeitsstätte bilden; Informationen liegen detailliert für den Verantwortungsbereich vor
7. Ebene (Schicht)	Systemeinheit, in der ein Mensch tätig sein kann; ausgewählte Informationen
8. Ebene (Werker)	Einzelner Arbeitsplatz; ausgewählte Informationen

Quelle: Eigene Darstellung in Anlehnung an (Eilermann 2013) und (Martin 1994)

Für eine unternehmensinterne IT, die für die (Weiter-)Entwicklung eines solchen standardisierten Berichtsystems verantwortlich ist, bedeutet dies, die Anforderungen der unterschiedlichen Standorte über die unterschiedlichen Hierarchieebenen hinweg zu sammeln, in Zusammenarbeit mit den jeweilig verantwortlichen organisatorischen Einheiten zu spezifizieren und bzgl. ihrer Umsetzbarkeit hin zu bewerten. Anschließend müssen die unterschiedlichen Anforderungen priorisiert und in die Softwareentwicklung eingeplant werden.

Außerdem zeigt sich, dass nicht nur die Produktion von Automobilen innerhalb eines globalen Produktionsnetzwerks vollzogen wird, auch die IT-Branche sieht sich mit einer stetig wachsen-

den Globalisierung konfrontiert. Wegen des Bedarfs an spezifischen Entwicklungskompetenzen, wirtschaftlichen Rahmenbedingungen und internationaler Entwicklungskooperationen werden Softwaresysteme teilweise auch mit geografisch verteilten Entwicklerteams realisiert.[1] Dieses stellt die IT vor neue Herausforderungen bzgl. der Verteilung von Entwicklungsaufgaben und der Integration der verteilten Entwicklungskompetenzen, um ein Softwaresystem zu entwickeln.

Um den Bedarf an spezifischen Entwicklungskompetenzen decken zu können sowie wirtschaftlich nachhaltig zu handeln, gibt es bei der Volkswagen AG sogenannte „Regional Competence Center" (RCC), bspw. in Indien. In diesen Kompetenzzentren werden bestimmte Entwicklungskompetenzen innerhalb des Volkswagenkonzerns vorgehalten, die bei Bedarf abgerufen werden können. Kooperationen mit RCCs sollen eine kosteneffiziente Umsetzung der zuvor in Deutschland mit den Fachbereichen erhobenen und spezifizierten Anforderungen ermöglichen. Durch eine strategische und langfristige Kooperation soll es außerdem möglich sein, auf einen größeren Pool an Entwicklerressourcen zurückzugreifen und die vorhandenen Kapazitäten dynamisch an die Projektauslastung anzupassen. Dabei kann in RCCs auf bereits vorhandenes Wissen aus bereits durchgeführten Projekten innerhalb von Volkswagen zurückgegriffen werden. Somit kann sichergestellt werden, dass Prozesswissen und Know-how im Konzern verbleiben und für Projekte unterschiedlicher IT-Abteilungen genutzt werden können.

Zusammenfassend stellt sich die Umgebung, in der ein standardisiertes Berichtssystem für Volkswagen entwickelt wird, so dar, dass es eine Vielzahl unterschiedlicher Anforderer mit unterschiedlichen Anforderungen über die unterschiedlichen Produktionsstandorte hinweg gibt. Außerdem müssen auch die unterschiedlichen Anforderer der verschiedenen organisatorischen Einheiten, innerhalb eines Produktionsstandortes, mit ihren divergierenden Anforderungen berücksichtigt werden. Innerhalb eines Standortes ergeben sich die unterschiedlichen Anforderer aus den Vertretern der unterschiedlichen organisatorischen Einheiten, mit auf den Geschäftsprozess abgestimmten Anforderungen.

Durch die Vielzahl an Anforderungen und den sich daraus ableitenden benötigten Entwicklungskompetenzen, sowie durch Entwicklungskooperationen in unterschiedlichen Ländern, werden nicht alle Anforderungen vollumfänglich an einem Standort umgesetzt. Unterschiedliche Kompetenzen unterschiedlicher Entwicklerteams sind zu integrieren, um umsetzbare Anforderungen zeitnah umzusetzen und in das Softwareprodukt einfließen zu lassen, z. B. durch ein neues Release.

2.2 Projektmanagement innerhalb der IT

Um schnell auf sich ändernde Anforderungen sowie Änderungen in der Priorisierung reagieren zu können, erhalten agile Entwicklungsmethoden immer mehr Aufmerksamkeit innerhalb von IT-Abteilungen von Unternehmen der „old Economy" (Petersen 2010; Derbier 2003). Ein weiterer Vorteil agiler Entwicklungsmethoden ist die Möglichkeit, die Komplexität von vielfältigen, sich ändernden Anforderungen besser beherrschen zu können. Eine der bekanntesten agilen Projektmanagement-Methoden ist Scrum (Hossain et al. 2009). Diese Methodik wird in

[1] Das System „Berichtswesen Fabriksteuerung" der Volkswagen AG ist ein Beispiel für eine solche Koopertaion.

Kapitel 3 wegen der hohen Bedeutung für diese Arbeit genauer erläutert. Um jedoch die aktuelle Situation bzgl. des Projektmanagements innerhalb der Konzern-IT für die Erstellung eines standardisierten Berichtssystems besser nachvollziehen zu können, erfolgt an dieser Stelle eine kurze Einführung in die Rolle des Product Owners. Die Rolle des Product Owners ist innerhalb von Scrum die wichtigste Rolle, welche mit einer Vielzahl von Aufgaben und Pflichten verbunden ist (vgl. Pichler 2010). In Kapitel 2.2.1 werden zunächst die klassischen Aufgaben der Rolle des Product Owner zusammengefasst. Anschließend wird in Kapitel 2.2.2 die Umsetzung und Auslegung dieser Rolle innerhalb des Projektkontextes erörtert.

2.2.1 Traditioneller Product Owner

Innerhalb von Scrum ist es die Aufgabe des Product Owners aus fachlicher Sicht die Anforderungen des Auftraggebers zu vertreten. Er ist innerhalb von Scrum derjenige, der neue Anforderungen aufnimmt und diese in Zusammenarbeit mit dem Auftraggeber definiert. Er verwaltet die Anforderungen und prüft zum Ende einer Iteration, ob die Umsetzung hinsichtlich der Funktionalität, Usability und Qualität den Anforderungen genügt. Die Besetzung der Rolle des Product Owners kann sowohl durch eine Person des Auftraggebers (Fachabteilung), als auch durch eine Person des Auftragnehmers (IT) erfolgen. Tabelle 2 stellt in kompakter Form die jeweiligen Vor- und Nachteile dar.

Tabelle 2: Vor- & Nachteile der Besetzung der Rolle des Product Owners durch Auftraggeber bzw. Auftragnehmer

	Vorteile	**Nachteile**
Besetzung durch Auftraggeber	• Interessensvertreter der Kunden • bessere Kommunikation • klare Vision bei maßgeschneiderten Lösungen	• wenig technisches Verständnis • weniger Erfahrung
Besetzung durch Auftragnehmer	• besseres technisches Verständnis • meist mehr Erfahrung • Entscheidung über die Umsetzbarkeit von Anforderungen	• geringe Kundenbindung • Kommunikationsbarriere

Quelle: Eigene Darstellung in Anlehnung an (Pichler 2010) & (Sutherland et al. 2007)

Unabhängig von der Besetzung ist der Product Owner der, der die Richtung vorgibt und dafür verantwortlich ist, dass zur richtigen Zeit die richtigen Anforderungen in spezifizierter Form vorliegen, sodass das Projekt Erfolg hat.

Zusammenfassend ergeben sich folgende Aufgaben und Verantwortlichkeiten für einen Product Owner, (vgl. (Pichler 2010), (Schwaber und Sutherland 2013a)):

- Vertretung der Anforderungen des Auftraggebers aus fachlicher Sicht
- Verwaltung der Anforderungen innerhalb des Product Backlogs
- Priorisierung der Anforderungen mit dem Ziel den Nutzwert des Produktes nach Abschluss der nächsten Iteration zu maximieren
- erster Ansprechpartner des Entwicklerteams bei Rückfragen
- erster Ansprechpartner für Stakeholder bei neuen Anforderungen bzw. bei veränderten Anforderungen

2.2.2 Die Rolle des Product Owners - Gelebte Praxis

Innerhalb der IT von Volkswagen wird der Product Owner vorwiegend durch die verantwortliche IT-Abteilung gestellt, da im Regelfall innerhalb der Fachabteilung(en) keine ausreichenden Kapazitäten vorhanden sind, um einen Vertreter teilweise oder ganz für einen längeren Zeitraum für die Wahrnehmung der Aufgaben und Pflichten des Product Owners in die IT zu entsenden.

Aufgrund der globalen Ausrichtung eines standardisierten Berichtssystems und dessen Verbreitung über mehrere Produktionsstandorte hinweg, kommt es zu einer hohen Anzahl an Neu- bzw. Änderungsanforderungen (Change Requests) an das bestehende System. Für den Product Owner innerhalb der IT bedeutet dieses, dass er die unterschiedlichen Anforderungen der unterschiedlichen Standorte und unterschiedlichen organisatorischen Einheiten auf ihre Umsetzbarkeit hin bewerten muss. Darüber hinaus liegt es innerhalb seiner Verantwortung für eine ausreichende Spezifikation der entsprechenden Anforderung zu sorgen. Ausgehend von (teil-)spezifizierten Anforderungen ist es ebenfalls in der Verantwortung des Product Owner zu einer konsolidierten Anforderungsmenge zu kommen. Eine konsolidierte Menge an Anforderungen bedeutet in diesem Zusammenhang, dass Anforderungen, welche inhaltliche Gemeinsamkeiten aufweisen, nach Möglichkeit zusammengefasst und ganzheitlich betrachtet werden sollten. Durch eine ganzheitliche Betrachtung wird die Grundlage für Standardisierung und wirtschaftlich nachhaltiges Handeln gelegt.

Da dem Product Owner auch die Verantwortung über die Priorisierung der Anforderungen obliegt, ist es seine Aufgabe den Nutzwert und ggf. andere Dringlichkeiten der unterschiedlichen Anforderungen abzuschätzen und somit zu einer Priorisierung der Anforderungen zu kommen.

Die Abschätzung des Nutzwertes von Anforderungen erfolgt häufig auf Grundlage des Erfahrungswissens des Product Owners. Product Owner mit ausreichender Erfahrung können sehr effizient den zu erwartenden Nutzen abschätzen und somit zu einer Priorisierung gelangen. In diesem Zusammenhang steht der Product Owner jedoch vor der Herausforderung, diese Priorisierung, welche auf dem Erfahrungswissen des jeweiligen Product Owners beruht, vor den unterschiedlichen Vertretern der unterschiedlichen organisatorischen Einheiten zu vertreten.

Ausgehend von einer abgestimmten Priorisierung ist es dann die Pflicht des Product Owners die Produktrückstandselemente auszuwählen, die das Entwicklerteam sinnvoll bearbeiten kann und die zu einer Nutzwertmaximierung beim Abschluss der nächsten Iteration führen. Außerdem ist der Product Owner auch Hauptansprechpartner für das Entwicklungsteam bei Rückfragen.

Neben den „scrum-typischen“ Aufgaben hat ein Product Owner bei Volkswagen häufig auch noch Aufgaben und Verpflichtungen bzgl. des Projektbudgets, der Beauftragung externer Dienstleister und andere projektspezifische Verpflichtungen.

Aufgrund der Vielfalt an Aufgaben und der großen Anzahl an Anforderern, welches auch mit einer hohen Anzahl von Anforderungen verbunden ist, ist es aktuell für den Product Owner problematisch alle Aufgaben vollumfänglich zu erledigen.

2.3 Identifikation bestehender Defizite

Ausgehend von der Beschreibung der aktuellen Projektmanagementstruktur innerhalb der IT in Kapitel 2.2 und der dort beschriebenen Anwendung von Scrum als agile Projektmanagementmethode ergibt sich zunächst der Bedarf nach einer transparenten Methode für die Priorisierung. Damit verbunden ist auch die Betrachtung des Nutzwertes von Anforderungen, die im Kontext von Managementinformationssystemen innerhalb von Volkswagen erhoben werden. Da, wie beschrieben, die Priorisierung und die Nutzwertbestimmung häufig auf Erfahrungswissen der Product Owner und Projektverantwortlichen beruhen, würde eine einheitliche Nutzwertbestimmung, die Akzeptanz jedes Lösungsansatzes, der im Rahmen dieser Arbeit entsteht, steigern. Eine einheitliche Nutzwertbestimmung diemnt somit als Grundlage der Priorisierung und Umsetzungsplanung.

Ein weiteres Problem ist, dass eine Priorisierung von Anforderungen, die auf Erfahrungswissen beruht, häufig schlecht quantifiziert werden kann und für die Anforderer schwerer nachvollziehbar ist, als eine Priorisierung und Nutzwertbestimmung nach vorher festgelegten Kriterien. Dieser erste Schritt, welcher sich auch in der ersten Forschungsfrage am Ende dieses Kapitel wiederspiegelt, bildet die Grundlage der weiteren Forschungen.

Aufgrund des Bestrebens stets wirtschaftlich zu handeln und sich kontinuierlich zu verbessern ist es das Ziel, bestehende Anforderungen möglichst kosteneffizient umzusetzen. Eine Möglichkeit, die Effizienz bei der Implementierung der Anforderungen zu steigern, ist die bestehenden Anforderungen sinnvoll zu bündeln und die bestehenden Synergiepotenziale zwischen Anforderungen unterschiedlicher Anforderer zu nutzen. Durch die Nutzung von Synergiepotenzialen können Kosten z. B. durch eine Aufwandsreduktion bei einer gemeinschaftlichen Betrachtung mehrerer Anforderungen eingespart werden (Sobiech et al. 2014). Ein Beispiel im Kontext von Managementinformationssystemen wären zwei Anforderungen bei denen unterscchiedliche Auswertungen erstellt werden soll. Diese Auswertungen basieren jedoch teilweise auf identischen Daten die über eine Schnittstelle aus einem Drittsystem geladen werden müssen. Durch eine gemeinschaftliche Betrachtung kann sichergestellt werden, dass die Schnittstelle nicht nachträglich erweitert werden muss um ggf. fehlende Daten für eine der beiden Anforderungen beziehen zu können. Ein weiterer Punkt, der bei der Abbildung und Nutzung von Synergien zwischen Anforderungen unterschiedlicher Anforderer mit berücksicht werden sollte, ist, dass nicht alle Potenziale, die sich durch vorhandene Synergieeffekte ergeben, dem Kunden komplett transparent dargestellt werden sollen. Je nach Finanzierungsmodell der IT-Abteilung kann es auch vorteilhaft sein, die Potenziale nicht vollumfänglich an den Kunden weiterzugeben. Sie können bspw. für die Querfinanzierung anderer Projekte, für die Evaluation neuer Technologien oder Know-how-Transfers bei sich ändernden Projektrahmenbedingungen genutzt werden.

Um diesen Ansatz besser nachvollziehen zu können, wird dieser Sachverhalt an einem fiktiven Beispiel illustriert. Ausgehend von den folgenden User Stories wird gezeigt, wie eine gemeinschaftliche Betrachtung der Anforderungen zu einer Aufwands- und Kostenreduktion führen kann:

User Story 1: Als Produktionsleiter des Werkes H möchte ich die zeitliche Entwicklung der ökologischen Kennzahl Ö je Fertigungslinie auswerten können, um meine Produktion nachhaltiger gestalten und die Wirkung von Maßnahmen besser beurteilen zu können.

User Story 2: Als Werkleiter möchte ich, dass die Kennzahl Ö mit in den Standardbericht X aufgenommen wird, um über die aktuelle Zielerreichung dieser Kennzahl innerhalb der unterschiedlichen Gewerke informiert zu sein.

Wie aus den beiden User Stories zu ersehen ist, beziehen sich beide Anforderungen auf die ökologische Kennzahl Ö. In der ersten Anforderung soll diese mittels einer separaten Auswertung berichtet werden. In der zweiten Anforderung soll der aktuelle Wert dem Soll gegenübergestellt und dieser Key-Performance-Indikator (KPI) in einen bereits bestehenden Bericht integriert werden. Bei einer getrennten Betrachtung dieser beiden Anforderungen entsteht ein erheblicher Mehraufwand im Vergleich zu einer gemeinschaftlichen Betrachtung. Da davon auszugehen ist, dass bei der Umsetzung der Anforderungen eine neue Schnittstelle zwischen dem Berichtssystem und dem System geschaffen werden muss, mit dem die ökologische Kennzahl Ö ermittelt wird, könnte es bspw. passieren, dass zunächst eine Schnittstelle implementiert wird, die nur den Anforderungen einer der beiden formulierten User Stories genügt. Je nach Abarbeitungsreihenfolge könnte es vorkommen, dass entweder keine detaillierten Daten auf Produktionslinien-Ebene vorliegen (2 vor 1) oder keine Zielwerte mit über die Schnittstelle übertragen werden (1 vor 2). Beides würde dazu führen, dass eine bestehende Schnittstelle nachträglich angepasst und erweitert werden muss. Dieses sorgt sowohl innerhalb der Abteilung, welche das Berichtssystem betreut als auch bei der Gegenstelle für einen erhöhten Arbeitsaufwand. Außerdem kann sich die Umsetzung, je nach Bereitstellungsdatum der Schnittstellenerweiterung, verzögern. Deswegen ist es ein Ziel dieser Arbeit, solche „zusätzlichen“ Arbeiten durch eine gemeinschaftliche Betrachtung von Anforderungen zu minimieren. Im schlechtesten Fall könnte es sogar vorkommen, dass für jede der beiden Anforderungen eie separate Schnittstelle implementtiert wird. Eine solche Doppelentwicklung verzögert zum einen die Auslieferung und zum anderen werden vorhandene Ressourcen nicht effizient genutzt. Dieses reale Problem wird durch die zweite Forschungsfrage am Ende dieses Kapitel adressiert.

Durch die Vielzahl an spezifizierten Anforderungen vor Beginn einer neuen Iteration bzw. eines neuen Entwicklickungssprints in Scrum steht der Product Owner vor der Herausforderung, die Anforderungen so zu priorisieren und in die Entwicklung einzuplanen, dass der Zuwachs des Nutzwertes bezogen auf das System maximiert wird. Die Erstellung eines „optimalen“ Sprints ist bei kleinen Entwicklerteams und für eine sehr begrenzte Anzahl von Anforderungen gut und effizient manuell von Menschen durchzuführen. Mit einer steigenden Anzahl von Anforderungen, mehreren Entwicklerteams und ggf. bestehenden Abhängigkeiten zwischen den unterschiedlichen Anforderungen, ist ein Product Owner nicht mehr in der Lage mit einem realistischen Zeitaufwand eine optimale Iteration zu planen und die Aufgaben auf unterschiedliche Teams bestmöglich zu verteilen. Das Problem wird ggf. durch die systematische Nutzung von Synergiepotenzialen noch verstärkt, da diese ebenfalls bei der Planung mit berücksichtigt werden müssten. Dieses real bestehende Problem wird in der dritten Forschungsfrage reflektiert.

In diesem Zusammenhang ergibt sich auch die Notwendigkeit nach einer formalen Beschreibung des Problems. Hierbei sind grundsätzlich zwei unterschiedliche Aspekte zu betrachten. Zum einen muss analysiert und anschließend beschrieben werden, unter welchen Rahmenbedingungen von einer validen Sprintplanung gesprochen werden kann. Dabei müssen z. B. die bereits erwähnten Abhängigkeiten zwischen Anforderungen, wie auch Kapazitätsgrenzen von Entwicklerteams berücksichtigt werden. Ausgehend von der Beschreibung, der zu erfüllenden Eigenschaften für eine valide, umsetzbare Iterationsplanung, muss in einem weiteren Schritt die Erarbeitung und Betrachtung der Optimierungskriterien erfolgen.

Dieses Modell muss anschließend analysiert werden, um festzustellen, ob es sich bei der erarbeiteten Formalisierung um ein einfaches algorithmisches Problem handelt, welches mit einem Algorithmus mit deterministisch polynomiell beschränktem Zeitaufwand gelöst werden kann. Sollte die Analyse zeigen, dass das Problem nicht in der Komplexitätsklasse P liegt, so müssen alternative Wege erarbeitet werden, die mit beschränkter Zeit gute, fast optimale, Ergebnisse liefern können.[2]

Aus denen in diesem Kapitel identifizierten Defiziten wurden drei Forschungsfragen abgeleitet:

RQ 1: Wie kann der Nutzwert von IT-Anforderungen im Kontext von Management-Informationssystemen für die interne IT von großen Konzernen definiert werden?

RQ 2: Wie können Synergien, die zwischen Anforderungen unterschiedlicher Anforderer bestehen, in Scrum abgebildet und systematisch genutzt werden?

RQ 2.1: An welche Bedingungen ist die Nutzung von vorhandenen Synergiepotenzialen gebunden?

RQ 2.2: Welche Implikationen hat die Nutzung von Synergiepotenzialen für die Iterationsplanung im Rahmen agiler Entwicklung?

RQ 3: Wie kann die Iterationsplanung bei gleichzeitiger Betrachtung von Synergien für den Product Owner vereinfacht werden?

RQ 3.1: Wie kann ein valider und optimaler Iterationsplan mathematisch modelliert werden?

Jedes in diesem Kapitel identifizierte Defizit wird in einer eigenständigen Forschungsfrage reflektiert, im Laufe dieser Arbeit beantwortet und zu einer gesamtheitlichen Lösung zusammengefasst werden. Die zweite Forschungsfrage, welche sich mit dem Thema der Verwendbarkeit von Synergien beschäftigt, ist zusätzlich noch mit zwei weiteren Unterfragen versehen, da nebst der reinen Abbildung auch die Anwendbarkeit diskutiert werden muss. Darüber hinaus müssen hier auch die Implikationen für die Iterationsplanung mit betrachtet werden. Diese Informationen dienen als Eingangsinformationen für die Erstellung einer Lösung zur Beantwortung der dritten Forschungsfrage.

[2] Annahme: $P \neq NP$

3 Grundlagen: Agiles Projektmanagement mit Scrum

Im vorherigen Kapitel wurden ausgehend vom Umfeld erste Defizite identifiziert. Aus diesen Defiziten wurden Forschungsfragen abgeleitet. In diesem Kapitel, wird in Ergänzung zur Beschreibung in Kapitel 2.2, das agile Projektmanagement mit Scrum eingeführt. Dieses Vorgehen bildet sowohl den Rahmen, als auch eine wichtige Grundlage für die folgenden Untersuchungen. Außerdem dienen die in diesem Kapitel beschriebenen Grundlagen einer besseren Einordnung und Bewertung des aktuellen Stands der Forschung (Kapitel 4). Dabei wird im späteren Verlauf auch geprüft werden, ob ggf. bestehende Lösungen mit dem Scrumvorgehen in Einklang stehen und anwendbar sind.

3.1 Allgemeines Vorgehen in Scrum

Scrum ist eine agile Projektmanagementmethode. Bei Scrum steht der Ablauf des Projekts im Vordergrund und nicht, wie z. B. beim „Extreme Programming“ (XP), die Zusammenarbeit innerhalb des Entwicklerteams oder dessen Strukturierung (Hanser 2010). Scrum verfolgt die Idee des „Lean Managements“ und wurde in den frühen 90er Jahren von Ken Schwaber und Jeff Sutherland entwickelt und 1995 auf der OOPSLA Konferenz erstmalig präsentiert (Sutherland 1997). Beide sind Erstunterzeichner des Agilen Manifests. Das agile Manifest beschreibt durch vier Leitsätze das Ziel der „Kundenzentrierung“ und der Wertschöpfung für den Kunden in der Softwareentwicklung.

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

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

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein.“ (Kent Beck et al. 2014).

Durch dieses Manifest werden der Rahmen und die Ziele für agile Softwareentwicklung sowie agiles Projektmanagement definiert. Um die Komplexität der Softwareentwicklung beherrschen zu können und besser auf Veränderungen in den Anforderungen oder deren Priorisierung reagieren zu können, werden Softwareprodukte inkrementell und iterativ, d. h. schrittweise, entwickelt. Eine Iteration in Scrum nennt sich Sprint.

Ein Sprint beschreibt einen festen Zeitraum. In der Regel umfasst ein Sprint eine Zeitspanne von einer bis vier Wochen (Schwaber 2004). Zu Beginn des Sprints priorisiert der „Product Owner“ eine Auswahl an Aufgaben aus dem globalen „Product Backlog“, in dem sich alle Aufgaben befinden, die bis zum Ende des Projekts abgeschlossen werden müssen. Eine Auswahl dieser User Stories wird dann in das sogenannte „Sprint Backlog“ übertragen. Diese Auswahl wird vom Entwicklerteam in einem Sprintplanungstreffen in ihrem Aufwand bewertet. In die Bewertung gehen neben dem Aufwand für die Entwicklung auch noch weitere Faktoren mit ein, wie z. B. weiterer Klärungsbedarf bei noch nicht komplett spezifizierten Anforderungen. Außerdem kann durch die Aufwandsschätzung ermittelt werden, welche Aufgaben innerhalb des Sprints erledigt werden können und welche Aufgaben den zeitlichen Rahmen überschreiten

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0_3

und somit in einen nachfolgenden Sprint verschoben werden müssen. Die Schätzung der User Stories erfolgt mittels der abstrakten Einheit von Storypunkten. Storypunkte sind relative Werte, keine festen. Es besteht keine direkte Korrelation zwischen dem tatsächlichen Aufwand in Stunden und Punkten (Lacey 2012).

Die Zielsetzung eines Sprints sollte so gewählt werden, dass zum Abschluss eine neue Version des Produktes vorliegt, die dem Kunden vorgestellt werden kann und einen Mehrwert für den Kunden erzeugt. So hat der Kunde in regelmäßigen Abständen eine Vorabversion bei Neuentwicklungen oder erweiterte und neue Features bei Weiterentwicklungen. Dadurch ergibt sich für den Kunden auch die Möglichkeit seine Anforderungen und deren Priorisierung in regelmäßigen Abständen anzupassen, um ein aus seiner Sicht bestmögliches Produkt zu erhalten (vgl. Gloger 2009). Die Aufgaben, die innerhalb eines Sprints zu bewältigen sind, werden nicht auf die Mitglieder des Entwicklerteams verteilt, sondern jedes Teammitglied entscheidet, je nach Vorwissen und Fähigkeiten, welche der offenen Aufgaben er oder sie am besten bearbeiten kann. Dieser Ablauf wird durch die folgende Abbildung (Abbildung 1) verdeutlicht.

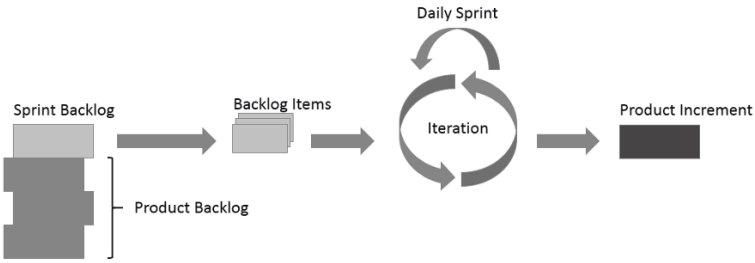


Abbildung 1: Eigene schematische Darstellung der Scrum-Vorgehensmethode nach Schwaber & Sutherland (Schwaber und Sutherland 2013b)

Neben der bereits erwähnten Rolle des Product Owners gibt es in Scrum noch weitere Rollen sowie bestimmte Arten von Treffen und weitere Artefakte. Die jeweiligen Bestandteile von Scrum werden in den folgenden Unterkapiteln jeweils kurz vorgestellt.

3.2 Rollen in Scrum

In diesem Unterkapitel werden die für Scrum typischen Rollen des ScrumMasters, des Product Owners sowie des Teams vorgestellt. Für jede dieser Rollen werden die ihr zugeordneten Aufgaben und Pflichten innerhalb des Vorgehensmodells erläutert, um ihren Einfluss auf den gesamten Prozess transparent darzustellen.

3.2.1 ScrumMaster

Der ScrumMaster hat die Aufgabe, den Scrum-Prozess zu überwachen und Probleme sowie Hindernisse zu beseitigen, um eine möglichst optimale Arbeitsumgebung zu schaffen und das Team vor äußeren Störungen zu bewahren. Es ist in der Verantwortung des ScrumMasters dafür

zu sorgen, dass die innerhalb dieser Methode vorgesehenen Meetings abgehalten und dass zuvor definierte Prozesse eingehalten werden. Der ScrumMaster bildet zusätzlich eine Art Gegenpol zum Product Owner. Es liegt in seiner Verantwortung dafür Sorge zu tragen, dass z. B. Sprintziele nicht während eines Sprints geändert werden oder zuvor für den Sprint definierte Arbeitsumfänge geändert werden. Ein weiteres Anliegen des ScrumMasters ist es, bei Störungen während der Abarbeitung der Aufgaben, z. B. durch fehlende Softwarelizenzen, fehlende Informationen, etc., diese aufzugreifen und schnellstmöglich zu beseitigen, um die Erreichung des Sprintziels nicht zu gefährden.

Je nach Größe des Entwicklerteams kann der ScrumMaster ganz oder teilweise von seinen Aufgaben als Entwickler innerhalb des Teams freigestellt werden (vgl. Gloger 2009).

3.2.2 Product Owner

Die Rolle des Product Owners ist eine zentrale Rolle innerhalb von Scrum. Nachdem bereits in Kapitel 2.2.1 eine Definition der Rolle des Product Owners erfolgt ist, werden an dieser Stelle nur die wichtigsten Aufgaben und Verantwortlichkeiten kurz dargestellt.

Der Product Owner übernimmt die Sichtweise des Endkunden. Er arbeitet mit dem Entwicklerteam zusammen und entscheidet, welche Aufgaben und Anforderungen in welchem Sprint wie umgesetzt werden sollen. Er steht dem Entwicklerteam die gesamte Zeit über als Ansprechpartner für Rückfragen zu Anforderungen zur Verfügung. Der Product Owner ist entweder ein direkter Vertreter des Kunden, es wird dann von einem On Site Customer gesprochen oder ein Mitglied des Entwicklerteams, welches die Sicht des Endkunden vertritt (Customer Proxy).

In Zusammenarbeit mit den Kunden muss der Produktverantwortliche bei der Aufnahme der Anforderungen darauf achten, diese nicht einfach nur weiterzugeben. Er muss sie bzgl. ihrer Umsetzbarkeit hin bewerten und „unerfüllbare“ Kundenwünsche direkt ablehnen oder in Zusammenarbeit mit dem Kunden so definieren, dass sie umsetzbar werden (vgl. Schwaber und Beedle 2002).

Kurz zusammengefasst übernimmt der Product Owner somit die folgenden Aufgaben innerhalb eines Projektes (vgl. z. B. (Daut 2013; Pichler 2010)):

- Priorisierung und Verwaltung der Anforderungen im Product Backlog
- Unterstützung des Kunden bei der Detaillierung und Spezifizierung von Anforderungen, sowie Klärung von Detailfragen
- Unterstützung bei Konzeption und Design
- Abnahme der User Stories im Sprint Review Meeting durch Akzeptanztests
- Stakeholder-Management

3.2.3 Scrum-Team

Das Scrum-Team oder auch kurz Team bildet den Mittelpunkt der Wertschöpfung innerhalb von Scrum. Das Team ist für die Umsetzung der Anforderungen in Produktfunktionalität und der damit verbundenen Nutzwertsteigerung des Systems verantwortlich. Ein Team sollte aus ca. 5-10 Personen bestehen (Schwaber und Sutherland 2013b). Bei größeren Projekten, welche mehr Entwicklerressourcen benötigen, sollte eine Unterteilung in mehrere miteinander koope-

rierende Teams erfolgen. Bei der Arbeit mit mehreren Teams können Teams entweder funktional unterteilt werden, d.h. ein Team ist für die Entwicklung der Datenbank verantwortlich und ein weiteres Team bspw. für die Entwicklung der Oberflächenfunktionalität oder alternativ können die Teams interdisziplinär oder auch cross-functional zusammengestellt werden, sodass ein Team immer in der Lage ist, eine Anforderung vollumfänglich umzusetzen. Die Verwendung interdisziplinärer Teams wird, sofern deren Zusammenstellung möglich und sinnvoll ist, in der Literatur als die zu präferierende Variante der Teambildung angesehen (vgl. Sutherland et al. 2007). Jedes Team hat dabei eine Eigenverantwortung für die Umsetzung der zuvor für den Sprint abgestimmten Anforderungen. Es ist die Aufgabe des Teams sich selbst zu organisieren und die Umsetzung der Aufgaben sicherzustellen, um am Sprintende ein potenziell an den Kunden auslieferbares Paket zu erzeugen. Dieses Paket wird im Sprint-Review-Meeting den Kunden und Projektverantwortlichen präsentiert und durch diese abgenommen.

3.3 Meetings

Neben den zuvor definierten und beschriebenen Rollen sieht die Scrum-Methodik einige regelmäßige Treffen und Veranstaltungen vor, die zum einen dem Wissensaustausch innerhalb des Teams dienen und zum anderen den regelmäßigen Kontakt mit dem Kunden fördern sollen. Die vorgesehenen Meetings sind das Daily Scrum, das Sprint-Planungstreffen, das Retrospektivtreffen und das Sprint-Review-Meeting. Diese Treffen sowie ihre Intentionen werden in den folgenden Unterkapiteln beschrieben.

3.3.1 Daily Scrum

Das Daily Scrum ist, wie dem Namen bereits zu entnehmen ist, ein tägliches Meeting der Entwickler, welches vom ScrumMaster geleitet wird. Ziel des Meetings ist es einen Überblick über die sich aktuell in der Bearbeitung befindlichen Aufgaben zu erhalten. Darüber hinaus dient das Meeting dem Wissensaustausch zwischen den Entwicklern, sowie zur Identifikation von etwaigen Problemen bei der Umsetzung und Abarbeitung von Aufgaben.

In diesem Meeting, welches nicht länger als 15 – 20 Minuten dauern sollte, stellt jeder Entwickler der Reihe nach kurz dar, mit welcher Aufgabe er sich gestern beschäftigt hat, mit welcher Aufgabe er sich heute beschäftigen wird und ggf., welche Probleme oder Hindernisse bei der Bearbeitung der Aufgaben aufgetreten sind. Sollten Probleme aufgetreten sein, ist es in der Verantwortung des ScrumMasters diese zu identifizieren und zu lösen (s. Kapitel 3.2.1).

3.3.2 Sprint Planning Meeting

Das Sprint Planning Meeting bildet im Regelfall den Startpunkt für einen neuen Sprint. Das Ziel dieses Meetings ist es, die Arbeitsumfänge für den folgenden Sprint festzulegen. Dabei werden die Anforderungen fixiert, die innerhalb des nächsten Sprints bearbeitet werden sollen. Wichtig ist hierbei das Wort „fixiert“. Einmal abgestimmte Arbeitsumfänge sind für den Zeitraum fixiert und können nicht mehr abgeändert werden. Somit soll sichergestellt werden, dass das Team seine Zusage über zu erwartenden Leistungen einhalten und sich auf die abgestimmte Menge an Anforderungen konzentrieren kann.

Dieses Meeting unterteilt sich in zwei Teile. Im ersten Teil des Meetings präsentiert der Product Owner dem Team die Produktrückstandselemente, meist in Form von User Stories, mit der höchsten Priorität. Basierend auf der Vorstellung der Anforderungen schätzt das Team den Aufwand des jeweiligen Produktrückstandselementes in Storypunkten.

Für die möglichen Story-Punktwerte wird z. B. ein Ausschnitt aus der Fibonaccifolge verwendet (1, 2, 3, 5, 8, 13, 21, …) (Downey und Sutherland 2013; Cohn 2006). Die Verteilung und die Vergabe der Punkte sind je Entwicklerteam unterschiedlich. Sie orientiert sich bspw. an bereits absolvierten User Stories, welche als Orientierungshilfe herangezogen werden können.

Bei der Planung geben alle Entwickler ungefähr zeitgleich eine erste Schätzung für die User Story ab. Sollten die Schätzwerte weit auseinander liegen, so muss im Anschluss dieser Unterschied von den entsprechenden Entwicklern erläutert und ggf. müssen vorhandene Unklarheiten beseitigt werden. Es folgt eine erneute Schätzung.

Der Product Owner stellt der Reihe nach weitere User Stories vor und lässt diese durch das Team hinsichtlich ihres Aufwandes bewerten bis die Kapazitätsgrenze des Teams erreicht ist. Die Kapazitätsgrenze eines Teams in Storypunkten ergibt sich meistens aus den Erfahrungen der geleisteten Umfänge der vorhergehenden Sprints. Kapazitätsgrenzen anderer Teams sind oftmals nicht übertragbar, da die Bewertung durch Storypunkte teamspezifisch erfolgt. Die Erreichung der Kapazitätsgrenze stellt den Abschluss des ersten Teils dieses Meetings dar. Das Team verpflichtet sich abschließend dazu, die bis zu diesem Zeitpunkt vorgestellten Umfänge innerhalb des nächsten Sprints umzusetzen und zu einem potenziell auslieferbaren Paket zusammenzufassen.

Im zweiten Teil dieses Meetings plant das Team selbstständig, ohne weitere äußere Einwirkungen, detailliert die Aufgaben bzw. Tasks, die notwendig sind, um die jeweiligen Anforderungen in Form von User Stories umzusetzen und somit das Sprintziel zu erreichen. Die Tasks werden zusätzlich mit einer ersten Aufwandsschätzung in Stunden versehen. Tasks sollten dabei solange verfeinert werden, bis jede einzelne Task einen Arbeitsaufwand von einem Tag für einen Entwickler nicht überschreitet. Aufgaben, welche so groß sind, dass sie in der Bearbeitung mehrere Tage oder gar Wochen benötigen sind problematisch, da der Fortschritt bei der Bearbeitung für den ScrumMaster sowie den Product Owner nur unzureichend nachvollzogen werden kann.

Diese Schätzung bildet die Grundlage für den Burndown Chart und die Nachverfolgbarkeit des Fortschritts während des Sprints. Die User Stories sowie die durch das Entwicklerteam geplanten Tasks werden abschließend zu einem neuen Sprint Backlog zusammengefasst. Dieses Backlog wird, sozusagen, eingefroren und kann nach dem Beginn der Iteration nicht mehr verändert werden.

3.3.3 Sprint Review Meeting

Das Sprint Review Meeting bildet den Abschluss eines Sprints in Scrum. Während des Meetings präsentiert das Team (live am System) die Arbeitsergebnisse des Sprints vor dem Product Owner und ggf. weiteren interessierten Stakeholdern.

Im Laufe des Meetings stellt das Team fertige und potenziell einsetzbare Funktionalität vor. An dieser Stelle sollen keine Prototypen oder PowerPoint-Folien gezeigt werden. Auf Grundlage der vorgestellten Funktionalität entscheidet der Product Owner, ob die zugehörige User Story

bzw. die zugehörigen User Stories entsprechend der Vorgaben umgesetzt worden sind. Anschließend hat der Product Owner das Recht und die Pflicht zu entscheiden, ob die gezeigte Funktionalität dem Kunden zur Verfügung gestellt werden soll, meistens durch ein Deployment einer neuen Version oder ob weitere Anpassungen und Weiterentwicklungen notwendig sind.

Die präsentierten Ergebnisse bilden darüber hinaus eine wichtige Grundlage für die Zusammenstellung der Anforderungen für den nächsten Sprint. Es ist das Ziel, möglichst frühzeitig Funktionalität für den Kunden durch das System bereitzustellen. Dadurch sollen geleistete Investitionen einen schnellen Return of Invest (ROI) bringen, bzw. soll bei einem Abbruch des Projektes vorhandene Funktionalität nutzbar bleiben und somit keinen Totalverlust darstellen (Hamilton 2008).

Ebenfalls wichtig sind Anmerkungen und Kommentare zur gezeigten Funktionalität durch den Product Owner oder weiterer Stakeholder, die dazu dienen können das System weiter zu verbessern. Diese Kommentare sollten vom Entwicklerteam aufgenommen werden und in die Arbeit mit einfließen.

3.3.4 Sprint Restropective Meeting

Das Sprint Retrospective Meeting dient dazu, den vorangegangenen Sprint methodisch zu hinterfragen und nach Möglichkeiten zur Prozessverbesserung zu suchen. Durch die Regelmäßigkeit des Meetings, kurz nach Ende einer jeden Iteration, wird versucht einen kontinuierlichen Verbesserungsprozess innerhalb von Scrum zu implementieren.

An diesem Meeting sollten das Team, der ScrumMaster sowie der Product Owner teilnehmen. Eine Möglichkeit zur Strukturierung des Meetings ist, dass jeder ähnlich zum Daily Scrum Meeting sagt, welcher Prozess oder welche Methode weitergeführt werden sollte, da sie sich aus der Sicht der jeweiligen Person als vorteilhaft erwiesen hat. Als Nächstes sollten Methoden oder Prozessausschnitte benannt werden, die keinen Vorteil für die tägliche Arbeit bringen und als überflüssig angesehen werden. Als dritter Punkt sollten bestehende Methoden- oder Prozesslücken benannt und Vorschläge gemacht werden, durch welche Neuerungen das bestehende Defizit behoben werden kann.

Solche Neuerungen können bspw. für eine oder wenige Iterationen erprobt werden. In den folgenden Sprint Retrospective Meetings kann dann mit allen Beteiligten diskutiert werden, ob sich die erhoffte Verbesserung eingestellt hat und diese Maßnahme weitergeführt oder wieder zurückgenommen werden soll.

Ein Beispiel für eine solche Maßnahme aus dem Projektumfeld bei Volkswagen war bspw. die Einführung von kleinen Icons am oberen Rand von Aufgabenkarten. Dadurch kann ein Entwickler auf einen Blick erkennen, ob die Aufgabe eher in den Bereich Java- oder Datenbankentwicklung gehört, ohne die gesamte Aufgabe lesen zu müssen. Die nachfolgende Abbildung 2 zeigt zwei unterschiedliche fiktive Aufgabenkarten, eine mit einem Java-Symbol und eine mit einer entsprechenden Abbildung, um diese Aufgabe für den Bereich der Datenbankentwicklung zu kennzeichnen. Diese sind auf dem Foto jeweils rot umrandet.

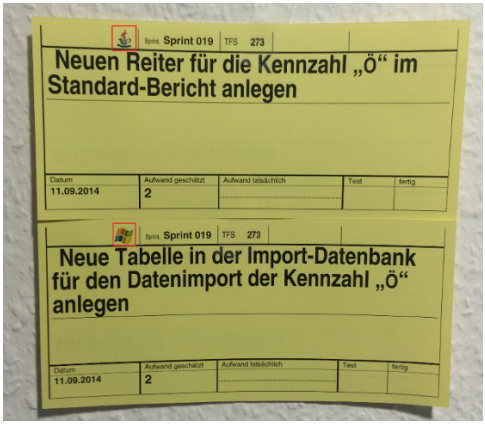


Abbildung 2: Fiktive Aufgabenkarten mit Bereichssymbolen (Quelle: eigenes Foto)

3.3.5 Backlog Grooming

Das Backlog Grooming oder auch Backlog Refinement ist kein Treffen, welches in dem von Schwaber und Sutherland ursprünglich erarbeiteten Scrum-Vorgehensmodell etabliert ist, dennoch wird es häufig als gleichberechtigtes Meeting mit in den Sprintzyklus aufgenommen. Das Ziel dieses Meetings ist es, das Produkt Backlog zu „pflegen“, dieses bedeutet, dass bspw. aus bestehenden Epen User Stories abgeleitet werden oder aber auch bestehende User Stories weiter verfeinert oder ergänzt werden.

Außerdem können User Stories bereits mit ersten Aufwandsschätzungen versehen und Unklarheiten beseitigt werden. Deswegen sollten an diesem Termin sowohl das Entwicklerteam, der Scrum Master und der Product Owner teilnehmen. Da bei der Volkswagen AG jedoch die Rolle des Product Owners und die Kommunikation zum Team hin in der Regel von der IT übernommen wird, nehmen an diesem Termin auch Interessenvertreter aus den unterschiedlichen Fachbereichen unterschiedlicher Werke teil. Dieses dient der Unterstützung der weiteren Spezifizierung von Anforderungen innerhalb des Product Backlogs. Durch ein regelmäßig durchgeführtes Backlog Grooming vor dem nächsten Sprintplanungstreffen können somit bereits Vorbereitungen getroffen werden, sodass sich das eigentliche Planungsmeeting verkürzen lässt, da Schätzungen und Rückfragen bereits im Vorfeld behandelt werden können (Rubin 2012; Cardinal 2013).

3.4 Artefakte in Scrum

Innerhalb von Scrum sind Artefakte Ergebnisse des Softwareentwicklungsprozesses. Zusätzlich zu Softwareartefakten, auch „Product Increment“ genannt, welche eine potenziell auslieferbare Software darstellen, gibt es innerhalb von Scrum noch weitere Artefakte. Diese werden im Folgenden genauer vorgestellt.

3.4.1 Product Backlog

Das Product Backlog beschreibt den Speicher an aktuell vorhandenen Anforderungen. Das Product Backlog selbst ist flexibel und dynamisch. Das bedeutet, dass jederzeit neue Anforderungen in das Product Backlog aufgenommen werden können. Außerdem können auch Anforderungen wieder aus dem Product Backlog entfernt werden, wenn z. B. der Product Owner feststellt, dass diese Anforderung nicht länger benötigt bzw. vom Kunden nicht mehr eingefordert wird. Innerhalb des Product Backlog ist es darüber hinaus möglich, bestehende Anforderungen zu ändern. Auch werden vom Kunden grob beschriebene Anforderungswünsche aufgenommen und im Zuge des Requirements Engineering Prozesses weiter verfeinert.

Innerhalb des Product Backlog liegen die Anforderungen in einer nach Priorität sortierten Reihenfolge (Racheva et al. 2008). Dabei liegen Anforderungen, welche soweit verstanden und spezifiziert sind, dass sie sich für eine Umsetzung eignen, vor den noch größeren, wenig verfeinerten Anforderungen. Ein weiteres Ordnungskriterium der Anforderungen in Form von User Stories innerhalb des Product Backlog ist neben dem reinen Kundenwunsch auch der einer User Story beigemessene Nutzwert und der zu erwartende Aufwand. Da es ein Ziel agiler Entwicklung im allgemeinen ist, möglichst viel Nutzwert am Ende einer Iteration für den Kunden bzw. die Kunden und Anwender bereitzustellen, werden Anforderungen häufig nach dem Verhältnis von Nutzen zu Aufwand priorisiert. Ein weiteres Element der Priorisierung sind ggf. vorhandene harte Deadlines für die Erledigung bestimmter Anforderungen. Diese müssen durch eine entsprechende Position innerhalb des Product Backlog reflektiert werden. Dieser Aufbau wird durch die nachfolgende Abbildung 3 verdeutlicht.

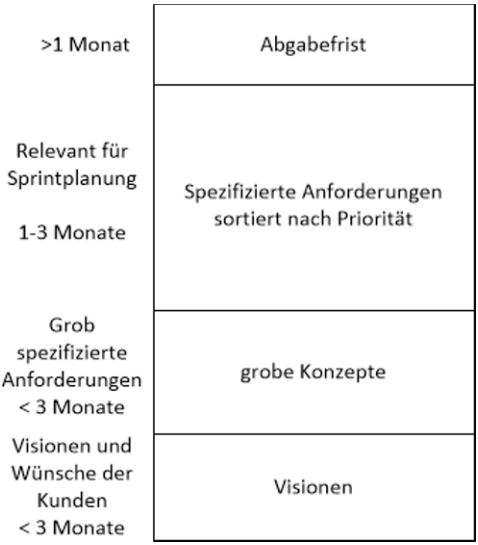


Abbildung 3: Schematischer Aufbau eines Product Backlogs (Quelle: eigene Darstellung)

Die Priorisierung, Verwaltung und die weitere Spezifikation und Verfeinerung vorhandener Anforderungen liegen in der Verantwortung des Product Owners.

Vlaanderen et al. stellen in ihrer Veröffentlichung (Vlaanderen et al. 2011) einen möglichen Prozess für die Spezifizierung und entsprechende Bewertung von Anforderungen und User Stories vor. Dieses Vorgehen ist dabei an das iterative Vorgehen von Scrum angelehnt (s. Abbildung 4).

Wie in Abbildung 4 dargestellt, beginnt ein „Software Product Management" (SPM) Sprint mit einer Vorbereitungsphase, in der die Produktrückstandselemente ausgewählt werden, die im nächsten Sprint weiter verfeinert und detailliert werden können oder müssen (1). Im nächsten Schritt werden entweder vorhandene Rückstandselemente für die weitere Spezifikation ausgewählt (2) oder neue Ideen und Visionen, welche bspw. aus der Kommunikation mit den Kunden erwachsen, neu aufgenommen (3). Dabei werden während eines SPM Sprints alle zuvor ausgewählten Elemente immer um einen Schritt weiter verfeinert. Dies bedeutet, dass es innerhalb dieses Vorgehensmodells je nach Grad der initialen Aufnahme mehrere SPM Sprints dauern kann, bis eine Anforderung weit genug spezifiziert ist, sodass sie für die Umsetzung eingeplant werden kann. Bei der eigentlichen Verfeinerung wird zwischen drei unterschiedlichen Stufen der Detaillierung unterschieden. Es wird zwischen groben „Themes" (4), mittelgroßen „Concepts" (5) und bereits in kleinere, verwaltbare Einheiten untergliederte Anforderungsdefintionen (6) unterschieden. Ziel ist es jeweils, eine Anforderung auf die nächste Stufe der Detaillierung zu bringen. Diese Verfeinerung wird iterativ für alle zuvor ausgewählte Elemente durchgeführt, bis der Sprint beendet ist. Zu Beginn der nächsten Phase (7) werden alle für die Umsetzung freigegebenen Anforderungen bzgl. des Nutzwerts und den zu erwartenden Umsetzungskosten bewertet.

Anschließend findet die Priorisierung der einzelnen Anforderung statt. Unterschiedliche Betrachtungsmöglichkeiten für eine Priorisierung von IT-Anforderungen werden in Kapitel 4.2 aufgezeigt. Auf Basis einer Liste mit priorisierten Anforderungen wird ein scrumtypisches Sprintplanungsmeeting durchgeführt und die Verantwortung für die im Planungsmeeting durch das Entwicklerteam bestätigten Anforderungen an das Entwicklerteam übertragen. Die bestätigten Anforderungen befinden sich somit nicht mehr im Produkt Backlog, sondern sind in das Sprint Backlog, welches in Kapitel 3.4.2 erläutert wird, übertragen. In Schritt (8) werden analog zum Vorgehen in Scrum ein Review und eine Retrospektive durchgeführt. Ziel hierbei ist es Abläufe zu identifizieren, welche gut und welche weniger gut liefen, um eine Art kontinuierlichen Verbesserungsprozeß zu initiieren (Vlaanderen et al. 2011).

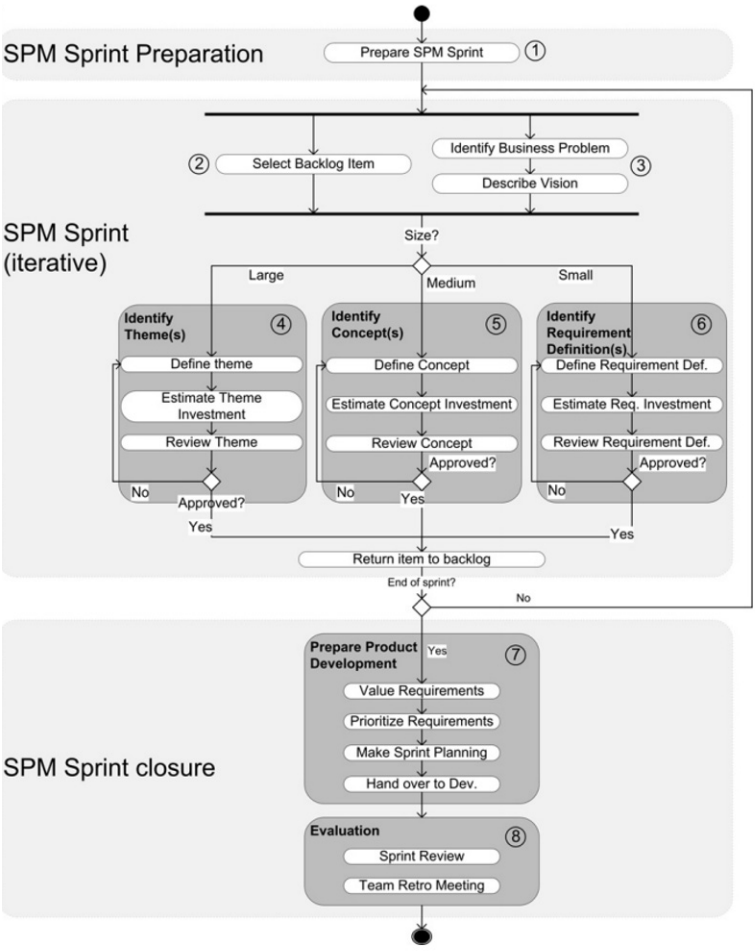


Abbildung 4: SCRUM software product management process, Quelle: (Vlaanderen et al. 2011)

3.4.2 Sprint Backlog

Das Sprint Backlog ist eine Menge von User Stories und den dazugehörigen Aufgaben, die erforderlich sind, um die für den Sprint fixierten Ziele und den damit verbundenen Anforderungen aus dem Product Backlog in ein auslieferbares Paket, welches einen Mehrwert für den Kunden liefert, zu überführen. Das Sprint Backlog entsteht als Ergebnis des Sprint Planning Meetings (s. Kapitel 3.3.2). Die Aufgaben, die für die Erledigung der User Stories notwendig

sind, werden im zweiten Teil des Sprint Planning Meetings erstellt und sollten eine Aufwandsschätzung in „normalen“ Arbeitsstunden erhalten. Diese Schätzwerte dienen als Grundlage für die Erstellung des Burn Down Charts, welcher in Kapitel 3.4.3 genauer beschrieben wird. Dieser dient als Grundlage für die Steuerung des Entwicklerteams und für die Nachvollziehbarkeit des Fortschritts.

Im Gegensatz zum Product Backlog, welches flexibel ist, ist das Sprint Backlog statisch. Das bedeutet, dass die Arbeitsumfänge, die zu Beginn eines Sprints innerhalb des Sprint Planning Meetings festegelegt wurden, für den Zeitraum eines Sprints fest fixiert sind und sich nicht mehr ändern. Dadurch soll sichergestellt werden, dass das Entwicklerteam sich auf diese definierten Aufgaben konzentrieren kann und bei der Abarbeitung nach Möglichkeit nicht unterbrochen oder gestört wird. Es ist die Aufgabe des ScrumMaster dafür Sorge zu tragen, dass diese Aufgabenfixierung eingehalten wird. Eine Ausnahme von dieser Regel stellen systemkritische Fehler bzw. Bugs dar. Sie machen teilweise eine Unterbrechung der aktuellen Arbeit notwendig, da die Behebung des Fehlers Vorrang vor den anderen Aufgaben erhält. Weniger kritische Bugs sollten im Product Backlog gesammelt werden und als „normale“ Aufgaben in den nächsten Sprint einfließen. Alternativ hierzu können vorhandene, nicht kritische Fehler auch über einen längeren Zeitraum gesammelt werden. Für die Behebung dieser Fehler wird dann ein separater Sprintdurchlauf vollzogen, in dem es ausschließlich um Fehlerbehebung und Refactoring geht.

Die User Stories mit ihren zugehörigen Aufgaben werden häufig in Form von Karteikarten oder Ähnlichem an eine sogenannte „Scrum Wand“ oder „Scrum Board“ geklebt. Diese Wand sollte sich an einer für alle Teammitglieder zugänglichen Stelle befinden, sodass jedes Teammitglied die Möglichkeit hat sich jederzeit über den aktuellen Zustand der Abarbeitung der Aufgaben innerhalb des Sprints zu informieren. Das Board ist dabei häufig in drei Bereiche untergliedert. Es gibt einen Bereich für noch zu erledigende Aufgaben („To Do“), einen Bereich für die Aufgaben in Bearbeitung („In Progress“) und einen Bereich für bereits innerhalb der Iteration abgeschlossene Aufgaben („Done“). Anstelle einer tatsächlichen Wand können auch Tools, wie z. B. der Team Foundation Server, kurz TFS, von Microsoft treten. Sie visualisieren das Scrum Board digital, bspw. in einem Web-Browser. Abbildung 5 zeigt ein fiktives Scrum Board innerhalb des Team Foundation Servers (TFS).

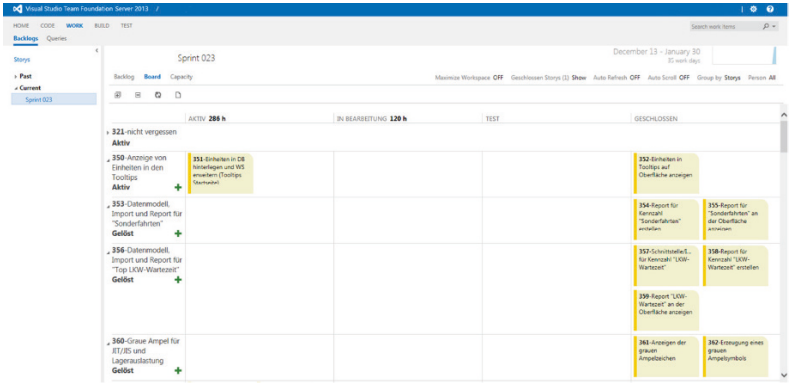


Abbildung 5: Darstellung eines Scrum Board innerhalb des TFS (eigene Darsellung)

Die Aufgaben innerhalb des Sprint Backlog werden keinem Entwickler direkt zugewiesen. Die Mitglieder des Teams organisieren eigenverantwortlich die Zuteilung der Aufgaben. Ein Entwickler, der eine Aufgabe abgeschlossen hat, verschafft sich nach Abschluss einen Überblick über die Aufgaben des aktuellen Sprint Backlog und wählt eine Aufgabe, von der er ausgeht, dass er diese sinnvoll bearbeiten kann und die noch nicht von einem anderen Entwickler bearbeitet wird, als seine nächste Aufgabe aus. Diese Aufgabe wird dann vom Status „To Do“ in den Status „In Progress“ gehängt. Damit übernimmt der Entwickler die Verantwortung für die Erledigung dieser Aufgabe. Nach Abschluss der Aufgabe hängt der Entwickler die Aufgabe in den Bereich für erledigte Aufgaben.

3.4.3 Burndown Chart

Der Burndown Chart ist eine für das Team essenzielle Auswertung über den Sprintfortschritt. Anhand des täglich erstellten Burndown Charts kann sich das Team messen und es können frühzeitig Probleme bei der Abarbeitung identifiziert werden, sodass das Team in der Lage ist sich auf Grundlage des Burndown Charts zu steuern.

Abbildung 6 zeigt einen fiktiven Burndown Chart kurz vor Beendigung eines Sprints. Der Burndown Chart zeigt auf der x-Achse das Datum, auf der y-Achse wird der geschätzte verbleibende Restaufwand der Tasks in Stunden repräsentiert. Die durchgehende schwarze Linie stellt den „optimalen“ Verlauf eines Sprints dar, mit einer konstanten kontinuierlichen Abnahme der verbleibenden Aufwände. An diesem Beispiel ist zu sehen, dass die Arbeitsaufwände zu Beginn etwas zu niedrig geschätzt wurden und es innerhalb der ersten Tage zu einer Anpassung der Aufwände kam. Anschließend hat das Team stetig den verbleibenden Aufwand verringert und sich zwischendurch an das Ziel angenähert. Wie zu sehen ist, sind nicht alle Aufgaben termingerecht fertig geworden, da zum Ende des Sprints noch ein Restaufwand vorhanden ist.

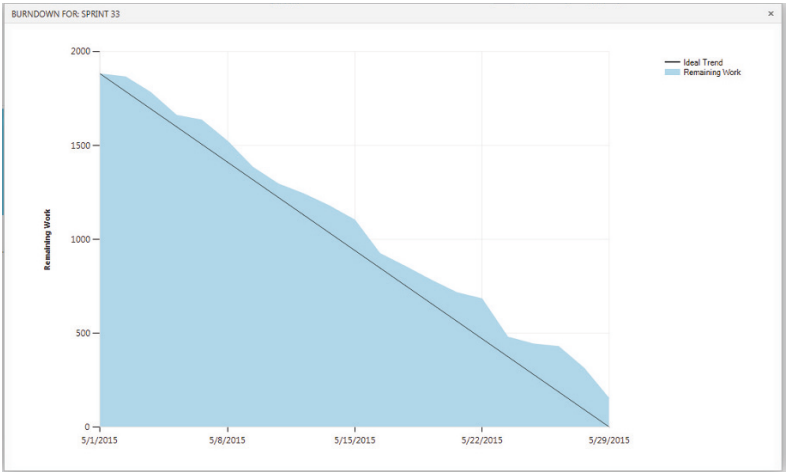


Abbildung 6: Beispielhafte Darstellung eines Burndown Charts (eigene Darstellung)

Das Team hat mittels des Burndown Charts die Möglichkeit seinen eigenen Fortschritt zu messen und wenn nötig frühzeitig Maßnahmen einzuleiten, um der Verfehlung des Sprintziels vorzubeugen. Stellt das Team fest, dass der verbleibende Aufwand über die restliche verfügbare Kapazität hinausgeht, ist zu prüfen, ob dieses innerhalb der verbleibenden Zeit realistischerweise eingearbeitet werden kann, z. B. durch die Nutzung vorhandener und eingeplanter Pufferzeiten. Ist dies nicht möglich, ist der Product Owner zu informieren und in Absprache mit ihm und ggf. weiteren Projektverantwortlichen zu bestimmen, wie weiter verfahren werden soll.

3.4.4 Blocks List

Die Blocks List ist eine separate, vom ScrumMaster gepflegte Liste, welche die aktuellen Blockaden, Hindernisse oder auch Impediments darstellt. Diese Impediments werden bspw. beim Daily Scrum Meeting benannt und dann zur Liste hinzugefügt.

Anschließend ist es die Aufgabe des ScrumMasters sich um die Lösung der vorhandenen Hindernisse bei Abarbeitung der Aufgaben zu kümmern, damit die Erreichung des Sprintziels nicht gefährdet wird. Typische Impediments sind bspw. fehlende Softwarelizenzen für einen Entwickler oder Verzögerungen bei Anschaffung benötigter Hardware.

Die Blocks List sollte wie das Scrum Board innerhalb des Projektkontextes zugänglich sein. Somit haben Entwickler die Möglichkeit sich über die aktuellen Impediments zu informieren und auf Grundlage der vorhandenen Impediments die weitere Abarbeitung der Aufgaben innerhalb des Sprints zu planen, sich als Team selbst zu organisieren und sich den entsprechenden Rahmenbedingungen anzupassen.

Dieses vorgestellte Rahmenwerk bildet neben den formulierten Forschungsfragen die Grundlage für die Analyse und Einordnung des aktuellen Stands der Forschung und dessen Übertragbarkeit auf die identifizierten Problemstellungen.

4 Aktueller Stand der Forschung

In diesem Kapitel wird der aktuelle Stand der Foschung, der in Zusammenhang mit den formulierten Forschungsfragen steht, vorgestellt und diskutiert. Zunächst wird, ausgehend von der Einführung in Scrum (s. Kapitel 3) evaluiert, welche Möglichkeiten bzgl. der Skalierbarkeit bestehen. In diesem Zusammenhang wird aufgezeigt, wie Scrum für große Softwareprojekte mit mehreren Entwicklerteams angewendet werden kann.

In einem weiteren Unterkapitel werden aktuelle Erkenntnisse zur Nutzwertbestimmung von Anforderungen und IT-Systemen wiedergegeben, da die Ermittlung des Nutzwertes auch einen Einfluss auf eine spätere Optimierung hat.

Darauf aufbauend werden ähnliche und allgemeinere Problemstellungen sowie entsprechende Lösungskonzepte vorgestellt, um für den späteren Verlauf der Arbeit die grundlegenden Probleme und ggf. noch anzupassende Lösungskonzepte einzuführen.

Außerdem werden Software-Produktlinien (SPL) eingeführt. Softwareprodukt-Linien haben dabei vom Grundsatz her ähnliche Ziele wie die, die in dieser Arbeit verfolgt werden: Vermeidung von Doppelentwicklungen, „standardisierte“ Software(-komponenten) für unterschiedliche Kunden, etc. Die Abgrenzung zwischen dieser Arbeit und Software-Produktlinien ist, dass sich SPL den Zielen aus Richtung der Systemarchitektur betrachtet. Diese Arbeit nähert sich den Zielen aus Sicht des Softwareentwicklungsprozesses und im Speziellen aus Sicht des Anforderungsmanagements und der Anforderungsanalyse.

4.1 Skalierbarkeit von Scrum für große Projekte

Die Scrum-Methode ist ursprünglich für einzelne Teams konzipiert worden und mittlerweile für diesen Einsatzzweck umfangreich beschrieben, erprobt und verbreitet. Für den Einsatz in großen Unternehmen und bei der Entwicklung eines IT-Systems durch größere bzw. mehrere Teams gibt es jedoch noch keinen allgemein verbreiteten Lösungsweg.

Für große Projekte, wie bspw. ein konzernweit eingesetztes Management-Informationssystem, ist es nicht immer möglich, alle Anforderungen innerhalb der gegebenen zeitlichen Restriktionen von einem Team vollumfänglich umzusetzen. Deswegen gibt es in der bestehenden Literatur unterschiedliche Ansätze, wie die organisatorischen Herausforderungen für eine größere Anzahl an Entwicklern innerhalb von Scrum behandelt werden können.

Ein Problem bei einem hohen Aufkommen an Anforderungen und an einem hohen Bedarf an Entwicklerressourcen ist, dass ein Scrum Team nicht beliebig groß werden sollte. Bei einer Teamgröße von mehr als zehn Personen steigt z. B. die Dauer des täglichen Scrum Meetings über die vorgesehene Zeit. Auch wird es für ein Entwicklerteam, welches eigentlich keine Hierarchien vorsieht, immer aufwendiger sich selbst zu organisieren. Somit leidet entweder die Kommunikation innerhalb des Teams oder der Zeitaufwand für Abstimmungen und organisatorische Belange steigt soweit an, dass die Produktivität des Teams negativ beeinflusst wird (Paasivaara et al. 2009).

Zur Lösung dieser Problematik bietet es sich an, dass mehrere Entwicklerteams weitestgehend selbstorganisiert, gemeinschaftlich an einem Projekt arbeiten. Hierbei ergeben sich unterschiedliche Problemstellungen bzgl. der Aufteilung der Entwickler zu unterschiedlichen Teams und für die Rolle des Product Owners.

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0_4

Für die Erstellung unterschiedlicher Teams wird in der Literatur häufig der Ansatz von „cross-functional Teams“ (auch „x-functional“) bevorzugt. Diese Teams zeichnen sich dadurch aus, dass sie in der Lage sind Anforderungen durch alle Ebenen der vorgesehenen Softwarearchitektur umzusetzen. Sie müssen also das nötige Know-how besitzen bspw. für Datenbank-Entwicklung, ggf. Kenntnisse über die sich im Einsatz befindlichen ETL-Techniken[3] haben, Front-end-Entwicklung betreiben können und benötigte Verbindungen zwischen Backend und Front-end herstellen können. Darüber hinaus müssen auch entsprechende Kompetenzen im Bereich der Qualitätssicherung vorhanden sein.

Eine andere Möglichkeit mehrere Scrum Teams zu erstellen ist, diese nach funktionalen Bereichen zu untergliedern. Es gibt somit ein Team für die Entwicklung der Datenbank sowie ein Team, welches sich nur mit der Entwicklung der Oberfläche und dessen Funktionalität befasst.

Die beiden unterschiedlichen Möglichkeiten für die Teamzusammensetzung werden in Abbildung 7 übersichtlich dargestellt und zusammengefasst.

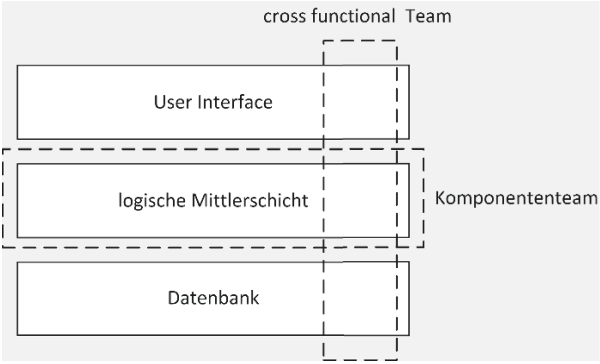


Abbildung 7: Cross-functional Team und Komponententeam (Quelle: eigene Darstellung in Anlehnung an (Pich- ler 2010))

Unterschiedliche Teams, welche gemeinsam an einem Projekt arbeiten, können z. B. über ein sogenanntes Scrum of Scrums oder auch Meta-Scrum integriert werden. Das Scrum of Scrums ist analog zum Daily Scrum zu sehen. In dieses Meeting entsendet jedes Team einen Vertreter, z. B. den jeweiligen ScrumMaster. Die jeweiligen Teamvertreter berichten dabei nach demselben Muster wie im Daily Scrum. Die unterschiedlichen Teams gleichen so ihren Fortschritt und ihre Erfahrungen auf einer höheren Ebene ab. Außerdem besprechen sie Impediments eines Teams, die nicht durch dieses Team allein gelöst werden können. Informationen und Absprachen aus dem Scrum of Scrums werden durch die jeweiligen Vertreter wieder bei Bedarf zurück in die Teams gespiegelt. Bei sehr großen Projekten mit mehr als 10 Entwicklerteams kann dieser Prozess auch mehrstufig erfolgen (Paasivaara et al., S. 2012).

[3] ETL steht für „Extract, Transform and Load“. Daten werden aus einer Quelle extrahiert, ggf. transformiert und anschließend geladen, zumeist in eine Datenbank.

Die Integration von mehreren Teams in einem Projekt wird auch von Sutherland et al. (2007) behandelt. Es wird ebenfalls darauf eingegangen wie mehrere Teams, die ggf. geographisch verteilt sind, mit eingebunden werden können.

Für diesen Zweck werden drei grundsätzliche Verfahren vorgestellt. Ein mögliches Vorgehen ist, dass die unterschiedlichen Teams isoliert arbeiten und „gesteuert" werden. Eine weitere Möglichkeit besteht nach Sutherland et al. (Sutherland et al. 2007) darin die Teams über ein bereits erläutertes Scrum of Scrums zu integrieren. Die dritte Alternative ist die Bildung von cross-funktionalen, cross-geographischen Teams. Diese 3 Möglichkeiten werden in Abbildung 8 veranschaulicht.

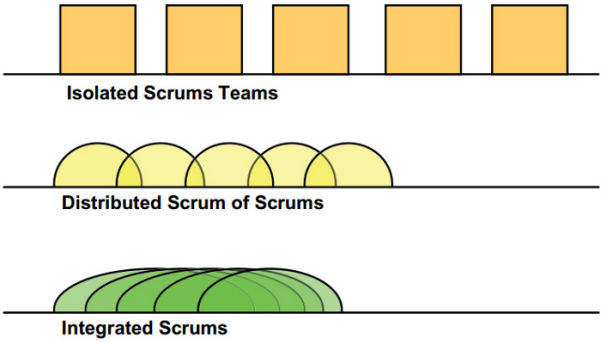


Abbildung 8: Strategien für verteilte Scrum-Teams (Nonaka und Takeuchi 2004; Sutherland et al. 2007)

In der Studie wird festgestellt, dass geografisch verteilte Entwicklerteams ähnlich produktiv sein können wie kleinere Entwicklerteams, die direkt an einem Ort zusammenarbeiten. Wichtige Erfolgsfaktoren sind ein regelmäßiger Informationsaustausch zwischen allen Teams und die Schulung der Teammitglieder im agilen Vorgehen.

Eine weitere Herausforderung bei der Skalierung von Scrum und dem Einsatz dieser Methode für große Entwicklungsprojekte ist die Rolle des Product Owners innerhalb des Prozesses und die dieser Rolle zugeschriebenen Aufgaben und Pflichten. Bei einem Entwicklerteam für ein System mit einem Kunden kann die Person, die die Rolle des Product Owners innehat, alle Aufgaben vollumfänglich erfüllen. In Projektsituationen mit mehreren Teams und mit mehreren Kunden, die sich unter Umständen nicht kennen und geografisch verteilt sind, ist es für einen Product Owner, der sowohl der Single Point of Contact für die Kunden, als auch erster Ansprechpartner für die Entwickler der unterschiedlichen Teams ist, schwer bzw. nahezu unmöglich allen Aufgaben und Pflichten mit der nötigen Sorgfalt nachzugehen.

In der Literatur gibt es unterschiedliche Lösungsansätze, um den Product Owner zu entlasten und die Erledigung der Arbeiten, welche an diese Rolle gebunden sind, sicherzustellen. Eine Möglichkeit ist ein sogenanntes Product-Owner-Komitee, wie es bspw. in Pichler (2010) vorgestellt wird.

Dies widerspricht jedoch der Auffassung von Sutherland und Schwaber, wonach es nur einen Product Owner geben sollte und die Entscheidungen über das Projekt und dessen Entwicklung in einer Hand liegen sollten (vgl. Schwaber und Sutherland 2013b).

Alternativ zum Product-Owner-Komitee schlägt Schwaber Product-Owner-Hierarchien vor (vgl. Schwaber 2004). Hierbei hat jedes Team einen eigenen dedizierten Product Owner, der für ein Team verantwortlich ist. Bei einer geringeren Anzahl an Teams wird einer der Team-Product-Owner zusätzlich auch noch Chief-Product-Owner. Als Chief-Product-Owner ist er derjenige, der für das Produkt verantwortlich ist und somit das Produkt Backlog pflegt. Als „normaler" Product Owner übernimmt er alle Aufgaben des Product Owner, die in Verbindung mit seinem zugeordneten Entwicklerteam stehen und ggf. das Management der zugehörigen Stakeholder. Bei größeren Projekten mit einer größeren Anzahl an Entwicklerteams können sich innerhalb des Product Owner Teams komplexe Hierarchien bilden, in denen bspw. einzelne Komponenten von mehreren Teams entwickelt werden. Diese werden dann ggf. über mehrere Product-Owner-Hierarchieebenen gesteuert, je nach Komplexität des Endproduktes.

Eine alternative Möglichkeit zur Skalierung der Rolle des Product Owner und zur Erhaltung der Agilität ist die Einführung der Rolle des Story Owner. Die Rolle des Story Owner wird zusätzlich zur Rolle des Product Owner eingeführt, um diesen zu entlasten. Hierbei ergibt sich ebenfalls eine Art Hierarchie. Der Product Owner bleibt dabei in seiner steuernden Position jedoch vollkommen uneingeschränkt. Er ist weiterhin alleinig in der Verantwortung das Product Backlog zu pflegen und die Priorisierung der Anforderungen vorzunehmen. Somit entfällt die bei einem Product-Owner-Komitee notwendige Abstimmung zwischen den Product Ownern.

Die neue Rolle des Story Owner ist nachgelagert zum Product Owner zu sehen. Es liegt in der Verantwortung des Story Owner die Detailabstimmung zu einzelnen User Stories durchzuführen und Rückfragen auf Story-Ebene zu beantworten. Somit soll der Product Owner in seiner Tätigkeit entlastet, ohne dabei „entmachtet" zu werden. Jedoch ist eine Entlastung des Product Owners für die Skalierung von Scrum erforderlich.

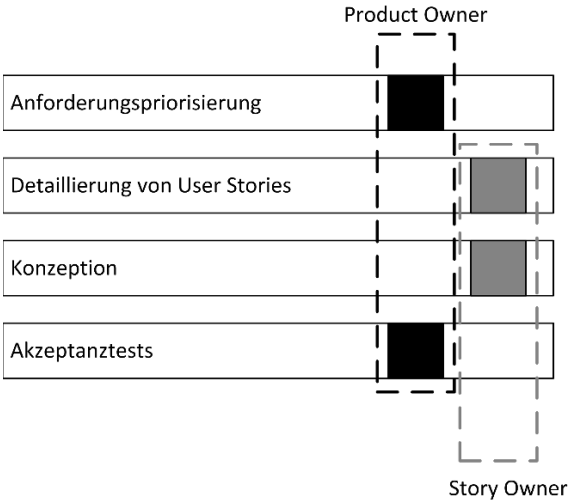


Abbildung 9: Aufgabenteilung zwischen Product Owner und Story Owner (eigene vereinfachte Darstellung nach Pichler 2010)

Wie hier vorgestellt, gibt es eine Vielzahl von Erweiterungen zur ursprünglichen Scrum-Methode, um diese in ihrem Kern zu erhalten und eine Skalierung über mehrere Teams gewährleisten zu können. Der Fokus dieser Arbeiten liegt auf der Behebung möglicher Probleme bei Scrum. Die Skalierung für Scrum selbst stellt voraussichtlich kein Problem dar, vielmehr die Auswahl der richtigen Methode unter den jeweils vorherrschenden Rahmenbedingungen.

Jedoch beschäftigt sich kein Beitrag mit den Chancen und Möglichkeiten für Kosteneinsparungen und die Nutzung von Synergiepotenzialen bei Projekten mit mehreren Kunden und die Umsetzung dieser Kundenanforderungen durch mehrere Teams. Ein Ziel dieser Arbeit ist die Erarbeitung einer Möglichkeit systematischer Erschließung und Nutzung vorhandener Synergiepotenziale zwischen Anforderungen unterschiedlicher Kunden und Stakeholdern.

4.2 Nutzwertbestimmung von IT-Anforderungen

Die Bestimmung des Nutzwertes von Anforderungen und die Frage, wie dieser bestimmt werden kann, wird in der ersten Forschungsfrage aus Kapitel 2.3 aufgegriffen. Es ist notwendig, bereits bestehende Lösungen bzgl. der Anwendbarkeit in der Domäne der Managementinformationssysteme zu überprüfen, da „Nutzwert“ in unterschiedlichen Domänen ggf. anders bestimmt wird. Eine mehrdimensionale Betrachtung des Nutzwertes ist ebenfalls erforderlich. Außerdem muss eine Möglichkeit bestehen unterschiedliche Priorisierungsergebnisse mehrerer Kunden konsolidieren zu können.

Die Themen „Priorisierung von Anforderungen“ und die „Bestimmung des Nutzwertes von Anforderungen“ sind sehr eng miteinander verwandte Forschungsrichtungen. Deswegen beschreibt Pettit Nutzwert („Business value“) als Hilfsmittel für die Kommunikation, um über Nutzwert, Priorität und Motivation zu sprechen (Pettit 2006).

Eine andere Studie zum Thema Wertbeitrag durch agile Projekte kommt zu dem Schluss, dass es zurzeit noch keine Definition von Nutzwert gibt, die sich durchgesetzt hat. Ein weiteres Ergebnis dieser Studie ist, dass es häufig zur Verknüpfung zwischen Nutzwert und monetärer Bewertung kommt und dass diese Verbindung problematisch sein kann. Außerdem wird festgestellt, dass die Definition von Nutzwert im Bereich der agilen Softwareentwicklung weiterhin volatil ist (Racheva et al. 2009). Diese Beobachtungen der Autoren könnten unter anderem auch den Schluss nahelegen, dass unter Umständen keine allgemeingültige Definition von Nutzwert existiert, sondern dass diese immer kontextabhängig ist.

Zu einem ähnlichen Ergebnis kommt auch die Studie von Kettunen (2013). Nach dieser Studie sind Softwaresysteme bzw. Softwareprodukte lediglich eine Aktivierungstechnologie. Dies bedeutet, dass nicht das direkte Ergebnis der Softwareentwicklung, also weder der Source-Code noch das System selbst, einen direkten Nutzwert aufweist. Ein finanzieller Nutzen entsteht erst durch die Verwendung und die Ergebnisse, die durch die Ausführung des Softwareproduktes im prozessualen Kontext erzielt werden. Der Nutzen kann also nicht unmittelbar durch die Software selbst erzeugt werden sondern erst durch die Ausführung im Kontext eines wertschöpfenden Prozesses (Kettunen 2013) (vgl. Abbildung 14).

Rein und Münch schlagen als Priorisierungskonzept für Funktionen einer Applikation auf mobilen Endgeräten ein Verfahren vor, welches auf den Implementierungskosten und dem zu erwartenden Erlös durch diese Implementierung beruht. Bei diesem Vorgehen wird der Aufwand

für die Implementierung eines Features zunächst durch die Entwickler abgeschätzt. Um Daten über den zu erwartenden Erlös zu erhalten, schlagen sie sogenannte „Mock-Purchases“ vor. Bei „Mock-Purchases“ wird dem Anwender vorgetäuscht, dass er das jeweilige Feature innerhalb der Applikation kaufen kann. Nach dem vorgetäuschten Kauf, wird der Anwender darüber informiert, dass es sich bei dem Kauf um keinen echten Kauf gehandelt hat und dass für ihn keine Kosten entstehen. Aus den Daten, die aus diesen fiktiven Käufen gesammelt werden, kann dann in einem weiteren Schritt berechnet werden, welches Feature zu welchen Preis verkauft werden kann und in Relation zu den Entwicklungskosten gesetzt werden, um daraus eine Priorisierung der Features innerhalb des Product Backlogs abzuleiten (Rein und Münch 2013). Diese Priorisierungsmethode basiert somit auf rein finanziellen Gesichtspunkten. Die Anwendung dieser Methode kann jedoch negative Auswirkungen auf die Kundenzufriedenheit haben.

Eine Alternative für die Priorisierung von Features wird von Denne und Cleland-Huang vorgeschlagen (Cleland-Huang und Denne 2005). Bei diesem Vorgehen werden „Softwarekomponenten“ nicht unter technischen Gesichtspunkten erstellt, sondern unter finanziellen Aspekten. In diesem Zusammenhang wird auch von „**M**inimum **M**arketable **F**eatures“ (MMF) gesprochen. Eine Zerteilung von einem komplexen Softwaresystem in mehrere MMFs ist möglich, da Softwareanwendungen im Gegensatz zu vielen anderen Produkten auch dann für den Anwender einen Nutzwert besitzen, wenn diese nicht vollständig umgesetzt sind. Außerdem ist es bei Softwareprodukten möglich, die Anwendung inkrementell weiterzuentwickeln und schrittweise neue Funktionen an den Kunden auszuliefern, selbst wenn eine erste Version dieser Software bereits im Einsatz ist. Dies ist unter anderem auch deswegen möglich, da auch „unfertige“ Softwareprodukte bereits einen Nutzwert aus Sicht des Kunden besitzen können. Als Beispiel hierfür wird eine komplexe Online-Bank-Applikation angeführt, die zu einem frühen Entwicklungsstand einem Kontoinhaber lediglich die Möglichkeit bereitstellt seinen aktuellen Kontostand einzusehen. Diese Eigenschaft, bzw. dieses MMF, hat bereits einen gewissen Nutzen aus Sicht des Nutzers. Ein solches MMF besitzt typischerweise einen Nutzen oder auch Marktwert in einer der folgenden Dimensionen:

- Wettbewerbsdifferenzierung
- Umsatzgenerierung
- Kosteneinsparung
- Markenstrahlkraft
- verbesserte Loyalität

Aus Sicht der Autoren ist Softwareentwicklung ein wertschöpfender Prozess mit dem Ziel die Nutzwerterzeugung gemäß den zuvor genannten Dimensionen zu optimieren. Bei der Betrachtung dieser Nutzwertdimensionen muss jedoch beachtet werden, dass diese aus der Domäne von Service-Anwendungen für den Endkunden entstanden sind, jedoch auch auf andere Anwendungsdomänen übertragbar sind (Denne und Huang 2004).

Dennoch sind diese Nutzenwerte nur teilweise auf eine interne IT-Abteilung, die vorwiegend Softwareprodukte für den internen Gebrauch entwickelt, zu übertragen, da aufgrund eines fehlenden „Marktes“ Attribute wie Markenstrahlkraft und Loyalität nicht oder nur unzureichend betrachtet werden können. Für die rein interne Betrachtung der Softwareentwicklung und –bereitstellung existiert kein echter Wettbewerb zwischen unterschiedlichen Produkten.

Die Nutzwertbestimmung von IT-Anforderungen dient letztlich der Priorisierung von Anforderungen und dem Aufzeigen von möglichen Anforderungen, die für die weitere Entwicklung

relevant sind. Die Entwicklung soll dadurch fokussiert und Wichtiges von Unwichtigem getrennt werden. Neben den bereits vorgestellten Methoden bzw. Vorgehensweisen gibt es auch Verfahren und Methoden, die allgemeinere Ansätze verfolgen und letzten Endes auch in einer priorisierten Reihenfolge von Anforderungen münden.

Eine Möglichkeit ist die Anwendung des „Analytischen Hierarchieprozesses“ (AHP). Dieses Vorgehen stammt ursprünglich aus der Management-Theorie zur Abwägung und Bewertung von Handlungsoptionen. Die genaue Funktionsweise dieses Vorgehens wird in Kapitel 6.4 noch genauer erläutert. Grundsätzlich besteht das Verfahren aus zwei Phasen. In der ersten Phase werden die Bewertungskriterien festgelegt und zueinander gewichtet. In der zweiten Phase werden dann die unterschiedlichen Optionen miteinander bzgl. der zuvor ermittelten Kriterien miteinander verglichen. (Saaty 1987; Saaty 1994)

Eine weitere Methode zur Erstellung einer priorisierten Liste mit Anforderungen ist der Einsatz der sogenannten MoSCoW Methode. MoSCoW ist ein Akronym, welches für die unterschiedlichen Kategorien innerhalb des Vorgehens steht. Die Kategorien sind dabei wie folgt definiert (Brennan 2009):

- M für Must; eine Anforderung, welche in der finalen Version erfüllt sein muss, damit das gesamte Ergebnis als Erfolg angesehen werden kann.
- S für Should; eine hochpriorisierte Anforderung, die, wenn möglich, umgesetzt werden sollte. Die Anforderung kann aber auch mit gewissen Abweichungen umgesetzt werden.
- C für Could; eine Anforderung, die potenziell erstrebenswert ist, aber nicht zwangsweise in das finale Produkt einfließen muss. Eine Anforderung innerhalb dieser Kategorie wird umgesetzt, wenn es die Ressourcen erlauben.
- W für Won’t; für Anforderungen in dieser Kategorie ist mit den Prozessbeteiligten abgestimmt, dass diese zunächst nicht umgesetzt werden.

Durch eine Zuordnung aller zur Verfügung stehenden Anforderungen zu einer dieser Kategorien ergibt sich auch eine Art Priorisierung. Bei dieser Methode kann jedoch bei Anforderungen innerhalb einer Kategorie nicht mehr unbedingt unterschieden werden, welche dieser Anforderungen wichtiger ist als eine beliebige andere Anforderung innerhalb der Kategorie.

Eine weitere Methode, um Anforderungen in eine Reihenfolge für die Umsetzung zu bringen, ist das Anwenden des Bubble Sortings. Dieses verläuft analog zum gleichnamigen Sortieralgorithmus (Gumm und Sommer 2006; Knuth 1998). Dabei wird die gesamte Liste der Anforderungen mehrmals durchlaufen. Es werden jeweils zwei benachbarte Anforderungen miteinander verglichen und bei Bedarf werden diese in Reihenfolge getauscht. Bei diesem Verfahren wird die gesamte Liste der Anforderungen mehrmals durchlaufen, bis alle Anforderungen richtig sortiert sind (Karlsson et al. 1998).

In Tabelle 3 werden die in diesem Abschnitt vorgestellten Verfahren bzgl. ihrer Eignung für den Einsatz im Umfeld von Managementinformationssystemen bewertet. Es zeigt sich, dass Mock-Purchases zwar darauf ausgelegt sind die Ergebnisse unterschiedlicher Kunden zusammenzuführen, jedoch wird der Nutzwert ausschließlich finanziell betrachtet. Außerdem werden sogenannte In-App-Käufe durchgeführt, bzw. vorgetäuscht. Solche Käufe für Erweiterungen, haben sich für industrielle Systeme noch nicht durchgesetzt. Deswegen scheint dieses Verfahren eher ungeeignet.

Die MoSCoW-Methode, wie auch die Bubble-Sortierung, sind nicht speziell für interne Softwaresysteme designed, jedoch sollten sie ohne weiteres auch in diesem Umfeld einsetzbar sein.

Die Nachteile dieser Methoden sind, dass der Nutzwert nur eindimensional erfasst werden kann und dass die Ergebnisse mehrerer Kunden nicht ohne weiteres aggregiert und zusammengefasst werden können. Aus diesen Gründen können auch diese Verfahren nicht ohne weiteres eingesetzt werden.

Das Verfahren, ein System über MMF „wachsen“ zu lassen, bietet den Vorteil, dass der Nutzwert mehrdimensional erfasst werden kann. Dieses Verfahren müsste bzgl. der zu bewertenden Nutzwertdimensionen angepasst werden, sodass diese nicht auf einen realen Markt zielen, sondern vielmehr auf die Bedürfnisse der internen Anforderer ausgerichtet sind. Es werden jedoch keine Aussagen über die Möglichkeit einer Konsolidierung unterschiedlicher Bewertungen getroffen.

Der „Analytische Hierarchieprozess“ erlaubt es grundsätzlich mehrere Nutzwertdimensionen zu betrachteten und auch Ergebnisse mehrerer Beteiligter zu aggregieren, um zu einem konsolidierten Ergebnis zu gelangen. Ein weiterer Vorteil dieses Verfahrens ist die Anwendung nicht nur in der Informatik sondern auch in der Wirtschaft zur Unterstützung anderer Entscheidungsprozesse (vgl. z.B. (Ramanathan und Ganesh 1994; Saaty 1999; Donegan et al. 1992)). Auch für AHP gilt, wie für MMF, dass zunächst die Bewertungsdimensionen neu bestimmt werden müssen, bevor dieses Verfahren eingesetzt werden kann.

Tabelle 3: Vergleich von Verfahren zur Priorisierung bzw. Nutzwertbestimmung

Verfahren\Kategorie	anwendbar im internen BI-Umfeld	Ergebnisse aggregierbar	mehrdimensionaler Nutzwert
Mock-Purchases	-	+	-
Minimum Marketable Features	○	-	+
Analytischer Hierarchieprozess	○	+	+
MoSCoW	○	-	-
Bubble Sort	○	-	-

Quelle: eigene Darstellung

Die hier kurz vorgestellten Methoden zeigen an dieser Stelle nur einen groben Überblick bzgl. der Variantenvielfalt an Vorgehensweisen bei der Priorisierung. Neben den hier beschriebenen Verfahren gibt es noch eine Reihe weiterer Verfahren und Abwandlungen. Jedes dieser Verfahren hat innerhalb unterschiedlicher Rahmenbedingungen unterschiedliche Qualitäten. Trotz einer großen Anzahl an unterschiedlichen Methoden scheint es, dass es nicht das absolut richtige Vorgehen gibt oder dieses noch nicht gefunden ist, sodass neue Ansätze wie auch Optimierungen auf diesem Gebiet weiter erforscht und benötigt werden (vgl. Wnuk et al. 2011).

Um einem Projektleiter die Auswahl der richtigen Methode für sein Projekt zu erleichtern, stellten Baja und Arora (2013) zunächst eine Einordnung der unterschiedlichen Vorgehensweisen zur Verfügung. Darauf aufbauend wird ein Algorithmus konstruiert, der auf Grundlage der durch die Projektbeteiligten angegebenen Wunscheigenschaften des Verfahrens das beste unter den in Betracht kommenden Verfahren auswählt.

Für die Forschungsfrage bezüglich des Nutzwertes bedeutet dieses, dass vermutlich kein vollkommen neues Verfahren entwickelt werden muss. Vielmehr kann ein Verfahren bestimmt werden, welches bspw. auf AHP aufsetzt. Dieses Verfahren muss dann entsprechende Definitionen für den Nutzwert und ggf. notwendige Erweiterungen bereitstellen, sodass das Verfahren im Kontext der Entwicklung von BI-Systemen für den internen Gebrauch angewendet werden kann.

4.3 Verwandte Problemstellungen

Im Folgenden werden auf Grundlage einer ersten groben Einschätzung des sich ergebenden Sprintproblems bereits bekannte Probleme vorgestellt, die zu dem Sprintproblem Ähnlichkeiten aufweisen. Dies steht im Zusammenhang mit der dritten Forschungsfrage, welche in Kapitel 2.3 abgeleitet wurde. Eine Vereinfachung der Iterationsplanung, die Durchführung der Planung mit weniger Aufwand für den Product Owner, ist vermutlich nur durch ein (semi-)automatisiertes Verfahren zu erreichen. Je nach Definition des Optimierungsproblems kann ggf. eine Reduktion auf ein bereits bekanntes Problem durchgeführt werden. Durch eine Reduktion würde eine Probleminstanz des Sprintproblems in ein anderes bereits bekanntes Problem übersetzt werden. Dieses Problem könnte dann mit einem bereits erprobten Verfahren gelöst und anschließend wieder rücktransformiert werden.

Zunächst wird das Iteration Scheduling Problem kurz erörtert und die Abgrenzung zur Zielstellung dieser Arbeit herausgearbeitet. Außerdem werden auch allgemeinere Probleme behandelt, wie das Rucksackproblem und das Bin Packing Problem. Beim Rucksackproblem wird eine Nutzwertoptimierung unter gegebenen Rahmenbedingungen angestrebt, beim Bin Packing Problem wird versucht, die Behälteranzahl für das Verpacken von Gegenständen zu minimieren. Abstrakt betrachtet, können Behälter oder Rucksäcke als Iterationen angesehen werden und Gegenstände oder Waren als IT-Anforderungen, die durch eine Größe (Aufwand) und einen Nutzwert beschrieben sind.

4.3.1 Iteration Scheduling

In Szőke (2009) wird ein Informationsmodell vorgestellt, welches mittels eines heuristischen Scheduling-Algorithmus Unterstützung bei der Iterationsplanung anbietet. Das in dieser Arbeit vorgestellte Informationsmodell wird in Abbildung 10 dargestellt.

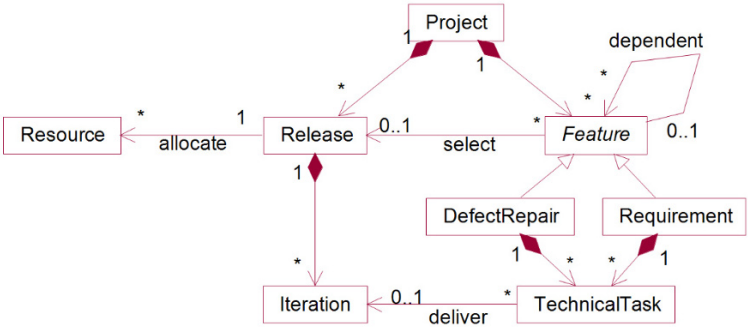


Abbildung 10: Informationsmodell für agile Planung (Quelle: (Szőke 2009))

Nach diesem Modell umfasst ein Projekt mehrere Features, welche untereinander Abhängigkeiten aufweisen können. Für ein Release wird eine Menge dieser Features ausgewählt. Dieses führt zu einer Allokation bestimmter Ressourcen (bspw. Entwicklerressourcen). Ein Release besteht wiederum aus einer oder mehreren Iterationen. In diesen Iterationen werden letzten Endes technische Tasks abgearbeitet, die entweder zur Umsetzung einer Anforderung oder zur Beseitigung eines vorhandenen Fehlers führen. In der Arbeit von Szöke wird dieses Modell mit dem „resource-constrained project scheduling optimization problem“ (RCPSP) (Schwindt 2005) in Verbindung gebracht. Das Ziel dieser Arbeit ist es auf Grundlage von beschränkten Ressourcen eine Verteilung der Aufgaben zu finden, sodass die Abarbeitungszeit minimiert wird.

Mittels eines heuristischen Ansatzes, welcher auf Grundlage vergangener Projekte evaluiert wurde, konnte festgestellt werden, dass die Verteilung der Arbeit auf die zur Verfügung stehenden Ressourcen verbessert werden konnte. Außerdem konnte durch das algorithmische Vorgehen eine bessere Entscheidungsgrundlage für die tatsächliche Sprintplanung erzeugt werden (vgl. Szőke 2009).

Bei diesem vorgeschlagenen Vorgehen wird versucht die Umsetzungszeit zu minimieren und die vorhandenen Ressourcen gleichmäßig auszulasten. Der Unterschied zu dem in dieser Arbeit angestrebten Modell ist zum einen, dass dieses Modell nicht vollständig ist, da z. B. Beschränkungen bei Zuordnung von Tasks auf Ressourcen nicht betrachtet werden. Außerdem ist fraglich, ob in einem agilen Kontext mit festen Zeiteinheiten für Iteration (z. B. 4 Wochen) eine Minimierung der Bearbeitungszeit für eine festgelegte Menge an Features das geeignete Optimierungskriterium ist. Außerdem muss in einem agilen Kontext davon ausgegangen werden, dass sich das Product Backlog zwischen zwei Iterationen ändert. Dabei können neue Anforderungen in das Product Backlog mit einfließen oder auch Anforderungen wieder entfernt werden. Diese Veränderungen können unter Umständen Auswirkungen auf die gesamte Planung haben, sodass bei sich kontinuierlich ändernden Anforderungen nicht zwangsweise in jeder Iteration eine möglichst große Systemverbesserung bezogen auf den Nutzwert des Systems erreicht wird.

4.3.2 Rucksackproblem

Das Rucksackproblem (KP) ist eines der 21 NP-vollständigen Probleme von Karp (1972). Es handelt sich hierbei um ein Optimierungsproblem, bei dem aus einer gegebenen Menge von Gegenständen, die durch ein Gewicht und einen Nutzwert beschrieben sind, eine Teilmenge ausgewählt werden soll, die zum einen eine bestimmte Gewichtsgrenze nicht überschreitet und zum anderen den Nutzwert maximiert.

Anschaulich kann dieses bspw. an einem Wanderer verdeutlicht werden. Dieser Wanderer möchte seinen Rucksack packen. Aus Erfahrung weiß der Wanderer, dass sein Rucksack eine bestimmte Gewichtsgrenze nicht überschreiten sollte, damit er diesen bei seinen Wanderungen gut tragen kann. Die Optimierung besteht in der Auswahl der Gegenstände, die der Wanderer auf seine Reise mitnimmt. Die Gegenstände werden dabei so ausgewählt, dass sie für den Wanderer den bestmöglichen Nutzwert erzielen. Die Formalisierung des Problems ist in Kapitel 12.2 dargestellt.

Die Analogien zur Planung einer Iteration sind, dass ein Team, wie ein Rucksack, über eine begrenzte Kapazität verfügt. Außerdem haben User Stories einen Aufwand oder abstrakt ein Gewicht und einen Nutzwert. Diese Eigenschaften können direkt übertragen werden. Deswegen können Lösungsstrategien für dieses Problem ggf. auf das Sprintproblem übertragen werden.

Für dieses Problem bestehen drei grundsätzliche Lösungsstrategien. Eine Strategie ist die Verwendung von heuristischen Verfahren, die andere ist der Einsatz von naturanalogen Metaheuristiken. Die letzte Alternative ist die Bestimmung einer tatsächlich optimalen Lösung durch speziell auf dieses Problem zugeschnittene Algorithmen. Im Folgenden werden alle drei Varianten an jeweils einem Beispiel illustriert.

4.3.2.1 Heuristisches Verfahren

Es gibt eine große Menge an unterschiedlichen heuristischen Verfahren für die Lösung des Rucksackproblems und anderer ähnlicher Probleme (Dyckhoff 1990). Viele Ansätze verwenden dabei das sogenannte „bang-for-bug“ Verhältnis (Pirkul 1987). Bei diesem Vorgehen wird z. B. aus dem Verhältnis von Nutzwert zu Gewicht ein Indikator abgeleitet, nach dem die Gegenstände zunächst sortiert und anschließend verarbeitet werden. Balas und Zemels (1980) Untersuchungen haben gezeigt, dass für kleine Problemgrößen eine sehr hohe Ergebnisqualität erreicht werden kann (Chu und Beasley 1998).

Alternative heuristische Verfahren, starten nicht mit einem leeren Rucksack, sondern mit einem „übervollen“ Rucksack, in dem zunächst alle Gegenstände liegen. Nacheinander wird jeweils ein Gegenstand, gemäß den Regeln der jeweiligen Heuristik, aus der Lösung entfernt, bis alle Rahmenbedingungen erfüllt sind (Zanakis 1977).

Für die Lösung von Probleminstanzen wird auch die Definition eines „Problemkerns“ verwendet. Durch Untersuchungen konnte bspw. festgestellt werden, dass eine Lösung, welche auf Grundlage eines Kosten-Nutzen-Verhältnisses erzeugt wird, meistens nur Abweichungen bei wenigen „Gegenständen“ aufweist. Außerdem konnte festgestellt werden, dass Bestandteile der optimalen Lösung zumeist sehr dicht in dem Bereich sind, in dem sie auch durch das heuristische Verfahren gewählt würden. Ein Kernproblem umfasst dabei die Variablen (Gegenstände), deren Kosten-Nutzen-Verhältnis in einem Bereich liegt, dass sie Bestandteil einer Lösung von KP sind aber nicht der von LKP und vice versa. LKP ist in diesem Zusammenhang die lineare

Relaxation der entsprechenden Instanz von KP. (Pirkul 1987; Balas und Zemel 1980). Die Bedingung, dass ein Gegenstand entweder in der Lösung enthalten ist oder nicht, d.h. Teil der Lösung ($x_j \in \{0, 1\}$, s. Kapitel 12.2) ist, wird aufgegeben und durch eine entsprechende relaxierte Nebenbedingung ersetzt, z. B. in der Form $0 \leq x_j \leq 1$. Die reelle Lösung, die für das abgeleitete, relaxierte Problem erzeugt wird, erlaubt Rückschlüsse auf die tatsächliche Lösung. Diese Methode ist nur dann vorteilhaft anwendbar, wenn die Lösung der Relaxation und die anschließende Überführung der Lösung mit weniger Aufwand geschehen können, als die Lösung des eigentlichen Problems. (Dudziński und Walukiewicz 1987)

4.3.2.2 Naturanaloges Verfahren

Für die Lösung des mehrdimensionalen Rucksackproblems wurde bspw. ein vom Simulated Annealing inspirierter Algorithmus verwendet, um diese Formulierung des Rucksackproblems zu lösen. Durch die Verwendung dieses Verfahrens, welches in Kapitel 8.3.3.2 genauer erläutert wird, werden näherungsweise gute Ergebnisse erzeugt. Das Verfahren konvergiert außerdem schnell gegen die tatsächliche optimale Lösung (Drexl 1988).

Khuri et al. (1994) verwenden einen einfachen genetischen Algorithmus, welcher in seiner Grundstruktur in Kapitel 8.3.3.3 dargestellt wird, für die Lösung des mehrdimensionalen Rucksackproblems. Durch Evolution und eine simple Bewertungsfunktion der Lösung unter der Verwendung von Strafen für die Überschreitung von Rahmenbedingungen wird die „Fitnes“ der Lösungskandidaten bestimmt. Durch diesen einfachen Ansatz konnte ohne zusätzliche Maßnahmen zur Verbesserung die Ergebnisqualität gesteigert werden.

Chu und Beasley (1998) verwenden vom Grundsatz her denselben Aufbau, jedoch benutzen sie aufwendigere Operatoren und „Reparaturmechanismen“, um die erzeugten Lösungen zu verbessern. Der Reparaturmechanismus bildet dabei auch auf eine gewisse Art und Weise eine Heuristik nach, da zunächst, sollte der Rucksack überfüllt sein, Gegenstände mit dem geringsten Kosten-Nutzen-Verhältnis gestrichen werden, bis es sich um eine valide Lösung handelt. Im nächsten Schritt wird die Lösung angereichert. Ggf. vorhandene Restkapazitäten werden aufgefüllt. Dieses geschieht ebenfalls auf der Basis des Verhältnisses von Nutzen zu Kosten.

Um die Ergebnisse naturanaloger Verfahren zu verbessern, werden diese auf unterschiedliche Arten mit anderen heuristischen Verfahren gekoppelt. So können z. B. andere heuristische Verfahren verwendet werden, um eine optimierte Startpopulation bei einem genetischen Algorithmus zu erzeugen oder um Einfluss auf die Verarbeitung zu nehmen (Djannaty und Doostdar 2008; Leung et al. 2012; Cotta und Troya 1998).

Leung et al. (2012) verwenden eine hybride Mischung aus Simulated Annealing und einem heuristischen Verfahren zur Lösung des zweidimensionalen Rucksackproblems. Als Rahmen wird Simulated Annealing angewendet und innerhalb dieses Rahmens werden heuristische Lösungen erzeugt. Innerhalb des äußeren Rahmens werden dabei Unterschiede bei der Sortierung der zu verpackenden Objekte erzeugt. Aufgrund der probabilistischen Akzeptanz einer Lösung kann mit einer hohen Wahrscheinlichkeit davon ausgegangen werden, dass lokale Optima auch wieder verlassen werden können.

4.3.2.3 Optimierter Algorithmus

Der Nemhauser/Ullmann-Algorithmus ist ein Beispiel für einen speziell für das Rucksackproblem entwickelten Algorithmus. Dieser Algorithmus macht sich dabei bestimmte Eigenschaften

des Rucksackproblems zunutze, um den Suchraum für eine Lösung schrittweise einzugrenzen und somit nicht mehr alle möglichen Lösungen betrachten zu müssen. Der Nemhauser/Ullman-Algorithmus berechnet dabei alle pareto-optimalen Lösungen für das Rucksackproblem. Pareto-optimale Lösungen sind in dem Zusammenhang alle theoretischen Beladungen des Rucksacks R, die nicht durch einen anderen Rucksack R‘ dominiert werden. Dabei dominiert ein Rucksack R einen Rucksack R‘ genau dann, wenn die Summe der Nutzwerte der Gegenstände innerhalb des Rucksacks R größer ist als die von R‘ und R dabei gleichzeitig weniger Gewicht aufweist als R‘. Das Gewicht eines Rucksacks ergibt sich dabei als Summe der einzelnen Gewichte der Gegenstände, die in den Rucksack gelegt sind.

Der Algorithmus startet dabei mit einem leeren Rucksack, erhält als Eingabe die entsprechende Kapazitätsgrenze des Rucksacks und die zur Verfügung stehenden Gegenstände, die durch Gewicht und Nutzwert beschrieben sind.

Anschließend werden alle pareto-optimalen Lösungen erzeugt, bei dem schrittweise nur die ersten i-Gegenstände betrachtet werden. Somit ist der Rucksack initial leer. Im nächsten Schritt wird eine Kopie dieses Rucksacks erzeugt und der erste Gegenstand der Kopie hinzugefügt. Anschließend werden die so neu erzeugten Rucksäcke zusammen mit dem aus dem vorherigen Schritt in einer Liste zusammengefasst, nach aufsteigendem Gewicht sortiert und dominierte Rucksacke werden entfernt. Rucksäcke, die die Gewichtsgrenze überschreiten, werden ebenfalls entfernt. Diese Liste bildet den Ausgangspunkt für die nächste Iteration, bei der erneut eine Kopie der resultierenden Liste erstellt wird und allen Rucksäcken der Kopie der nächste Gegenstand hinzugefügt wird. Abschließend werden die Ergebnisse erneut konsolidiert und alle dominierten Lösungen entfernt. Dieser Vorgang wiederholt sich so lange, bis alle Gegenstände betrachtet sind.

Die Laufzeit des Algorithmus beträgt O(nq) mit n als Anzahl der Gegenstände und q als Anzahl der pareto-optimalen Lösungen. Hierbei ist zu beachten, dass im schlechtesten Fall 2^n pareto-optimale Lösungen existieren. Jedoch haben Beier und Vöcking gezeigt, dass die Anzahl der pareto-optimalen Lösungen polynomiell in n ist, bei unterschiedlichen zufallsverteilten Eingaben (Beier und Vöcking 2003).

4.3.3 Bin Packing

Bin Packing oder auch Behälterproblem ist ein Problem, bei dem es anschaulich darum geht, eine gegebene Menge an Gegenständen auf eine bestimmte Anzahl an Behältern zu verteilen, sodass alle Gegenstände zugeordnet sind und kein Behälter überfüllt ist. Die Behälter haben dabei eine bestimmte Kapazität und die Gegenstände sind durch Abmessungen beschrieben.

Die Ähnlichkeiten zur Iterationsplanung bestehen in der bestmöglichen Auslastung mehrerer Entwicklerteams mit einer bestimmten Kapazität in der folgenden Iteration. Die Kapazität der Behälter entspricht der Kapazität der Entwicklerteams, die zu verteilenden Waren entsprechen den Anforderungen die auf die Teams bzw. die Behälter zu verteilen sind. Somit könnten, je nach Formulierung des Sprintproblems, auch Lösungsstrategien, die für dieses Problem anwendbar sind, übertragbar sein.

Das Behälterproblem ist in der Entscheidungsvariante NP-vollständig. In der Optimierungsvariante, also der Frage nach der minimalen Anzahl an Behältern, ist das Problem NP-schwer.

Formal ergibt sich somit die folgende Problemstellung (Martello und Toth 1990):

Gegeben:

- J als Menge der Gegenstände
- w_j als Gewicht für Gegenstand j aus J
- c als Kapazität für jeden Behälter

Gesucht:

- minimiere $z = \sum_{i=1}^{n} y_i$, Anzahl der verwendeten Behälter soll minimiert werden

Bedingungen:

- $\sum_{j=1}^{n} w_j x_{ij} \leq c y_i, i \in \{1, \dots, n\}$, Gewichtsgrenze wird eingehalten
- $\sum_{i=1}^{n} x_{ij} = 1$, jeder Gegenstand wird nur genau einem Behälter zugeordnet
- $y_i = \begin{cases} 1 \; wenn \; Behälter \; i \; benutzt \; wird \\ 0 \; sonst \end{cases}$
- $x_{ij} = \begin{cases} 1 \; wenn \; Gegenstand \; j \; in \; Behälter \; i \; gelegt \; wird \\ 0 \; sonst \end{cases}$

Im Folgenden werden für das Problem heuristische und naturanloge Lösungsverfahren vorgestellt.

4.3.3.1 Heuristisches Verfahren

Für das Bin Packing Problem gibt es eine Vielzahl von unterschiedlichen Heuristiken (vgl. z. B. (Baker 1985; Rao und Iyengar 1994; Johnson 1974)). Eine bekannte und gut studierte Heuristik ist die „**F**irst **F**it **D**ecreasing“ (FFD) Heuristik.

Bei diesem heuristischen Verfahren wird die Liste aus zu verpackenden Gegenständen zunächst absteigend nach Größe sortiert. Anschließend werden die Objekte der Reihenfolge nach in Behälter verpackt. Ein Gegenstand wird dabei immer in den ersten Behälter gelegt, der noch über genügend Kapazität verfügt. Sollte kein bereits verwendeter Behälter mehr über ausreichend Kapazität verfügen wird ein neuer Behälter geöffnet und dieses Objekt wird in den neuen Behälter gelegt (vgl. Kunde und Steppat 1985).

Für die FFD Heurisitk wurde bereits gezeigt, dass das Ergebnis für eine beliebige Liste (L) an Gegenständen maximal $\frac{11}{9} OPT(L) + 4$ beträgt (Johnson 1973) bzw. $FFD(L) \leq \frac{11}{9} OPT(L) + 3$ (Baker 1985).

Ein alternativer heuristischer Algorithmus ist „**F**irst **F**it **I**ncreasing“(FFI), welcher analog zu FFD verläuft. Der Unterschied ist, wie auch aus dem Namen hervorgeht, dass die Gegenstände nicht absteigend, sondern aufsteigend nach ihrer Größe sortiert werden.

Andere Heuristiken für dieses Problem sind „**F**irst **F**it“ (FF) Heuristiken, bei denen der Gegenstand jeweils in den ersten freien Behälter gelegt wird, der über ausreichend Kapazität verfügt oder sogenannte „**B**est **F**it“ (BF) Heuristiken, bei denen der Gegenstand in den Behälter gepackt wird, der ausreichend Platz besitzt und den entsprechenden Behälter soweit wie möglich ausfüllt, ohne jedoch die Grenze zu überschreiten (Johnson et al. 1974).

Alle hier kurz vorgestellten Heuristiken öffnen erst dann einen neuen Behälter (Bin), wenn es keinen Behälter gibt, der den entsprechenden Gegenstand aufnehmen kann (Johnson 1974).

4.3.3.2 Naturanaloges Verfahren

In der Veröffentlichung von Rao und Iyengar (1994) wird gezeigt, wie das Simulated Annealing Verfahren auf das Bin Packing Problem angewendet werden kann. Der verwendete Algorithmus orientiert sich dabei am klassischen Vorgehen und ist im Nachfolgenden grob skizziert (vgl. Rao und Iyengar 1994):

Durch die richtige Konfiguration der Start- und Abbruchtemperatur, also auch durch die Wahl des passenden Abkühlschemas konnten Rao und Iyengar zeigen, dass mittels dieses Verfahrens für einen Spezialfall des Bin Packing Problems die Ergebnisse einfacher heuristischer Verfahren verbessert werden können. Für den Vergleich wurden vier unterschiedliche Heuristiken ausgewählt.

Nach Angaben der Autoren zeigte sich, dass Simulated Annealing konstantere Ergebnisse liefert und gegen Störungen unempfindlicher ist als die heuristischen Verfahren. Außerdem habe sich gezeigt, dass die Ergebnisse in den meisten Fällen die der heuristischen Verfahren übertreffen.

```
begin
    erzeuge zufällige Lösung C
    T ← T_start
    while T > T_0 do
        repeat
            erzeuge neues C'
            ΔC = E(C') − E(C)
            if ΔC < 0 or
                C ←C'
        until Verweildauer auf T ist erreicht
        T ← F(T)
    end do
    C ist beste Lösung
end
```

Abbildung 11: Simulated Annealing (Pseudocode)

Genetische Verfahren (Falkenauer und Delchambre 1992) und hybride genetische Verfahren (Reeves 1996) sind für das Bin Packing Problem ebenfalls evaluiert wurden. Hierbei zeigt die Studie von Reeves, dass ein „traditioneller" genetischer Algorithmus auf das Bin Packing Problem anwendbar ist, jedoch weitere Verbesserungspotenziale vorhanden sind. Durch die Verwendung von online Heuristiken in Kombination mit einem genetischen Algorithmus konnten im Rahmen der Studie sowohl die Ergebnisse einer einfachen First Fit Heuristik, als auch die Resultate eines reinen genetischen Algorithmus verbessert werden.

In der Studie von Stawowy (2008) wird eine andere „hybride" Vorgehensweise vorgeschlagen. Abweichend von einer reinen Kombination von heuristischen Verfahren und genetischen Algorithmen, wird in dieser Studie vorgeschlagen, zunächst eine einfache Heuristik, wie z. B. die

„First Fit Heuristik“ anzuwenden, um eine initiale Lösung zu erhalten. In der weiteren Verarbeitung wird ein „reduzierter“ genetischer Algorithmus verwendet, welcher ausschließlich einen Mutationsoperator benutzt. Die „Nachfahren“ werden dabei also ausschließlich durch Mutation erzeugt. Der generelle Ablauf ist in der folgenden Abbildung dargestellt.

```
Begin
    generate and evaluate start solution
    best solution  := start solution
    counter := 0

    Repeat
        For i:=1 to X DO
        Begin
            copy parent to create child i
            mutate child i
            evaluate child i
        End

        Choose best child based in fitness function as new parent

        If FF(new parent) < FF(best solution) Then
            counter := counter+1
        Else
            counter := 0
            best solution := new parent
        End

        If counter ≥ max Counter Then
            counter := 0
            choose the worst child based on evaluation as new parent
        End

    Until max iterations

    Output best solution

End
```

Abbildung 12: Pseudo Code: reduzierter genetischer Algorithmus (Quelle: Stawowy 2008)

Innerhalb des Algorithmus wird, jeweils ausgehend von der besten Lösung einer Iteration, durch Mutation eine neue Generation erzeugt. Sollte der Algorithmus in eine Art „Sackgasse“ geraten und über einen gewissen Zeitraum keine bessere Lösung entstehen, so wird die Wurzel für die nächste Iteration durch den „schlechtesten“ Kandidaten ersetzt. Durch diese Neuausrichtung soll sichergestellt werden, dass die Suche nach der besten Lösung an einer anderen Stelle neu beginnt (vgl. Stawowy 2008).

Aufgrund der Vielfalt an unterschiedlichen Lösungsverfahren für das Bin Packing Problem wie auch für das Rucksackproblem erscheint es notwendig, für ein neues Problem innerhalb dieser Komplexitätsklasse unterschiedliche Verfahren zu evaluieren und das entsprechend beste Verfahren auszuwählen.

4.3.4 Zusammenfassung

Die drei hier vorgestellten Probleme zeigen alle Analogien zu einer Iterationsplanung in der agilen Softwareentwicklung.

Das Iteration Scheduling, weist dabei die offensichtlichsten Ähnlichkeiten bzgl. des Anwendungsgebietes auf. Jedoch wird bei diesem Problem die Abarbeitungszeit aller Anforderungen minimiert. Es wird also nicht unbedingt eine Nutzwertoptimierung mit jeder Iteration angestrebt, wie es eigentlich Ziel der agilen Entwicklung ist.

Das Rucksackproblem besitzt analogien zur Planung einer Iteration mit einem Team. Hierbei wird versucht den Nutzwert zu optimieren ohne eine bestimmte Gewichts- bzw. Aufwandsgrenze zu überschreiten. Die Betrachtung mehrer Teams kann dazu führen, dass bestehende Lösungsmechanismen nicht übertragbar sind.

Ähnlich verhält es sich mit den Gemeinsamkeiten zwischen Iterationsplanung und Bin Packing. Im Gegensatz zum Rucksackproblem können hier direkt mehrere Teams mit einer Probleminstanz assoziiert werden. Allerdings ist hier das Ziel die Minimierung der verwendeten Behälter. Dieses steht eher im Zusammenhang mit der Reduktion der Umsetzungszeit. Nutzwerte und deren Optimierung werden in diesem Problem nicht betrachtet.

Mit dieser Betrachtung wurde ein erster Überblick über Optimierungsprobleme geschaffen und aufgezeigt, dass unterscheidliche Verfahren zur Lösung der Probleme existieren. Zurzeit kann jedoch noch nicht beurteilt werden, ob und inwieweit bestehende Lösungsverfahren auf das noch zu definierende Sprintproblem angewendet werden können.

4.4 Software-Produktlinien

Die zweite Forschungsfrage beschäftigt sich mit der Dokumention und späteren Nutzung von Ähnlichkeiten zwischen Anforderungen. Deswegen beschäftigt sich dieses Kapitel mit Softwareproduktlinien. Softwareproduktlinien zeichnen sich dadurch aus, dass Anforderungen mit einem oder mehreren bestehenden Features umgesetzt werden können. Diese Features werden modular zu einem auf die Anforderungen des Kunden angepassten Softwaresystem zusammengestellt. Hierbei müssen die Gemeinsamkeiten zwischen Features und Anforderungen berücksichtigt werden.

Eine Software-Produktlinie ist nach Clements und Northrop eine Menge von softwareintensiven Systemen, die aus einer gemeinsam verwalteten Menge von Funktionen bestehen. Diese Funktionen oder auch Features werden dabei für unterschiedliche Kunden, Märkte oder Ziele jeweils durch eine Auswahl dieser Funktionen in einem zuvor beschriebenen Verfahren zusammengefasst und als eigenständiges Produkt ausgeliefert (Paul Clements und Linda Northrop 2001).

David L. Parnas definiert Programmfamilien wie folgt:

„We consider a set of programs to constitute a family, whenever it is worthwhile to study programs from the set by first studying the common properties of the set and then determining the special properties of the individual family members. A typical family of programs is the set of versions of an operating system distributed by a manufacturer“ (Parnas 1976)

Aus dieser Definition geht hervor, dass Programme einer Familie eine Menge an Gemeinsamkeiten aufweisen. Außerdem müssen genug Gemeinsamkeiten vorhanden sein, dass es aus Sicht der Entwicklung lohnenswert ist, diese Gemeinsamkeiten vor den Unterschieden zu betrachten. Aus dem von Parnas gelieferten Beispiel kann abgeleitet werden, dass dieser Definition der Gedanke von unterschiedlichen aber ähnlichen Softwareprodukten für unterschiedliche Kunden unterliegt.

Beide Definitionen haben das gemeinsame Ziel, ausgehend von einer gemeinsamen Plattform unterschiedliche Produkte in kürzerer Zeit an den Markt zu bringen und diese auf die Bedürfnisse der Kunden zuzuschneiden. Einer der wichtigsten Aspekte bei Software-Produktlinien ist dabei die konsequente Wiederverwendung von bereits implementierten Features und der Gedanke eines gemeinsamen Kerns. Durch diese Wiederverwendung von Programmteilen in unterschiedlichen Systemen, die von einer gemeinsamen Plattform abgeleitet sind, sollen teure Doppelentwicklungen vermieden werden (Böckle et al. 2005). Nach J. Bosch ist der Ansatz von Software-Produktlinien der erste intraorganistorische Softwarewiederverwendungsansatz, der sich als erfolgreich herausgestellt hat. (Bosch 2002)

Der Beginn einer Software-Produktlinie ist eine bewusste Entscheidung der Organisation und erfordert Aufwand für die Initialisierung der Produktlinie. Nach Bosch müssen bei der Initiierung zwei Dimensionen betrachtet werden. Zum einen muss beachtet werden, ob es sich um ein bestehendes Produkt handelt oder ob eine komplett neue Produktlinie aufgebaut werden soll. Zum anderen ist zu unterscheiden, ob ein evolutionärer oder ein revolutionärer Ansatz zu Beginn genutzt werden soll. Dabei hat jeder Fall bestimmte Risiken, die betrachtet werden müssen und bestimmte Vorteile, die sich ergeben können. In der nachfolgenden Tabelle 4 sind die Charakteristiken dieser Fälle kurz beschrieben.

Tabelle 4: Zwei Dimensionen der Software-Produktlinien-Initiierung nach (Bosch 2002)

	evolutionär	**revolutionär**
existierende Produkte	Entwicklung einer Vision für eine Produktlinienarchitektur, die auf den Architekturen der Familienmitglieder basiert. Komponenten werden schrittweise entwickelt, ggf. als Weiterentwicklung bestehender Komponenten.	Produktlinienarchitektur und Komponenten sind als Obermenge der bestehenden und für die Zukunft vorhergesagten Anforderungen entwickelt.
neue Produktlinie	Produktlinienarchitektur und Komponenten entwickeln sich mit den Anforderungen schrittweise durch neue Anforderungen weiter.	Produktlinienarchitektur und Komponenten werden direkt so entwickelt, dass sie die Anforderungen aller Produktlinienteilnehmer erfüllen können.

Ein Produkt entsteht durch die Nutzung eines gemeinsamen Kerns und konfigurierbaren Features oder Funktionen. In diesem Zusammenhang wird auch von Variationspunkten gesprochen. Ein Variationspunkt definiert dabei, welche Features oder Eigenschaften variieren können. Dabei werden auch zusätzliche Bedingungen im Variationspunkt mit verankert. Beispielsweise könnte der Variationspunkt eine Auswahl zwischen den Features A und B erlauben. Eine Bedingung, die mit in diesem Variationspunkt hinterlegt sein könnte, ist, dass, wenn Feature B ausgewählt wird zwangsweise auch Feature C in das Produkt mit einfließen muss, damit Feature B korrekt arbeiten kann. Eine Variante beschreibt dabei eine konkrete Ausprägung bzw. eine konkrete Variation, die ein „Produkt“ beschreibt (vgl. Pohl und Metzger 2008). In der Produktentwicklung werden also die Variationspunkte auf Grundlage der konkreten Anforderungen angebunden. Es werden die Varianten ausgewählt, die die zugrunde liegenden Anforderungen am besten abdecken können. Variationsmöglichkeiten können bspw. in Form von Feature-Diagrammen (van Gurp et al. 2001) oder UML Use Cases (Bühne 2002; von der Maßen, Thomas und Lichter 2003) visualisiert und verwaltet werden.

Ein weiteres potenzielles Einsparpotenzial ergibt sich bei Software-Produktlinien durch verringerte Testaufwände, da Features nur einmalig getestet werden müssen und nicht in jedem Produkt separat.

Software-Produktlinien sind zusammengefasst eine intraorganistorische und durch die Softwarearchitektur getriebene Vorgehensweise, um unterschiedliche Softwareprodukte für unterschiedliche Kunden mit kürzeren Entwicklungszyklen und einer hohen Qualität zu liefern. Jedoch ist die Adaption von bestehenden Produkten hin zu einer Produktlinie auch mit Risiken behaftet.

Der Ansatz von Software-Produktlinien weist eine ähnliche Zielsetzung auf, wie sie auch in dieser Arbeit beschrieben ist. Die Gemeinsamkeit ist, Doppelentwicklungen zu vermeiden und Entwicklerressourcen einzusparen, bspw. durch Wiederverwendung. Der Unterschied zu den in dieser Arbeit angestrebten Zielen ist, dass es sich bei Produktlinien bzw. Familien um mehrere (unterschiedliche) Systeme handelt und die Anforderungen innerhalb dieser Produkte umgesetzt werden müssen. Außerdem ist dieser Ansatz durch die Softwarearchitektur getrieben. Innerhalb dieser Arbeit soll zum einen eine Bündelung von Anforderungen behandelt werden, welche „unabhängig“ von der zugrunde liegenden Architektur ist. Zum anderen kann es, wie beschrieben, Anforderungen mit Synergiepotenzialen auch im Zusammenhang mit nur einem monolithischen System geben.

5 Überblick über die Lösungskonzeption

Ausgehend von der Beschreibung des Umfeldes sowie den identifizierten Handlungsfeldern in Kapitel 2.3 gibt dieses Kapitel einen Überblick über den gesamtheitlichen Lösungsansatz. Hierbei werden auch Erkenntnisse aus dem vorangestellten Kapitel 3, welches einen Überblick über den aktuellen Stand der Forschung bzgl. der Problemstellung liefert, mit verwendet.

Der grobe Überblick über die Lösungskonzeption, welcher hier vorgestellt wird, verweist an den jeweiligen Stellen auf die entsprechenden Kapitel, in denen die einzelnen Teillösungen erarbeitet, vorgestellt und diskutiert werden. Die nachfolgende Abbildung 13 gibt einen schematischen Überblick über die einzelnen Kapitel und ihre Zusammenhänge für die Lösungskonzeption.

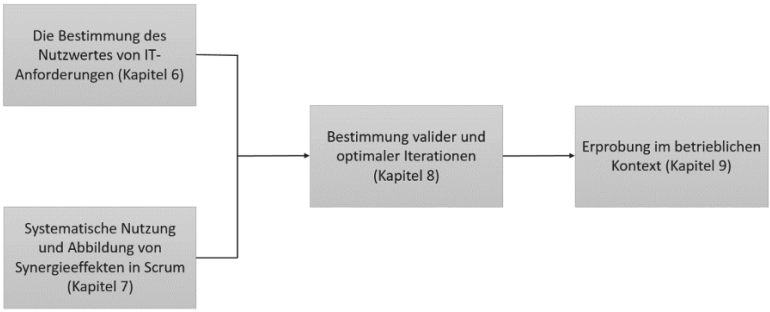


Abbildung 13: Aufbau der Arbeit im Überblick (eigene Darstellung)

Abbildung 13 zeigt, dass in den Kapiteln 6 und 1 zunächst „eigenständige Teillösungen" erarbeitet werden, die die Forschungsfragen RQ1 und RQ 2.1 aus Kapitel 2.3 beantworten. Die Lösungen zur Beantwortung der jeweiligen Forschungsfragen dienen im späteren Verlauf der Arbeit als Grundlage zur Beantwortung von RQ 2.2 und RQ 3 in Kapitel 1. Kapitel 1 zeigt anschließend, wie der Ansatz im industriellen Kontext erprobt wird und stellt entsprechende Resultate übersichtlich dar.

Ein zusätzliches Kapitel zur Skalierung von Scrum unter Einbeziehung mehrerer Teams ist nicht notwendig. In der Literatur (s. Kapitel 4.1) finden sich bereits mehrere Zusammenarbeitsmodelle, die diese Skalierung erlauben. Für die Integration mehrerer Teams in ein Projekt ist es somit lediglich notwendig die Rahmenbedingungen zu schaffen, sodass ein bereits bestehendes Modell angewendet werden kann.

In Kapitel 6 wird zunächst eine Methode erarbeitet, die es erlaubt den Nutzwert von Anforderunge zu quantifizieren. Die Literaturrecherche in Kapitel 4.2 hat gezeigt, dass bereits mehrere Methoden für die Nutzwertbestimmung und die damit verbundene Priorisierung bestehen. Die Recherche zeigt auch, dass die bestehenden Verfahren nicht direkt in einer rein internen IT mit mehreren Kunden für ein System angewendet werden können. Die Gründe hierfür sind, dass die bestehenden Verfahren teilweise keine Aggregation der Ergebnisse erlauben oder wie z.B. im Fall von „Mock Purchases" ein „echter Markt" vorausgesetzt wird.

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0_5

In Kapitel 6 wird daher erläutert, wie, ausgehend von einem systematischen Review bestehender Literatur zum Thema „value of IT“, unterschiedliche Nutzwertdimensionen von Anforderungen identifiziert werden. Anschließend wird diese Vorauswahl an Nutzwertdimensionen durch die enge Kooperation mit Projektverantwortlichen innerhalb der Konzern-IT von Volkswagen analysiert und es wird eine Menge an relevanten Nutzwertdimensionen bestimmt. Danach wird in diesem Kapitel erläutert, wie durch den Einsatz des „Analytisch-Hierarchischen-Prozesses“ (AHP) von Saaty (vgl. Saaty 1990) der Einfluss der unterschiedlichen, zuvor als relevant identifizierten Nutzwertdimensionen für die Bestimmung des Nutzwertes einer Anforderung bzw. User Story ermittelt werden kann. Ausgehend vom Ergebnis dieser Studie ist es möglich, den Nutzwert einer User Story transparent zu ermitteln und anschließend die Priorisierung der Produktrückstandselemente im Product Backlog für die Anforderer nachvollziehbar zu gestalten.

Darauf folgend wird in Kapitel 1 zunächst der Begriff „Synergiepotenzial zwischen Anforderungen“ genauer definiert. Ausgehend von dieser Definition werden die damit verbundenen Möglichkeiten für eine verbesserte Abarbeitung von Anforderungen gezeigt. Außerdem wird demonstriert, wie diese Potenziale in Scrum systematisch erschlossen werden können. Das Thema „Synergien auf Anforderungsebene“ mit einer entsprechenden Überführung in die Entwicklung ist zurzeit weder Betrachtungsgegenstand der Forschung zur Skalierung agiler Vorgehensmodelle noch der Forschung im Bereich der Softwareproduktlinien. Es besteht somit die Notwendingkeit die vorhandenen Potenziale auf eine vollkommen neue Art unf Weise zu erschließen.

Um dieses Ziel zu erreichen, wird die im traditionellen Scrum bestehende Systematik aus Epen, User Stories und Tasks (vgl. Pichler 2010) um ein neues Produktrückstandselement erweitert. Dieses neue Element fügt sich in der bestehenden Systematik zwischen User Stories und Tasks ein.

Der Aufbau dieses neuen Elements orientiert sich am Aufbau einer User Story. Der Unterschied, der es ermöglicht, Synergien zwischen User Stories systematisch abzubilden und für den Product Owner nachvollziehbar zu verwalten ist, dass dieses Element nicht mehr, wie klassische User Stories, einem Anforderer, sondern einer oder mehreren User Stories zugeordnet wird. Somit ergibt sich für den Product Owner die Möglichkeit bestehende Synergien zwischen Anforderungen zu identifizieren und diese nachvollziehbar zu verwalten. Für den einzelnen Anforderer ist es dadurch auch weiterhin möglich seine dedizierte Anforderung im Entwicklungsprozess zu verfolgen. Für die IT besteht jedoch die Möglichkeit, systematisch vorhandene Einsparpotenziale zu erschließen und zu nutzen. In Kapitel 1 wird dieses neue Element genauer definiert. Es wird aufgezeigt, wie dieses neue Produktrückstandselement im Kontext von Scrum anzuwenden ist. Außerdem wird auch auf die notwendigen prozessualen Änderungen eingegangen, um die Erschließung von Synergiepotenzialen im Prozess zu verankern.

Die Einführung eines neuen Produktrückstandselementes, die mit dem Ziel einer optimierten Abarbeitung von Anforderungen verbunden ist, stellt jedoch auch neue Anforderungen an die Iterationsplanung, bzw. Sprintplanung. In Kapitel 1 werden, ausgehend von der Nutzwertdefinition für User Stories aus Kapitel 6 und unter Verwendung eines neuen Produktrückstandselementes zwischen User Stories und Tasks, Richtlinien für einen validen Sprint identifiziert.

Ausgehend von der Definition eines validen Sprints wird unter Verwendung prädikatenlogischer Formeln (Kastens und Kleine Büning 2005) sowie der Mengenlehre (Meinel und Mundhenk 2006) eine mathematische Definition eines optimalen Sprints gegeben.

Ein solches Modell beantwortet in diesem Zusammenhang auch die Forschungsfrage RQ 3.1. Der aktuelle Stand der Forschung hat gezeigt, dass bereits ein Modell zur Modellierung von Iterationen besteht. Dieses Modell berücksichtigt jedoch keine Synergien zwischen Anforderungen. Außerdem verwendet dieses Modell ein abweichendes Optimierungskriterium. (vgl. Kapitel 4.3.1)

Das Modell beschreibt zunächst, welche Bedingungen eingehalten werden müssen, damit ein Iterationsplan als valide gilt. Dabei soll für jedes Entwicklerteam, von denen es potenziell mehrere geben kann, eine Teilmenge an Anforderungen bestimmt werden, sodass deren jeweilige spezifische Kapazität für die nächste Iteration nicht überschritten wird. Außerdem werden die teamspezifischen Kompetenzen mitberücksichtigt, da nicht in jedem Umfeld davon ausgegangen werden kann, dass jedes Entwicklerteam in der Lage ist alle Anforderungen vollumfänglich umzusetzen. Für eine valide Iterationsplanung ist es außerdem notwendig, dass nur die Anforderungen eingeplant werden, die auch umgesetzt werden können. Dieses bedeutet, dass die Anforderungen z. B. der jeweiligen „Definiton of Ready“ genügen müssen und keine weiteren, zeitaufwendigen Vorarbeiten mehr notwendig sind. Darüber hinaus wird durch das Modell abgesichert, dass keine Anforderung, User Story oder Product Owner Story mehr als einem Team gleichzeitig zugewiesen wird, um zeit- und kostenaufwendige Doppelentwicklungen zu vermeiden. Der letzte Punkt, der innerhalb des Modells mitberücksichtigt wird, sind feste Abgabefristen bzw. Deadlines, die für bestimmte Anforderungen gelten. Anforderungen, die eine harte Abgabefrist besitzen, müssen so eingeplant werden, dass diese eingehalten werden. Nur dann ist eine Iterationsplanung valide.

Das Optimierungskriterium einer Iterationsplanung ist der durch die Umsetzung zu erwartende Zuwachs an Nutzwert für das zu entwickelnde System. Bei der Betrachtung des Nutzwertes wird der komplette Nutzwert über alle Entwicklerteams betrachtet und nicht explizit der Nutzwert je Team. In Kapitel 1 wird dieses hier in natürlicher Sprache beschriebene Optimierungsproblem in ein mathematisches Modell überführt.

Für dieses Optimierungsproblem wird anschließend durch Reduktion gezeigt, dass dieses Sprint-Problem in der Entscheidungsvariante nur in nichtdeterministisch polynomieller Zeit (NP) (Hopcroft et al. 2002) gelöst werden kann bzw. in der Optimierungsvariante die NP-Schwere besitzt. Da ein Product Owner aufgrund seiner vielfältigen Verpflichtungen und Tätigkeiten nicht mehr zeitlich in der Lage ist eine optimale Sprintplanung durchzuführen, bzw. diese nicht mehr mit einem realistischen Zeitaufwand erstellen kann, werden ausgehend von der Problemdefinition unterschiedliche Lösungsmöglichkeiten untersucht und gegeneinander verglichen.

Da Probleme aus der Klasse NP-vollständig bzw. NP-schwer meist nicht mehr effizient von einem Computer berechnet werden können (Garey und Johnson 1979), werden neben der Möglichkeit einer optimalen Lösung des Sprint-Problems auch andere Lösungsmöglichkeiten betrachtet. Diese Möglichkeiten, zu einer Problemlösung zu gelangen, orientieren sich dabei an bereits für diese Klasse von Problemen erprobten Strategien, die auch in den Kapiteln 4.3.2 und 4.3.3 vorgestellt wurden. Hierbei ist zu prüfen, inwieweit diese Verfahren auf das Problem der Iterationsplanung und –optimierung angewendet werden können. Eine solche berechnete Lösung kann einem Product Owner zur Verfügung gestellt und als Grundlage für die Planung verwendet werden. Sie ist eine Art der Unterstützung, die für den Product Owner bei der Iterationsplanung algorithmisch geleistet werden kann.

Eine untersuchte Möglichkeit hierfür ist die Verwendung von Heuristiken, also mit begrenztem Wissen und innerhalb einer kurzen Zeitspanne zu einer guten, nicht zwangsweise optimalen, Lösung zu kommen (Gigerenzer und Todd 1999). Eine andere Möglichkeit, welche ebenfalls untersucht wird, ist die Verwendung eines evolutionären Algorithmus, welcher durch einfache an die Natur angelehnte Operationen und durch ein iteratives Vorgehen Lösungen für das Problem erzeugt. Darüber hinaus werden auch noch die Möglichkeiten für die Verwendung hybrider Ansätze untersucht. Bei hybriden Ansätzen werden eine oder mehrere mögliche Lösungen einer einfachen Heuristik verwendet und anschließend wird versucht, diese bspw. durch ein naturanaloges Verfahren zu verbessern. Eines dieser Verfahren ist das Simulated Annealing (vgl. Gerdes et al. 2004).

Diese unterschiedlichen Möglichkeiten werden miteinander bzgl. ihrer Performance (Laufzeit und Ergebnisgüte) mittels zufällig erzeugter Testdaten für das Sprint-Problem verglichen.

In Kapitel 1 werden die qualitativ besten Verfahren aus Kapitel 1 weiterführend evaluiert. Im ersten Schritt wird hierzu geprüft, welche Ergebnisse Projektverantwortliche auf einer ebenfalls zufällig erstellten Testinstanz erreichen. Diese Ergebnisse werden mit den Resultaten des algorithmischen Vorgehens in Verbindung gesetzt.

In einer zweiten Untersuchung werden mehrere reale Iterationen geplant. Die Planungsergebnisse werden mit den realen Ergebnissen verglichen. Anschließend erfolgt eine Auswertung der Untersuchungsergebnisse. Aus den Ergebnissen dieser beiden Studien werden auch Empfehlungen für den weiteren Einsatz abgeleitet.

Nach einer sich anschließenden Diskussion der Ergebnisse wird ein Ausblick für weitere Forschungen in diesem Kontext gegeben, bevor alle Ergebnisse in einem Fazit in Kapitel 1 zusammengefasst werden.

6 Die Bestimmung des Nutzwertes von IT-Anforderungen

Die Frage nach der Bewertung des Nutzwertes von Anforderungen, wie auch die Frage nach dem Nutzen von IT-Systemen, ist immer noch eine offene Forschungsfrage. Die Besonderheiten sind, dass Softwaresysteme von jedem Nutzer auf unterschiedliche Art und Weise für verschiedene Zwecke verwendet werden. In der Literatur lassen sich unterschiedliche Definitionen für den Nutzwert von Anforderungen und IT-Systemen finden. Viele dieser Definitionen sind jedoch nicht vollständig, da sie nicht alle Aspekte, die für eine ganzheitliche Betrachtung des Nutzwertes notwendig sind, berücksichtigen oder aber sie sind in unterschiedlichen Kontexten entstanden und für den jeweiligen Einsatzzweck angepasst.

Eine weitere Herausforderung bei der Ermittlung des Nutzens ist, dass dieser meist nur mittelbar entsteht und deswegen auch indirekte, geschäftsprozessabhängige Effekte identifiziert und verstanden werden müssen (Kohli und Devaraj 2003; Bitzer Philipp et al. 2014). Dieses wird auch durch Abbildung 14 verdeutlicht. Da keine unmittelbare Verbindung zwischen IT und Nutzen besteht, kann der Nutzen einer Anwendung somit immer nur im eingesetzten Kontext bestimmt werden. Deswegen ist es erforderlich eine kontextabhängige Methode zur Bewertung von IT-Anforderungen bereitzustellen und ein allgemeines Vorgehen zu erarbeiten, wie systematisch ein entsprechendes Bewertungssystem für den jeweiligen Kontext erarbeitet werden kann.

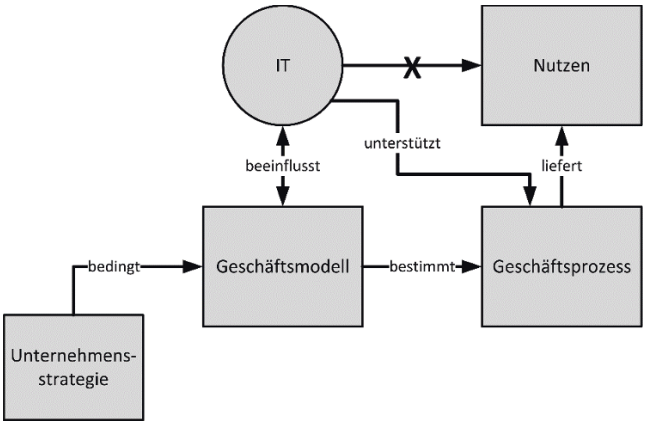


Abbildung 14: Wertbeitrag von IT (Bitzer Philipp et al. 2014)

In Scrum ist es die Aufgabe des Product Owners sicherzustellen, dass zum Ende einer Iteration möglichst viel Nutzwert für den Kunden erzeugt und dieser in einem potenziell auslieferbaren Paket zusammengefasst wird. Jedoch ist diese Aufgabe für einen Product Owner kaum leistbar, wenn es keine gültige und von allen Prozessbeteiligten anerkannte Definition des Nutzwertes gibt. Üblicherweise basiert die Abschätzung des Nutzwertes einer Anforderung auf dem Erfahrungswissen des jeweiligen Projektleiters oder des zuständigen Product Owners.

Bei Erfahrungswissen handelt es sich um eine organisierte und strukturierte Form des expliziten, wie auch des impliziten Wissens (Polanyi 1985), sodass die Erledigung von Tätigkeiten und die Erreichung von Arbeitszielen optimiert werden. Das auf Sachverhalte wie auch auf

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0_6

Vorgehensweisen bezogene Wissen steht im Dienst einer effizient angelegten Handlungsausführung (Bolte und Martin 1992). Somit ergibt sich Erfahrungswissen aus dem Zusammenspiel vom expliziten, praktischen und theoretischen Wissen wie auch aus implizitem, nicht direkt ausdrückbarem aber anwendbarem Wissen. Auf Grundlage von Erfahrungswissen ergibt sich eine differenzierte Erfassung des Geschehens und von Informationen und einer daraus abgeleiteten effizienten, ergebnisorientierten Bewältigung von Arbeitsaufgaben (Plath 2002). Dieses Zusammenwirken wird in Abbildung 15 zusammenfassend dargestellt.

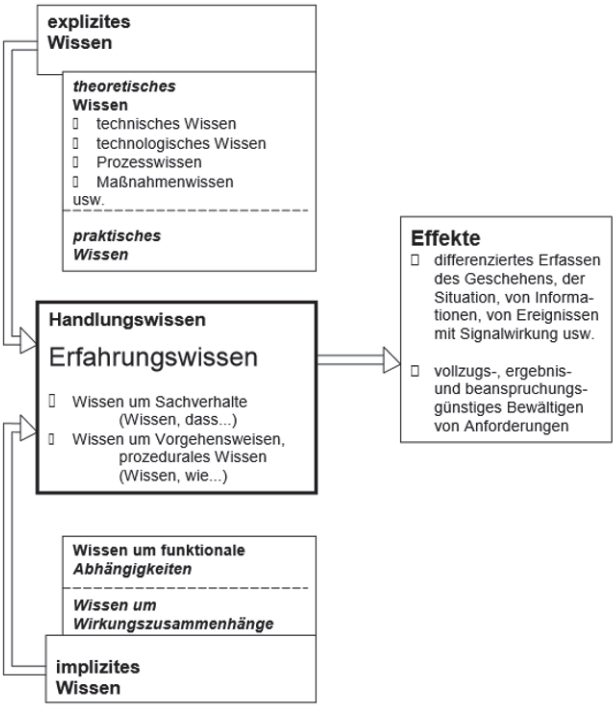


Abbildung 15: Einordnung und Effekte von Erfahrungswissen, Quelle: (Plath 2002)

Es kann also davon ausgegangen werden, dass Product Owner mit einer mehrjährigen Berufserfahrung geübt und auch effizient bei der Abschätzung des Nutzwertes sein können, da sie sich eine ausreichende Menge an explizitem, wie auch implizitem Erfahrungswissen angeeignet haben. Eine Abschätzung beruht also auf dem angesammelten Erfahrungswissen und der dadurch differenzierten Betrachtung der Rahmenbedingungen und der vorliegenden Informationen, die für eine Priorisierung verwendet werden können.

Priorisierungen, die auf der Grundlage eines schlecht zu quantifizierenden und teilweise nur implizit vorhandenen Erfahrungswissens beruhen, sind für Kunden, Stakeholder und andere Prozessbeteiligte schwer nachvollziehbar. Die abschließende Priorisierung der Anforderungen ist für die Prozessbeteiligten wenig transparent.

Das Ziel, welches in diesem Kapitel verfolgt wird, ist eine mögliche Definition für den Nutzwert von IT-Anforderungen im Kontext von Managementinformationssystemen für die Volkswagen AG zu erarbeiten. Damit soll sichergestellt werden, dass die Priorisierung der Anforderungen, welche in Form von User Stories im Product Backlog verwaltet werden, für den Kunden und für die am Prozess beteiligten Personen transparenter wird. Außerdem soll durch die Verwendung einer transparenten Nutzwertdefinition für IT-Anforderungen und deren Verwendung bei der (teil-)automatisierten Optimierung von Iterationen, die Akzeptanz dieses Lösungsansatzes bei den am Prozess beteiligten Personen gesteigert werden.

6.1 Vorgehen zur Definition des Nutzwertes von IT-Anforderungen

Für die Erstellung einer Definition bzgl. des Nutzwertes von IT-Anforderungen im Kontext von Managementinformationssystemen ist ein vierstufiges Vorgehen angedacht. Im ersten Schritt werden zunächst mögliche Nutzwertdimensionen, welche in diesem speziellen Kontext relevant sein könnten, definiert. Um diese zu definieren, wird sowohl auf bestehende Literatur zu diesem Thema zurückgegriffen, als auch auf das angesammelte Erfahrungswissen von Product Ownern und Projektleitern innerhalb der IT.

In diesem Unterkapitel (6.1) wird zunächst das allgemeine Vorgehen für die Erarbeitung der Definition des Nutzwertes beschrieben.

Im folgenden Unterkapitel (6.2) wird dargestellt, welche unterschiedlichen Dimensionen desselben für eine ganzheitliche Bestimmung des Nutzwertes betrachtet werden müssen. Hierbei ist eine mehrdimensionale Bewertung der IT-Anforderungen notwendig, denn eine ausschließlich auf finanzielle Effekte fokussierte Betrachtung liefert keine ausreichende Bewertungsgrundlage für die Bestimmung des Nutzwertes einer Anforderung. Auch wenn z. B. Cleland-Huang und Denne (2005) den Fokus auf die finanzielle Bewertung von Anforderungen und die Erstellung von sogenannten „minimal vermarktbaren Features“ legen, wird davon ausgegangen, dass diese auf finanzielle Effekte beschränkte Sicht für die Priorisierung von Anforderungen einer internen IT-Abteilung innerhalb eines Konzerns nicht ausreichend ist (s. auch Kapitel 4.2). Deswegen ist es notwendig, den Nutzwert mehrdimensional zu betrachten, um eine ganzheitliche Sicht auf den zu erwartenden Nutzen zu erhalten (vgl. Bitzer Philipp et al. 2014). Ein Grund für eine mehrdimensionale Nutzwertbetrachtung ist, dass nicht alle Anforderungen a priori auf finanzielle Effekte hin bewertet werden können. So sind z. B. Anforderungen, welche sich z. B. auf die Anpassung an das Corporate Design beziehen, fast unmöglich mittels finanzieller Effekte zu bewerten.

Auf Basis der zunächst identifizierten Nutzwertdimensionen wird dann im dritten Unterkapitel (6.3) beschrieben, wie die für den speziellen Kontext von Managementinformationssystemen relevanten Nutzwertdimensionen ausgewählt wurden. Daran anschließend erfolgt eine Gewichtung der unterschiedlichen relevanten Nutzwertdimensionen, um ihren Beitrag am gesamten Nutzwert einer Anforderung zu bestimmen. An dieser Stelle wird die AHP-Methode von Saaty (s. z. B. Saaty 1990) verwendet, um zu einem Gewichtsvektor zu gelangen.

Nachdem der Nutzwert für eine Anforderung auf Grundlage der als relevant identifizierten Nutzwertdimensionen und deren Gewichtung bestimmt werden kann, wird im letzten Unterkapitel (6.5) diskutiert, wie der endgültige Nutzwert einer Anforderung ermittelt werden kann. Anforderungen für IT-Systeme, die weltweit im Einsatz sind, beinhalten zum einen Funktionen,

die bspw. Prozesse in mehreren Produktionsstandorten gleichermaßen unterstützen. Eine Verbesserung der bestehenden Software innerhalb „standardisierter“ Funktionen hat eine hohe Anzahl an Profiteuren, da nicht nur der Anforderer, bzw. die von ihm vertretene organisatorische Einheit, profitiert, sondern eine Vielzahl an Nutzern in unterschiedlichen Produktionsstandorten. Zum anderen beinhalten solche Softwaresysteme häufig auch Funktionen, die zur Unterstützung einzelner Prozessschritte und zur Unterstützung von örtlichen Besonderheiten benötigt werden. Anforderungen zur Implementierung oder Verbesserung einer solchen Funktionalität sind meist von begrenztem Nutzen, da sie nur innerhalb einer oder weniger organisatorischer Einheiten gewinnbringend genutzt werden können. Somit muss neben der Bewertung der einzelnen nutzbringenden Eigenschaften auch die Anzahl der Profiteure bei Abschätzung des Nutzwertes einer Anforderung im jeweiligen Unternehmenskontext mit berücksichtigt werden.

Dieses hier theoretisch beschriebene Vorgehen zur Defintion eines Nutzwertes für IT-Anforderungen muss eine auf das Umfeld von BI-Systemen angepasste Menge an Nutzwertdimensionen liefern, sodass dieser mehrdimensional betrachtet werden kann. Im weiteren Schritt wird ein Gewichtsvektor erzeugt, der diese mehrdimensionale Betrachtung auf einen Zahlenwert abbildet. Dadurch sind die Nutzenbewertungen auch potenziell aggregierbar.

Im letzten Schritt erfolgt die Auswertung der Ergebnisse. Die nachfolgende Abbildung 16 verdeutlich diesen Ablauf grafisch.

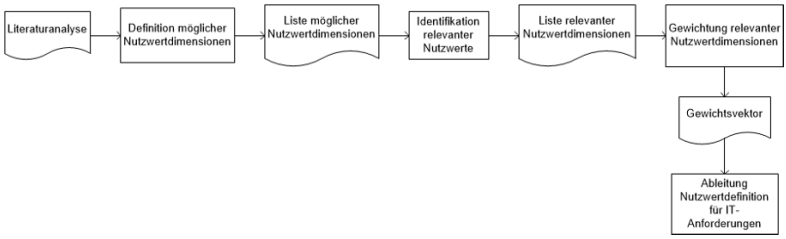


Abbildung 16: Grafische Präsentation des Prozesses zur Nutzwertdefinition (Quelle: eigene Darstellung)

Abschließend wird ebenfalls diskutiert, ob dieses mehrstufige Verfahren auch auf andere Bereiche angewendet werden kann.

6.2 Identifikation möglicher Nutzwertdimensionen

Um die möglichen Nutzwertdimensionen für den Kontext von Management-Informationssystemen zu sammeln, wurde zunächst durch unstrukturierte Beobachtung einer Diskussionsrunde zwischen mehreren Projektleitern, Product Ownern und repräsentativen Anwendern, welche die einzelnen Anwenderrollen repräsentieren (Key-User), ein erster Eindruck zur Thematik und Gewichtung der Schwerpunkte gewonnen.

Dieser Eindruck wurde anschließend durch die Reflexion der Erkenntnisse mit mehreren Projektverantwortlichen verifiziert. Hierbei ergab sich, dass zurzeit keine allgemeingültige Definition des Nutzwertes von IT-Anforderungen im entsprechenden Kontext vorhanden ist.

Außerdem stellte sich heraus, dass verschiedene Prozessbeteiligte den Anforderungen beigemessene Qualitäten voneinander abweichend bewerten. Auch an dieser Stelle zeigt sich, dass die Resultate einer Priorisierung unterschiedlicher Anforderer aggregierbar sein müssen.

Als Konsequenz dieser Beobachtungen und unter der Zielsetzung, den Nutzwert für IT-Anforderungen für alle Prozessbeteiligten transparent bestimmen zu wollen, wurde zunächst auf eine bestehende Literaturanalyse und Auswertung von Bitzer, Heidecke und Leimeister zurückgegriffen. Ihre Untersuchungen zur Identifikation von Nutzenwerten für IT-Systeme wurden ebenfalls in einem ähnlichen Kontext bei Volkswagen durchgeführt und veröffentlicht (Bitzer Philipp et al. 2014). Aus den zur Verfügung stehenden Unterlagen zu dieser Studie und dem von ihnen abgeleiteten Referenzmodell zur systematischen Identifikation des Nutzens von IT-Anwendungen, welches in Abbildung 17 dargestellt ist, wurden zunächst Vorschläge für mögliche Nutzwertdimensionen von IT-Anforderungen erarbeitet. Das Ergebnis dieser Studie ist jedoch keine Reproduktion der Ergebnisse und des Vorgehens von Bitzer et al. Die Studie bildet vielmehr die Grundlage für die hier durchgeführten Untersuchungen. Das hier erarbeitete und im industriellen Kontext erprobte Vorgehen stellt zwei grundsätzliche Erweiterungen zu den Ergebnissen der oben genannten Studie dar. Zum einen werden nicht Nutzeneffekte von Softwaresystemen erfasst, sondern Nutzeneffekte auf Anforderungsebene. Zum anderen endet das hier präsentierte Vorgehen nicht bei der systematischen Identifikation von Nutzeneffekten. Das hier vorgestellte Vorgehen erlaubt es vielmehr die Nutzeneffekte zu identifizieren, zu gewichten und die Nutzeneffekte der unterschiedlichen Nutzwertdimensionen zu einem gesamtheitlichen Nutzwert zu aggregieren.

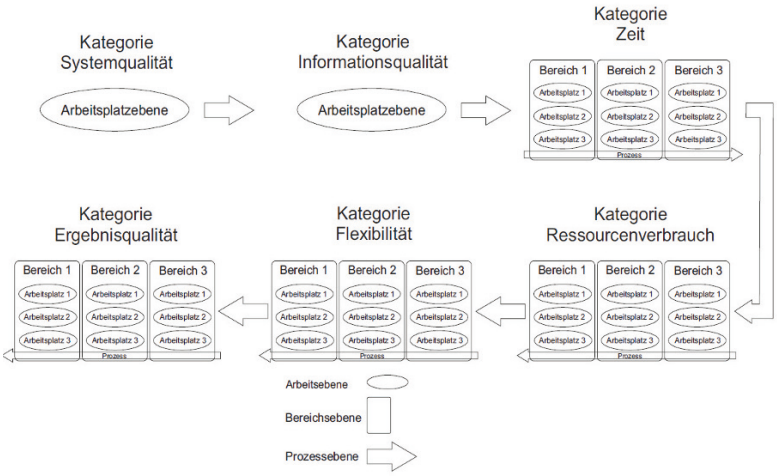


Abbildung 17: Aufbau eines IT-Nutzenformulars (eigene vereinfachte Darstellung in Anlehnung an (Bitzer Philipp et al. 2014))

In Abbildung 18 wird die Abgrenzung zwischen der Studie von Bitzer und der Studie dieser Arbeit verdeutlicht.

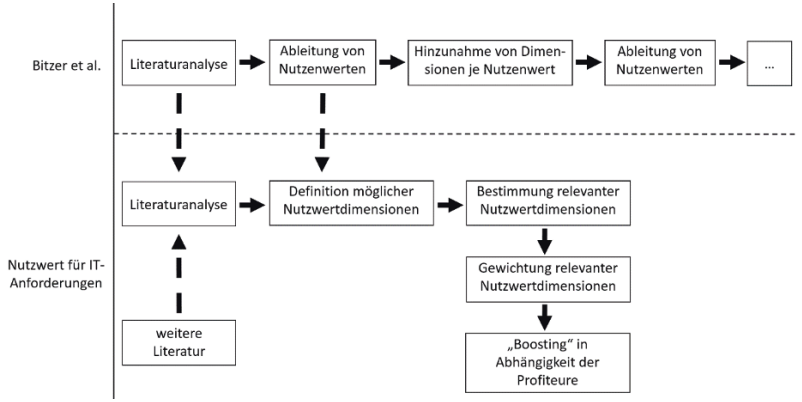


Abbildung 18: Unterschiede zwischen dem Vorgehen von Bitzer et al. (2014) und der angestrebten Vorgehensweise

Der Abbildung ist zu entnehmen, dass beide Studien beginnend von der Literaturanalyse bis hin zur Definition der relevanten Nutzwertdimensionen nahezu identisch vorgehen. Die sich anschließenden Schritte dieser Studie können somit als Erweiterung und Anpassung des bestehenden Vorgehens verstanden werden. Innerhalb des parallelen Strangs der Studien wurden in die Ergebnisse dieser Literaturrecherche die Ergebnisse von Bitzer mit einbezogen, jedoch bzgl. der Bewertung von IT-Anforderungen erweitert. Die weiteren Schritte bauen somit auf einer anderen Grundlage auf. Die weiteren Untersuchungen beginnen mit der Auswahl möglicher Nutzwertdimensionen bis hin zur Definition der relevanten Nutzwertdimensionen und sind somit unabhängig von der Studie, die in (Bitzer et al. 2014) veröffentlicht wurde. Sämtliche Befragungen sind unabhängig von den bestehenden Ergebnissen entstanden.

Unterschiedliche mögliche Ansätze zur systematischen Identifikation des Nutzens von IT-Anforderungen wurden anschließend in mehreren Diskussionsrunden mit Prozessbeteiligten bei Volkswagen in der Unterabteilung für Managementsysteme besprochen und verfeinert. So wurde bspw. entschieden, die nach der Umsetzung einer IT-Anforderung zu erwartenden Nutzwertpotenziale nicht mehr zusätzlich nach Arbeitsplatz-, Bereichs- und Prozessebene zu untergliedern. Der Hintergrund ist, dass die Zuordnung von Nutzeneffekten und dem damit verbundenen Nutzwert einer Anforderung sich häufig nicht direkt einem dieser Bereiche zuordnen lässt. Eine solche Untergliederung wird, nach Aussage der Beteiligten, erst auf Systemebene für sinnvoll erachtet.

Als Ergebnis mehrerer Abstimmungsrunden mit unterschiedlichen Projektleitern und Product Ownern der IT (N = 6) sowie mit Vertretern des Kunden (N = 3) wurden zunächst sechs allgemeine Nutzwertdimensionen definiert. Um diese Nutzwertdimensionen zu definieren, wurden zu Beginn die von Bitzer et al. vorgeschlagenen Nutzwertdimensionen innerhalb des IT-Nutzenformulars genauer betrachtet. Bei den vorgeschlagenen Kategorien handelt es sich um:

- Systemqualität
- Informationsqualität
- Zeit

- Ressourcenverbrauch
- Flexibilität
- Ergebnisqualität

Zusätzlich wurden weitere aus der Literatur bekannte Nutzeneffekte mit betrachtet, hierbei handelt es sich bspw. um den intangiblen Nutzwert nach Irani (2002), den strategischen Nutzwert nach (Bradley et al. 2011) oder aber auch um negative Nutzeneffekte, wie z. B. den relativen Schaden, die durch K. Wiegers geprägt wurden (Wiegers 1999). Bei der initialen Erarbeitung von Nutzwertdimensionen sind aber auch unterschiedliche Erfahrungen der an diesem Prozess Beteiligten mit eingeflossen.

Als Ergebnis des Schritts der Definition möglicher Nutzwertdimensionen sind zunächst sechs dieser Dimensionen definiert worden. Hierbei handelt es sich teilweise um Nutzeneffekte, die in der Literatur beschrieben sind, sowie um neue Nutzwertdimensionen, die aus der Verbindung mehrerer Nutzeneffekte und deren Übertragung auf das konkrete Umfeld von Managementinformationssystemen entstanden sind. Im Folgenden werden zunächst diese sechs Nutzwertdimensionen genauer definiert und durch Beispiele illustriert.

Finanzieller Nutzwert

Unter der Nutzwertdimension des finanziellen Nutzwertes werden alle zu erwartenden Nutzeneffekte zusammengefasst und bewertet, die zu einer direkten und bewertbaren Kostenreduktion führen. Ein Beispiel hierfür wäre die Erstellung und Durchführung eines optimierten Deployments von Softwaresystemen oder deren Komponenten (Ruehl et al. 2012). Durch ein optimiertes Deployment können z. B. zur Verfügung stehende Hardwareressourcen effizienter genutzt und ggf. auch Instanzen eingespart werden. Somit ist es bspw. möglich einen Server komplett abzuschalten, wodurch sowohl die Betriebskosten als auch die Wartungskosten direkt reduziert werden können.

Arbeitsorganisatorischer Nutzwert

Anforderungen bzw. User Stories, welche einen arbeitsorganisatorischen Nutzwert aufweisen, ermöglichen es einem Anwender, nach deren Umsetzung eine Aufgabe am System einfacher, weniger fehleranfällig oder schneller umzusetzen. Bei dieser Nutzwertdimension liegt das Hauptaugenmerk auf der effizienten Benutzbarkeit des Systems unter der Berücksichtigung der Arbeitsumstände, unter denen das System durch die Anwender genutzt wird. Ein einfaches Beispiel für die Steigerung des arbeitsorganisatorischen Nutzwertes eines Systems wäre bspw. die folgende fiktive User Story:

„Als Datenerfasser in der Produktion möchte ich durch eine Schaltfläche auf der Startseite direkt in die Eingabemaske zur Erfassung der Lagerbestände gelangen, damit ich diese schneller eingeben kann und den entsprechenden Menüpunkt nicht in Untermenüs suchen muss."

Die entsprechende Umsetzung dieser User Story durch die IT würde dem Anwender, in diesem Fall dem Datenerfasser, die Möglichkeit geben, einen häufig genutzten Workflow schneller und effizienter durchzuführen. Der Anwender muss sich nicht mehr merken, in welchem Untermenü sich die entsprechende Schaltfläche zur Aktivierung des Workflows verbirgt und darüber hinaus wäre der Anwender in der Lage diesen Workflow schneller zu starten, was zu einer optimierten Benutzbarkeit des Systems führt.

Tertiärer / Intangibler Nutzwert

Durch den tertiären Nutzwert werden Eigenschaften einer Anforderung bewertet, die keinen direkten Einfluss auf die Funktionalität oder die Gebrauchstauglichkeit des Softwareproduktes haben. Mit dieser Nutzwertdimension werden vielmehr Veränderungen bzgl. der Außenwirkung eines Systems bewertet.

Ein Beispiel für eine Anforderung, welche ausschließlich einen tertiären Nutzwert erzeugt, wäre bspw. folgende fiktive, als User Story formulierte, Anforderung:

„Als iPad-Nutzer der Applikation Ö möchte ich, dass das App-Icon[4] neu gestaltet und an die Organisationsrichtlinien angepasst wird, um die Präsentationsfähigkeit und Vorzeigbarkeit zu steigern."

Eine Umsetzung dieser fiktiven Anforderung hätte vermutlich keine Auswirkungen auf die Applikation selbst, jedoch ergeben sich für die Sichtbarkeit und Vermarktung Vorteile. Durch eine eindimensionale Betrachtung des Nutzens von Anforderungen könnte dieser Nutzeneffekt nicht mit berücksichtigt werden.

Negativer Nutzwert bei Nichtumsetzung

Die Nutzwertdimension des negativen Nutzwertes bei Nichtumsetzung orientiert sich an dem von Karl Wiegers vorgeschlagenen negativen Nutzwert bzw. relativen Schaden (Wiegers 1999). Er beschreibt den Nutzwertverlust durch die Nichtumsetzung. Danach kann dieser Verlust, der durch die Umsetzung verhindert wird, als positiver Nutzwert der jeweiligen Anforderung aufgefasst werden. Zur besseren Illustration dieses Sachverhaltes dient das folgende Beispiel:

Managementinformationssysteme sind meist auf Daten anderer Systeme, bspw. aus der Produktion oder dem Bereich Human Ressources, angewiesen, um die von den Systemen erhobenen Daten zu aggregieren und für ein entsprechendes Berichtswesen aufzubereiten und zu visualisieren. Wenn es nun zu einer Änderung in einem dieser benötigten (Vor-) Systeme kommt, sei es durch ein neues Release oder die Migration vom bestehenden System hin zu einem neuen System, kann es notwendig werden, dass die Schnittstelle zum Datenaustausch angepasst werden muss. An dieser Stelle hat die IT-Abteilung, welche mit der Entwicklung des Managementinformationssystems betraut ist, mindestens zwei Alternativen. Zum einen ist es möglich, nichts zu tun. Dieses würde in der Konsequenz dazu führen, dass das eigene System nicht mehr mit allen benötigten Daten versorgt wird und somit bestehende Funktionalitäten nicht mehr genutzt werden können. Zum anderen wäre eine Alternative, die bestehende Schnittstelle an die neuen Gegebenheiten anzupassen, sodass das System weiterhin mit notwendigen Daten versorgt wird. An dieser Stelle entsteht jedoch Aufwand für eine entsprechende Umsetzung, welcher nicht zu neuen Funktionalitäten oder einem gesteigerten Nutzwert der Anwendung führt. Es wird lediglich sichergestellt, dass bereits vorhandene Funktionalität weiterhin verfügbar ist, somit wird der Nutzwertverlust verhindert. Dies kann jedoch als positiver Nutzwert in Höhe des potenziellen Nutzwertverlustes bewertet werden.

[4] App-Icon: Die grafische Darstellung der Schaltfläche zum Starten einer Applikation

Softwarequalitativer Nutzwert

Unter dem Begriff Softwarequalität wird die Gesamtheit der Merkmale und Merkmalswerte eines Softwareproduktes verstanden, die sich auf dessen Eignung, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen, beziehen. (Balzert 1998)

Ein Beispiel für ein Qualitätsmodell ist der ISO-Standard ISO/IEC 9126 (ISO/IEC 2001). Dieser unterteilt Softwarequalität in sechs unterschiedliche Gruppen von Qualitätsmerkmalen: Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit. In der nachfolgenden Abbildung 19 werden diese Qualitätskriterien nochmals dargestellt und um jeweilige Untermerkmale erweitert.

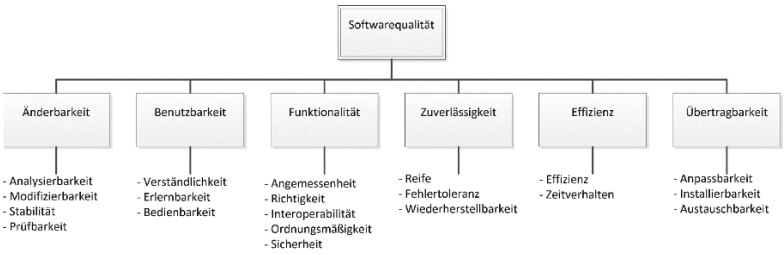


Abbildung 19: Qualitätskriterien der Softwarequalität nach ISO 9126, eigene Darstellung

Die Abgrenzung zwischen den Kriterien „Erlernbarkeit“ und „Bedienbarkeit“ aus dem Bereich der „Benutzbarkeit“ ist nicht absolut trennscharf zum bereits definierten arbeitsorganisatorischen Nutzwert möglich. Jedoch ist die Nutzwertdimension der Softwarequalität essenziell für die ganzheitliche Bewertung der Anforderungen im Systemkontext.

Anforderungen, denen ein softwarequalitativer Nutzwert beigemessen werden kann, sind z. B. technische Anforderungen, die die Ausfallsicherheit eines Systems positiv beeinflussen, ggf. bestehende Sicherheitslücken schließen oder neue Funktionalität bereitstellen, um die Angemessenheit des Softwaresystems bzgl. seiner Verwendung durch die Anwender zu verbessern.

Strategischer Nutzwert

Die Dimension des strategischen Nutzwertes bewertet die Eigenschaften einer Anforderung sich über einen längeren Zeitraum hinweg strategischen Zielen zu nähern oder die Zielerreichung der strategischen Ziele messbar und auswertbar zu gestalten. Dieser Nutzenwert wird z. B. auch im „value-orientated prioritization framework“ von Azar et al. verwendet (Azar et al. 2007).

Durch die Erlangung der Berichtsfähigkeit über die Zielerreichung langfristiger strategischer Ziele können bspw. Maßnahmen seitens des Kunden hinsichtlich ihrer Auswirkung auf die Zielerreichung des strategischen Ziels hin bewertet und nachverfolgt werden. Für ein Managementinformationssystem ergeben sich hierdurch jedoch nicht zwangsweise unmittelbare und bewertbare Auswirkungen hinsichtlich der anderen bereits definierten Nutzwertdimensionen.

Ein weiterer Aspekt des strategischen Nutzwertes ist auch die Berücksichtigung strategischer Ziele der IT-Abteilung innerhalb des Unternehmens, da auch diese organisatorische Einheiten

strategische Ziele hinsichtlich eines Systems, dessen Verbreitung und Weiterentwicklung verfolgen. Auch ist es die Aufgabe einer internen IT-Abteilung innerhalb eines großen Konzerns, nachhaltige und perspektivische Entscheidungen zu treffen.

Da der strategische Nutzwert sowohl auf Seiten des Kunden als auch auf Seiten der IT für eine ganzheitliche Bewertung berücksichtigt werden muss und an dieser Stelle die Bewertung der Nutzendimension nicht mehr ausschließlich durch eine Person erfolgen kann, wurde entschieden, die Dimension des strategischen Nutzwertes auf zwei Nutzwertdimensionen aufzuteilen. Zum einen soll der strategische Nutzwert für den Kunden bewertet werden. Dies geschieht im Regelfall durch den Kunden oder einen Key-User, der den strategischen Nutzwert bewerten kann. Der strategische Nutzen auf Seiten der IT sollte, zum anderen, unabhängig durch den durch die IT gestellten Product Owner oder einen mit der Aufgabe betrauten Projektverantwortlichen erfolgen. Die Notwendigkeit dieser Aufspaltung ergibt sich auch aus der Tatsache, dass die strategischen Ziele des oder der Kunden in Konflikt mit den strategischen Zielen der IT-Abteilung stehen können.

6.3 Bestimmung relevanter Nutzwertdimensionen

Nach der Definition möglicher Nutzwertdimensionen wird in einem anschließenden Schritt geprüft, ob die zuvor mit Projektverantwortlichen und Prozessbeteiligten aus dem Umfeld von Managementinformationssystemen erarbeiteten Nutzwertdimensionen auch eine Relevanz bei der eigentlichen Priorisierung der Anforderungen bzw. der Priorisierung der User Stories innerhalb des Product Backlogs besitzen.

Um die Relevanz der unterschiedlichen Nutzwertdimensionen zu bestimmen, erfolgte eine Befragung von Projektleiter, Product Owner und Mitarbeiter der IT-Planung aus unterschiedlichen Hierarchieebenen. Die Teilnehmer wurden dabei so ausgewählt, dass unterschiedliche Sichtweisen und Erfahrungen aus unterschiedlichen Projekten und aus verschiedenen hierarchischen Ebenen reflektiert werden. Dabei wurde darauf geachtet, dass sowohl die Erfahrungen der IT ($N = 10$) als auch die Sichtweise der Anforderer über die unterschiedlichen Hierarchieebenen hinweg betrachtet werden ($N = 11$). Zu diesem Zweck gab es zwei zeitlich versetzte Treffen, um die Zahl der Teilnehmer je Befragungsrunde zu begrenzen. Wegen der Kürze der Befragung wurde beschlossen, diese in Gruppen durchzuführen mit 10, respektive 11 Teilnehmern. Es wäre auch möglich gewesen diese Befragung in separaten Terminen oder die Erhebung mittels eines Fragebogens durchzuführen. Bei einer Befragung mittels eines Fragebogens ist es jedoch problematisch auf mögliche Rückfragen direkt zu reagieren und sicherzustellen, dass alle Beteiligten ein möglichst ähnliches Verständnis von den zuvor definierten Nutzwertdimensionen erhalten und eine Verzerrung durch unterschiedliches Vorwissen der Beteiligten minimiert wird.

Die Hypothese dieser Untersuchung ist, dass jede zuvor definierte Nutzwertdimension mindestens für einen Teil der Befragten relevant ist.

Beide Befragungsrunden liefen nach einem identischen Schema ab. Zunächst wurden die Teilnehmer begrüßt und das Anliegen der Befragung kurz umrissen. Anschließend wurde jede Nutzwertdimension mittels ihrer Definition und anhand eines Beispiels vorgestellt. Im nächsten Schritt wurden Unklarheiten bzw. Verständnisprobleme der jeweiligen Nutzwertdimension be-

sprochen, um allen Teilnehmern mit unterschiedlichem Vorwissen eine ähnliche Wissensgrundlage für ihre Entscheidung zu ermöglichen. Darauf aufbauend wurde für die jeweilige Nutzwertdimension per Handzeichen abgestimmt, ob diese aus ihrer Sicht relevant für die Nutzwertbestimmung und der sich daraus ableitenden Priorisierung der Elemente im Product Backlog ist. Anschließend wurden die Meldungen ausgezählt und protokolliert. Die erhobenen Daten sind an dieser Stelle nicht personengebunden und wurden aggregiert. Um menschliche Fehler beim Auszählen zu vermeiden, haben sowohl der Präsentator als auch der Protokollant die Auszählung unabhängig voneinander vorgenommen und nach jeder Auszählung verglichen.

Tabelle 5 zeigt die Ergebnisse der beiden Befragungsgruppen in aggregierter Form. Wie dieser Tabelle zu entnehmen ist, zeigt sich, dass alle 21 Befragten die Nutzwertdimensionen des finanziellen, wie auch des arbeitsorganisatorischen Nutzwertes zu 100 % als relevant erachten. Mit einer Zustimmung von über 90 % folgt die Nutzwertdimension des negativen Nutzwertes bei Nichtumsetzung. Es schließen sich der softwarequalitative Nutzwert, der tertiäre Nutzwert und der strategische Nutzwert aus Sicht des Kunden, an. Mit einer Zustimmung von lediglich 14,29 % zum strategischen Nutzwert der IT erhält diese Nutzwertdimension die geringste Zustimmung.

Tabelle 5: Relevanz der definierten Nutzwertdimensionen

Nutzwertdimension	Akzeptanz in [%]
finanzieller Nutzwert	100,00
arbeitsorganisatorischer Nutzwert	100,00
negativer Nutzwert bei Nichtumsetzung	90,48
softwarequalitativer Nutzwert	85,71
tertiärer Nutzwert	61,90
strategischer Nutzwert (Kunde)	57,14
strategischer Nutzwert (IT)	14,29

Anmerkung: (N = 21)

Bei einer späteren Reflexion und Diskussion der Ergebnisse wurde vor allem die niedrige Zustimmung zum strategischen Nutzwert der IT nochmals genauer angesprochen. Die geringe Zustimmung leitet sich aus vielfältigen Gründen ab.

Eine Begründung war bspw., dass zurzeit eher die Rolle als Dienstleister im Vordergrund steht und nicht die des Innovators und Lenkers. Jedoch konnte festgestellt werden, dass der Nutzwertdimension des strategischen Nutzwerts der IT bei der Priorisierung mehr Beachtung geschenkt werden müsste und vorhandene Strategien konkretisiert werden sollten. Aufgrund dieser Befunde wurde beschlossen, diese Nutzwertdimension weiter mit in die nächste Phase zu nehmen und nicht zu streichen, auch wenn es die Untersuchungsergebnisse an dieser Stelle nahelegen würden.

6.4 Gewichtung der als relevant identifizierten Nutzwertdimensionen

Ausgehend von den im vorherigen Schritt als relevant identifizierten Nutzwertdimensionen ist es nun notwendig den Einfluss der jeweiligen Nutzwertdimension auf den gesamten Nutzwert zu bestimmen. Wie bereits die vorherige Studie gezeigt hat, scheinen nicht alle Nutzwertdimensionen von gleicher Relevanz zu sein. Deswegen ergibt sich die Hypothese, dass die unterschiedlichen Nutzwertdimensionen eine unterschiedliche Bedeutung für die Definition des gesamten Nutzwertes einer Anforderung haben.

Zur Prüfung dieser Hypothese wurden Befragungen im Rahmen des betrieblichen Kontextes durchgeführt. Durch eine strukturierte Befragung von Prozessbeteiligten und die Auswertung der Ergebnisse soll der Zusammenhang und die Priorisierung der unterschiedlichen Nutzwertdimensionen vorgenommen werden. Um eine Priorisierung vorzunehmen, eignen sich grundsätzlich unterschiedliche Methoden. Eine Möglichkeit ergibt sich z. B. durch die Verwendung des kumulativen Votings (CV), welches auch als 100-$-Test bekannt ist (Berander und Jönsson 2006). Hierbei erhält jeder Befragte 100 Punkte, die er beliebig zwischen den unterschiedlichen Möglichkeiten aufteilen kann. Anschließend werden die Ergebnisse aller Teilnehmer aggregiert und abschließend wird das Ergebnis normiert. Somit kann mittels einer sehr einfachen Methodik eine Priorisierung vorgenommen werden. Der Nachteil dieser Methode ist jedoch, dass die Ergebnisse eher ungenau sind und die Verhältnisse untereinander nicht ausreichend genau angegeben werden. Das Ziel liegt hier eher bei der Reihenfolge der zu priorisierenden Elemente als bei der Bestimmung der Gewichtung der Elemente untereinander.

Eine Methode, deren Anliegen es ist, einen Gewichtsvektor zur Priorisierung zu erstellen und somit auch eine Gewichtung der Entscheidungsfaktoren a priori zu bestimmen ist der Analytic Hierarchy Process bzw. Analytische Hierarchieprozess (AHP). Die AHP-Methode basiert im Gegensatz zur CV-Methode nicht auf der Verteilung eines immateriellen Gutes, sondern auf paarweisen Vergleichen der zu priorisierenden Elemente, ohne Dopplungen. Weitere Vorteile dieser Methode sind, dass sie ursprünglich aus dem Bereich der Management- und Entscheidungstheorie stammt und somit auch außerhalb der IT verbreitet und bekannt ist. Außerdem ist es möglich, eine rudimentäre Konsistenzprüfung der Antworten durchzuführen, auch ohne Einbeziehung von doppelten oder getauschten Fragen. Der Nachteil dieser Methode ist jedoch, dass je nach Setting und Anzahl der zu priorisierenden Elemente viele Vergleiche durchgeführt werden müssen.

Bei den jeweiligen Vergleichen wird im Regelfall eine Skala von 1-9 verwendet bzw. die zugehörigen Reziproke. Tabelle 6 erläutert die Verwendung der unterschiedlichen Skalenwerte.

Aus den Ergebnissen der paarweisen Vergleiche lässt sich eine Matrix je Befragtem ableiten, außerdem sind die Ergebnisse aggregierbar (vgl. Saaty 1994). Für die Durchführung der Studie wurde zunächst eine einfache Web-Anwendung implementiert, die sowohl den Aufnahmeprozess als auch den Auswertungsprozess unterstützt. Abbildung 20 zeigt ein Bildschirmabbild der Oberfläche; wie zu ersehen ist, erfolgt die Eingabe mittels Schiebereglern. Auf der rechten Seite befindet sich nochmals eine kurze Erklärung der Skalenwerte, analog zu Tabelle 6, sowie auch eine Übersicht über alle 21 Vergleiche. Somit hat der Anwender die Möglichkeit nach Abschluss seine Eingabe übersichtlich zu prüfen und ggf. Korrekturen vorzunehmen.

Tabelle 6: Definition der Skalenwerte im Analytischen Hierarchieprozess

Skalenwert	Definition	Erklärung
1	Gleiche Bedeutung	Beide Eigenschaften haben die gleiche Bedeutung bzgl. des Ziels
3	Moderater Unterschied	Aus Erfahrung ist bekannt, dass ein moderater Unterschied besteht
5	Starker Unterschied	Aus Erfahrung ist bekannt, dass ein starker Bedeutungsunterschied besteht
7	Sehr starker Unterschied	Eine Eigenschaft wird gegenüber der anderen sehr stark favorisiert; der Bedeutungsunterschied ist praktisch demonstrierbar
9	Extremer Unterschied	Die Beweise zugunsten einer Eigenschaft gegenüber einer anderen sind von höchstmöglicher Ordnung der Affirmation
2, 4, 6, 8	Zwischenwerte	
Reziprok der genannten Skalenwerte	Wenn i einem der oben genannten Skalenwerte zugeordnet wird beim Vergleich mit j, dann erhält j den reziproken Wert, wenn j mit i verglichen wird.	

Anmerkung: Eigene Darstellung in Anlehnung an (Saaty 2008)

Die Durchführung der Befragung wäre auch analog möglich gewesen, jedoch hätte sich dadurch ein erheblicher Mehraufwand für die Zusammenführung und anschließende Auswertung der Daten ergeben, da insgesamt 21 paarweise Vergleiche durchgeführt werden mussten. Außerdem wäre es durch die Implementierung möglich gewesen, die Befragung online durchzuführen.

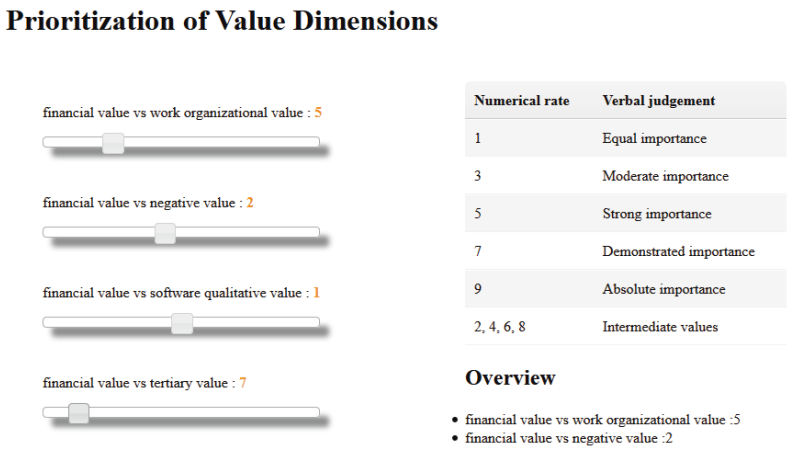


Abbildung 20: Oberflächendesign der Web-Anwendung zur Unterstützung von AHP

Die Durchführung der Studie als Onlinebefragung wurde verworfen, da durch persönliche Treffen sichergestellt werden konnte, dass alle Beteiligten dieser empirischen Erhebung ein möglichst identisches Verständnis von den unterschiedlichen Nutzwertdimensionen erhalten. Dieses wurde durch eine ca. 5-minütige Einleitung gewährleistet, in der sowohl die Web-Oberfläche vorgestellt wurde, als auch AHP und die Definition der Nutzwertdimensionen.

Bei der Durchführung mit den ersten Teilnehmern zeigte sich, dass es auch während der Durchführung noch zu Rückfragen bzgl. der Benutzung und der genaueren Definition der unterschiedlichen Nutzwertdimensionen kam. Deswegen wurde entschieden, die Bewertung jeweils im Rahmen eines ca. 30-minütigen Termins durchzuführen.

Insgesamt wurde die Studie mit 32 Teilnehmern durchgeführt. Dabei wurden 15 aktive Product Owner bzw. Projektleiter aus dem Umfeld von Managementinformationssystemen bei Volkswagen befragt. Die weiteren 17 Teilnehmer sind Key User unterschiedlicher Managementinformationssysteme aus unterschiedlichen hierarchischen Ebenen. Die Spanne erstreckte sich hierbei von der Ebene der Sachbearbeiter bis hin zu Mitgliedern des höheren Managements. Somit konnte ein breites Erfahrungswissen mit in die Auswertung einfließen. Eine Auswertung nach einzelnen Hierarchieebenen konnte jedoch aufgrund von bestehenden Datenschutzregelungen des Konzerns nicht erfolgen.

In Tabelle 7 werden die aggregierten Gewichtsvektoren für die Key User, die Product Owner sowie der aggregierte Gewichtsvektor über alle Teilnehmer dargestellt. Der Gewichtsvektor ist der Eigenvektor des maximalen Eigenwertes (vgl. Kwong und Bai 2002, Saaty 2003).

Tabelle 7: Berechnete Gewichtsvektoren der Nutzwertdimensionen, sortiert nach der relativen Bedeutung

Nutzwertdimension	Key User	Product Owner	Total
negativer Nutzwert	0,255	0,2	0,229
arbeitsorganisatorischer Nutzwert	0,164	0,183	0,173
strategischer Nutzwert (Kunde)	0,166	0,142	0,155
finanzieller Nutzwert	0,149	0,146	0,148
softwarequalitativer Nutzwert	0,118	0,104	0,111
strategischer Nutzwert (IT)	0,092	0,127	0,109
tertiärer Nutzwert	0,056	0,098	0,076

Anmerkung: (N = 32)

Wie Tabelle 7 zu entnehmen ist, ergeben sich zwischen den beiden unterschiedlichen Gruppen nur kleinere Unterschiede. Dies ist ein Indikator dafür, dass beide Gruppen ein gemeinsames Verständnis bzgl. des Nutzwertes von Anforderungen bzw. User Stories besitzen. Auffällig ist, dass der negative Nutzwert bei Nichtumsetzung bei dieser Erhebung von höchster Bedeutung ist, obwohl dieser bei der Bestimmung der relevanten Nutzwertdimensionen nicht unter den am häufigsten genannten war (vgl. Tabelle 5).

Als ein Ergebnis dieser Untersuchung ergibt sich die Möglichkeit, den ganzheitlich betrachteten Nutzwert einer User Story im Kontext von Managementinformationssystemen zu bestimmen. Hierzu muss lediglich die Bewertung je Nutzwertdimension (n_i) mit dem korrespondierenden Gewichtungsfaktor (w_i) multipliziert werden. Anschließend sind die Teilergebnisse zu summieren. Dieser Sachverhalt wird in der nachfolgenden Formel dargestellt:

$$\sum_{i \in W} n_i * w_i$$

Formel 1: Nutzwertberechnung auf Grundlage des bestimmten Gewichtsvektors

Die zu erwartenden Nutzwerte je Nutzwertdimension können bspw. durch Nutzwertpunkte, analog zu Story-Punkten für die erste Aufwandsschätzung, bestimmt werden. Der Vorteil dieser Methode ist, dass im Vergleich zu einer monetären Bewertung die Kommunikation und mögliche Diskussionen vereinfacht werden, da im ersten Schritt kein Bezug zur absoluten monetären Bewertung besteht. Außerdem sind einige Nutzeneffekte nicht unmittelbar monetär bewertbar, da sie nur einen mittelbaren Nutzen aufweisen.

Durch die Verwendung der Ergebnisse dieser Studie ergibt sich die Möglichkeit, jeder User Story einen, für alle Prozessbeteiligten transparent ermittelten, Nutzwert zuzuweisen. Die Tabelle 8 und die Tabelle 9 zeigen ein fiktives Beispiel, das aus zwei unterschiedlichen User Stories besteht. Jeder User Story werden zunächst die Nutzwertschätzungen für jede Nutzwertdi-

mension hinzugefügt. Anschließend werden die Schätzungen mit dem in dieser Studie bestimmten Gewichtsvektor multipliziert, um den jeweiligen Nutzwert zu erhalten. Bei diesem Beispiel wird davon ausgegangen, dass die Nutzenpunktverteilung analog zur Storypunktevergabe auf Basis der Fibonacci-Folge erfolgt.

Wie Tabelle 8 und Tabelle 9 zeigen, weisen die beiden fiktiven User Stories die identische Anzahl an Nutzenpunkten (12 Nutzenpunkte) auf. Wird jedoch der zuvor bestimmte Gewichtsvektor auf die Nutzwertschätzung angewendet, zeigt dieses Beispiel einen deutlichen Unterschied für die Nutzwertbetrachtung. Es ergibt sich ein relativer Unterschied von ungefähr 60 %. Für eine daraus ableitbare Priorisierung unter Einbeziehung der vermeintlichen Aufwände für eine Umsetzung würden sich je nach geschätztem Aufwand auch deutliche Unterschiede feststellen lassen.

Tabelle 8: Erstes fiktives Beispiel für die Nutzwertberechnung einer User Story

Nutzwertdimension	Nutzenpunkte	Gewichts-vektor	Nutzenwert * Gewichtsvektor
finanzieller Nutzwert	*0*	*0.148*	*0.0*
arbeitsorganisatorischer Nutzwert	*3*	*0.173*	*0.519*
negativer Nutzwert bei Nichtumsetzung	*5*	*0.229*	*1.145*
softwarequalitativer Nutzwert	*0*	*0.111*	*0.0*
tertiärer Nutzwert	*1*	*0.076*	*0.076*
strategischer Nutzwert (Kunde)	*1*	*0.155*	*0.155*
strategischer Nutzwert (IT)	*2*	*0.109*	*0.218*
Summe	***12***	***1***	***2.113***

Tabelle 9: Zweites fiktives Beispiel für die Nutzwertberechnung einer User Story

Nutzwertdimension	Nutzenpunkte	Gewichts-vektor	Nutzenwert * Gewichtsvektor
finanzieller Nutzwert	5	0.148	0.74
arbeitsorganisatorischer Nutz-wert	0	0.173	0.0
negativer Nutzwert bei Nich-tumsetzung	0	0.229	0.0
softwarequalitativer Nutzwert	2	0.111	0.222
tertiärer Nutzwert	5	0.076	0.38
strategischer Nutzwert (Kunde)	0	0.155	0.0
strategischer Nutzwert (IT)	0	0.109	0.0
Summe	**12**	**1**	**1,342**

Da diese hier verwendeten Daten durch zwei empirische Studien erhoben wurden, ergeben sich Einschränkungen bzgl. der Validität und der Allgemeingültigkeit der Ergebnisse. Aufgrund der starken Fokussierung auf den Bereich der Managementinformationssysteme bei einem Original Equipment Manufacturer (OEM) in der Automobilindustrie, ist die Zahl der Teilnehmer wie auch die der potenziellen Teilnehmer stark eingeschränkt. Bei einem ähnlichen Vorgehen in anderen Domänen, wie der Shopfloor-IT, in der auch produktionskritische Softwaresysteme im Einsatz sind, hätte dieses hier angewendete Vorgehen unter Umständen andere relevante Nutzwertdimensionen identifiziert. Auch die Durchführung zur Gewichtung der relevanten Nutzwertdimensionen hätte vermutlich zu anderen Ergebnissen geführt, da hier ein wesentlich höheres Augenmerk auf der Sicherheit, der Stabilität und der Ausfallsicherheit liegt. Deswegen muss festgestellt werden, dass das Verfahren grundsätzlich universell einsetzbar ist, die Ergebnisse jedoch nicht auf andere Bereiche oder andere Organisationen direkt übertragbar sind.

6.5 Erweiterung der Nutzwertdefinition um weitere Rahmenparameter

In diesem Unterkapitel wird beschrieben, welche weiteren Rahmenparameter bei der Betrachtung des Nutzwertes einer User Story mit berücksichtigt werden müssen und wie unter deren Einbeziehung eine Priorisierung, bzw. eine Ordnung der User Stories innerhalb des Product-Backlogs erfolgen kann.

Wie in Kapitel 6.4 beschrieben, erfolgt die Abschätzung des zu erwartenden Nutzwertes durch eine Punktevergabe, analog zur Vergabe von Storypunkten. Hierbei sind die zu erwartenden

Nutzeneffekte je Nutzwertdimension zu bewerten und die Abstimmung zwischen dem Anforderer und dem Product Owner zu plausibilisieren. Eine Besonderheit stellt hierbei die Nutzwertdimension des strategischen Nutzwerts aus Sicht der IT dar. Diese kann im Regelfall nicht vom Anforderer bestimmt und abgeschätzt werden. Die Bewertung liegt in der Verantwortung des Product Owner in Zusammenarbeit mit weiteren Projektbeteiligten auf Seiten der IT.

Um ausgehend von der beschriebenen Nutzwertabschätzung hin zu einer Priorisierung zu kommen, müssen jedoch weitere Rahmenbedingungen mit in die Betrachtung eingeschlossen werden. Zum einen müssen fixe Abgabefristen bei der Erstellung der Priorisierung mit betrachtet werden. Zum anderen wird bei dem beschriebenen Verfahren zur Nutzwertabschätzung noch Bezug auf die Verbreitung der Anforderungen genommen. Dies bedeutet, dass im Kontext dieser Untersuchung zumeist ein Anforderer eines bestimmten Fachbereichs federführend für eine explizite Anforderung verantwortlich ist. Weitere mögliche Profiteure werden noch nicht mit in die Nutzwertabschätzung mit einbezogen. Es ist also zu unterscheiden, ob der Nutzen, der durch die Umsetzung der Anforderung entsteht, nur exklusiv einem Fachbereich oder sogar nur einem Anforderer zugeordnet werden kann oder ob bspw. durch einen neuen konzernweiten Standardbericht viele Anwender und Berichtskonsumenten von der Umsetzung dieser Anforderung profitieren.

User Stories, welche einen festen Umsetzungstermin haben, sollten zunächst separat und unabhängig von den ihnen beigemessenen Nutzwerten betrachtet werden. An dieser Stelle bietet sich eine Kombination mit der MoSCoW-Priorisierung an (s. Kapitel 3). Anforderungen, deren Umsetzungsfrist in unmittelbare Nähe rückt, sodass eine weitere terminliche Verschiebung zur Nichteinhaltung dieser Frist führen würde, müssen der Gruppe der Must-have-Anforderungen zugeordnet werden. Diese Anforderungen sollten ebenfalls, unabhängig vom Nutzwert, die höchste Priorität innerhalb dieser Gruppe erhalten, um deren Abarbeitung sicherzustellen. Alle weiteren Must-have-User Stories würden dementsprechend sortiert nach dem Verhältnis von Nutzwert in Nutzenpunkten zu erster Aufwands- und Komplexitätsschätzung in Storypunkten. Die User Stories in den beiden Gruppen „Should-have“ und „Could-have“ werden ausschließlich nach ihrem jeweiligen Verhältnis von Aufwand zu Nutzen sortiert. Anforderungen aus der Gruppe „Won’t-have“ besitzen für die Iterationsplanung keine weitere Relevanz und müssen deswegen nicht sortiert werden.

Durch die Kombination von Nutzwertbestimmung und der Verwendung von Prioritätengruppen ist der Product Owner weiterhin in der Lage, das Projekt steuernd zu leiten, Transparenz bei Priorisierung der Anforderungen herzustellen und sicherzustellen, dass Anforderungen mit fixen Fristen eingehalten werden, sofern möglich. Je nach Projektsituation und Anzahl der Anforderungen kann es auch ausreichend sein, die Anforderungen in drei anstelle von vier Gruppen zu unterteilen. Es würde dann die Gruppe der Muss-, der Kann- und der nicht relevanten Anforderungen geben. Dieses ist anwendbar, da durch die Kombination der beiden Verfahren eine explizite Priorisierung der User Stories je Gruppe möglich ist.

Soll die Nutzwertdefinition um die Betrachtung der mit der Umsetzung verbundenen Profiteure erweitert werden, müsste die ungefähre Anzahl derer, die einen Nutzen durch die Umsetzung erfahren, von den Projektverantwortlichen bestimmt oder abgeschätzt werden und in die finale Berechnung des Nutzwertes mit einfließen. Da die Nutzwertdefinition weiterhin transparent und leicht nachvollziehbar erfolgen soll, wurde im Rahmen eines Expertenkomitees entschieden, dass die zuvor erhobenen Nutzwerte mit einem jeweiligen „Verbreitungsfaktor“ multipliziert werden und somit Anforderungen, von denen mehr organisatorische Einheiten bzw. Nut-

zer und Konsumenten[5] profitieren, ein sogenanntes „boosting“ erfahren. Im Rahmen dieses Komitees wurden die in Tabelle 10 dargestellten Faktoren mit ihrer jeweiligen Bedeutung festgelegt.

Tabelle 10: Faktoren für das "Boosting" des Nutzwertes von Anforderungen in Abhängigkeit der Profiteure

Faktor	Definition
1	Profiteur ist ein einzelner Fachbereich
2	Profiteure sind unterschiedliche Fachbereiche eines Werkes
3	Profiteure sind unterschiedliche Fachbereiche unterschiedlicher Werke
10	Profiteure sind Fachbereiche aller an das System angeschlossenen Werke

Zusammenfassend kann die erste Forschungsfrage bzgl. der Definition des Nutzwertes aus Kapitel 2.3 wie folgt beantwortet werden: Der Nutzwert von IT-Anforderungen ist im ersten Schritt nicht eindimensional zu erfassen. Zunächst müssen die unterschiedlichen Dimensionen, in denen Nutzeneffekte auftreten können, definiert und betrachtet werden. Erst im folgenden Schritt, durch die Erstellung und Anwendung eines Gewichtsvektors, können diese dann zu einem absoluten Wert abgebildet werden. Anschließend kann der geschätzte Nutzwert um weitere Faktoren, wie die Anzahl der Profiteure, erweitert werden.

Im letzten Schritt ist es notwendig, nicht über Nutzwerte quantifizierbare Eigenschaften, wie fixe terminliche Fristen und Aufwände für die Umsetzung, mit in die Betrachtung einfließen zu lassen. Dadurch können die identifizierten und bewerteten Nutzeneffekte dazu verwendet werden, eine für alle Prozessbeteiligten transparente und plausible Priorisierung der Anforderungen vorzunehmen, ohne explizit auf das Erfahrungswissen eines Projektleiters oder Product Owners innerhalb der IT angewiesen zu sein.

Durch diese mit den Prozessbeteiligten abgestimmte Identifikation und Bewertung des Nutzwertes von Anforderungen ist für den weiteren Verlauf dieser Arbeit eine Grundlage für die spätere Akzeptanz des ganzheitlichen Lösungsansatzes geschaffen worden. Die Akzeptanz dieses Ansatzes wird dabei vor allem durch den hier verwendeten partizipativen Ansatz erreicht. Dadurch konnten viele Prozessbeteiligte unmittelbar in den Entstehungsprozess mit eingebunden werden und aktiv am Modell zur Abschätzung des Nutzwerts von IT-Anforderungen im Umfeld von Managementinformationssystemen bei Volkswagen direkt mitwirken. Die partizipative Herangehensweise hat auch dazu geführt, dass sich diese Methodik innerhalb des Prozesses etablieren konnte.

[5] Konsumenten sind in diesem Kontext Personen, die zuvor durch Nutzer erstellte Auswertungen und Berichte verwenden, ohne einen direkten Kontakt mit dem Softwaresystem zu haben.

7 Systematische Nutzung und Abbildung von Synergieeffekten in Scrum

Die Automobilindustrie wird geprägt durch den gesellschaftlichen Wandel. Dies stellt zum einen die Produkte selbst, zum anderen die dazugehörigen Prozesse vor immer neue Herausforderungen (Rudow und Heidecke 2014; Mühleck, K. H. & Heidecke, H.-C. 2012). Für die IT bedeutet dies, zum einen die Qualität der Systeme zu verbessern, zum anderen die Kosten für die Entwicklung und für die Aufrechterhaltung zu reduzieren, um einen Wettbewerbsvorteil zu erlangen. Es ist somit nötig Prozesse zu vereinigen und zu standardisieren.

Die Herausforderung an die IT ist hierbei das Einnehmen von verschiedenen Rollen. Zum einen übernehmen sie die Rolle des Dienstleisters, der IT-Lösungen global bereitstellt, um den einzelnen Fachbereichen ihr operatives Tagesgeschäft zu ermöglichen. Zum anderen sollen technische Innovationen zur Unterstützung der Prozesse geschaffen werden, um wettbewerbsfähig zu bleiben bzw. einen Wettbewerbsvorteil zu erhalten. Der dritte Punkt ist als der schwierigste anzusehen. Die IT übernimmt hierbei eine Art Ordnerfunktion, welche ermöglichen soll, IT-Standards zu schaffen und einzuhalten. In großen Konzernen, wie bspw. bei der Volkswagen AG, sollen markenübergreifende Standards definiert und eingehalten werden, um eine Harmonisierung zu schaffen. Es leiten sich drei grundsätzliche Ziele ab: Effektivität durch Optimierung, Effizienz durch Standardisierung und Kulturzusammenführung durch kollaboratives Arbeiten.

Hierbei ergibt sich die folgende Schwierigkeit: Bei der Entwicklung von den einzelnen Systemen muss zum einem darauf geachtet werden, die unterschiedlichen Kundenwünsche aus unterschiedlichen Standorten an das System zu erfüllen, zum anderen müssen die Entscheidungen und strategischen Vorgaben des Managements berücksichtigt werden. Aufgrund dieser Aspekte ist es wichtig einen transparenten Priorisierungsprozess der einzelnen Anforderungen und der sich daraus ergebenden Reihenfolge der Umsetzung zu schaffen. Die einzelnen Kunden fordern zumeist, dass ihre Anforderungen schnellstmöglich umgesetzt werden, was dazu führen kann, dass Ressourcen nicht mehr optimal ausgeschöpft werden können. Dieses kann zu Situationen führen, in denen unterschiedliche Kunden eine gewisse Erwartungshaltung gegenüber der IT haben und die IT zwischen diesen hin- und hergerissen wird, da sie aufgrund beschränkt vorhandener Ressourcen nicht alle Anforderungen zeitgleich umsetzen kann (Wnuk et al. 2011).

Agile Prozesse, wie bspw. Scrum, sollen dabei helfen, diese ständig wachsenden und sich volatil verhaltenden Anforderungen effizienter zu bewältigen (Coram und Bohner 2005). Agile Entwicklungsmethoden stellen den Kunden in den Mittelpunkt der Entwicklung. Laut einer Studie der Hochschule Koblenz mit 600 Teilnehmern aus 30 Ländern aus dem Jahr 2014, setzen ca. 60 % aller Befragten agile Vorgehensmodelle für ihr Projektmanagement ein (Komus 2014). Das agile Vorgehen Scrum ist dabei von besonders hoher praktischer Relevanz und laut dieser Studie sogar die agile Methode mit der höchsten praktischen Bedeutung.

Bei großen Softwaresystemen, wie bspw. Managementinformationssysteme in großen Konzernen, kommt es dazu, dass mehrere Entwicklerteams an einem einzelnen Produkt simultan arbeiten müssen, um die Wünsche des Kunden innerhalb der geforderten Zeit und Qualität umzusetzen. Die Kunden kommen hierbei aus unterschiedlichen Fachbereichen und Standorten. In der internen IT müssen Softwaresysteme entwickelt und verbessert werden, welche alle Anforderungen der einzelnen Fachbereiche erfüllen. Dieser Faktor beeinflusst den Prozess des Requirements Engineerings maßgeblich. Eine einfache Anforderungsanalyse mit Priorisierung ist hier nicht mehr ausreichend, es wird vielmehr eine Methode zur Bündelung gleichartiger

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0_7

Anforderungen benötigt. Das klassische Scrum-Vorgehensmodell repräsentiert seine Anforderungen durch User-Stories, welche im Product Backlog beschrieben und priorisiert werden und eine Nutzwert- und Aufwandabschätzung erhalten. Für den Kunden und die Stakeholder entsteht hierdurch zwar ein transparenter Entwicklungsprozess, für die Softwareentwickler und IT-Abteilungen bleiben jedoch Synergie- und Einsparpotenziale ungenutzt.

Die Idee, welche in diesem Kapitel genauer erläutert und beschrieben wird, ist, gleichartige Anforderungen auf Projektebene zu bündeln, um Doppelentwicklungen und nachträgliche Anpassungen von Beginn an zu vermeiden. Ein Beispiel hierfür wären zwei Anforderungen aus unterschiedlichen Fachbereichen oder Standorten, die jeweils eine eigens gestaltete Auswertung einer ähnlichen oder identischen Kennzahl fordern. Beide Auswertungen würden aber bspw. Daten aus einem identischen Vorsystem benötigen, zu dem eine neue Schnittstelle aufgebaut werden müsste. Zur besseren Veranschaulichung dieses Problems werden an dieser Stelle die fiktiven User Stories aus Kapitel 2.3 aufgegriffen:

User Story 1: Als Produktionsleiter des Werkes H möchte ich die zeitliche Entwicklung der ökologischen Kennzahl Ö je Fertigungslinie auswerten können, um meine Produktion nachhaltiger gestalten und die Wirkung von Maßnahmen besser beurteilen zu können.

User Story 2: Als Werkleiter möchte ich, dass die Kennzahl Ö mit in den Standardbericht X aufgenommen wird, um über die aktuelle Zielerreichung dieser Kennzahl in den unterschiedlichen Gewerken informiert zu sein.

Wenn diese beiden Anforderungen, welche beispielhaft als User Stories dokumentiert sind, getrennt voneinander betrachtet werden würden, könnte eine Schnittstelle implementiert werden, die nur die benötigten Daten für eine der beiden Anforderungen beinhaltet und nicht für beide. Dieses würde dazu führen, dass entweder eine zweite Schnittstelle aufgebaut werden müsste, um die Daten für die jeweils andere Auswertung zu erhalten oder die ursprüngliche Schnittstelle müsste erneut angepasst werden. Dieses bedeutet sowohl auf der Seite des Berichtssystems, als auch auf der Seite des Datenlieferanten, zusätzliche Aufwände entstehen, die mit zusätzlichen Kosten in beiden Entwicklungsteams verbunden sind.

Die zusätzlichen Aufwände könnten jedoch bei einer gemeinschaftlichen Betrachtung dieser beiden fiktiven Anforderungen vermieden werden. Dabei stellt die Bündelung von Anforderungen kein grundsätzlich neues Konzept dar, hierbei handelt es sich vielmehr um einen Vorgang, den viele Product Owner und Projektleiter implizit durchführen. Jedoch mangelt es noch an einer systematischen Verankerung und der Möglichkeit der Abbildung dieser Tätigkeiten innerhalb des Scrum-Vorgehensmodells.

Zu diesem Zweck ist dieses Kapitel in mehrere Unterkapitel unterteilt. Zunächst wird erläutert, wie zurzeit Anforderungen und Form von User Stories dokumentiert werden. Anschließend wird der Begriff „Synergiepotenzial zwischen Anforderungen“ erläutert definiert und abgegrenzt. Darauf aufbauend wird die Einführung eines neuen Produktrückstandselementes motiviert und es wird aufgezeigt, wie dieses in die bestehende Scrum-Systematik eingebunden werden kann. Ausgehend davon werden die Implikationen für den Prozess an sich, als auch die Auswirkungen auf die Tätigkeitsbeschreibungen und Verantwortungen der traditionellen Rollen in Scrum diskutiert.

7.1 Traditionelle Dokumentation von Anforderungen mit User Stories

User Stories sind ein etabliertes und weit verbreitetes Konzept für die Beschreibung und das Management von Anforderungen in agilen Softwareprojekten, die bspw. Scrum als agile Projektmanagementmethode einsetzen. Im Vergleich zu traditionellen Mitteln des Anforderungsmanagements sind User Stories weniger detailliert formuliert, um die Kommunikation mit dem Kunden und den Stakeholdern zu fördern. Dieses dient dem Ziel, Anforderungen genauer zu hinterfragen und durch die direkte Kommunikation „bessere" Anforderungen zu formulieren, die für den tatsächlichen Anwendungsfall des Kunden dienlicher sind.

User Stories beschreiben hierbei Anforderungen aus der Sicht des Kunden bzw. des späteren Nutzers (User). User Stories werden zumeist auf kleineren Karten und mit einem bis wenigen Sätzen beschrieben. Die beschriebene Karte dient hierbei der Dokumentation, sie ersetzt aber keinesfalls eine weitere Anforderungsanalyse und eine genauere Spezifikation dieser Anforderung. Deswegen besteht nach Wirdemann (2011) eine User Story immer aus mindestens drei Aspekten. Zum einen gibt es eine Story-Karte, die eine Anforderung mittels eines oder weniger Sätze dokumentiert und sie „verwaltbar" macht. Zum anderen ist eine direkte Konversation zwischen Entwickler und Product Owner bzw. Product Owner und Anforderer unerlässlich, um die Anforderung, welche in Form einer Story-Karte dokumentiert ist, genauer zu spezifizieren und den eigentlichen Nutzen für den Kunden und den unterstützten Prozess zu verstehen. Story-Karten können dabei z. B. mittels standardisierter Formen erstellt werden. Dies hat zum Ziel, dass die wichtigsten Informationen von allen Prozessbeteiligten schnell erfasst werden können und dass alle notwendigen Informationen vorhanden sind. Ein mögliches Template ist in Abbildung 21 dargestellt.

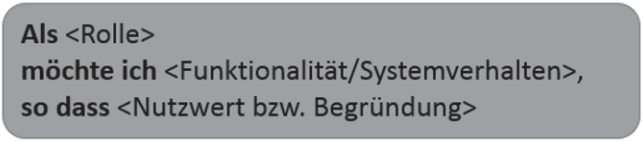


Abbildung 21: Standardisierte Form einer User Story, Quelle: eigene Abbildung in Anlehnung an (Rupp und SOPHISTen 2014)

Die dritte Komponente einer User Story sind Akzeptanzkriterien. Diese legen fest, unter welchen Bedingungen ein Produktrückstandselement als vollständig implementiert gilt und unter welchen Bedingungen es vom Anforderer abgenommen wird.

Im Zusammenhang mit User Stories findet sich in der Literatur auch der Begriff Epen (engl. epics) (Pichler 2010). Hierbei handelt es sich um grobe, noch nicht ausreichend definierte und fokussierte User Stories. Im Laufe der Anforderungsanalyse wird dann ein Epic weiter spezifiziert, sodass sich eine oder mehrere User Stories aus diesem Epic ergeben. Diese werden um weitere Attribute angereichert. Hierzu gehören bspw. die Aufwandsschätzung oder die zugehörigen Akzeptanzkriterien. User Stories werden vor Umsetzung in Tasks untergliedert. Dieses geschieht z. B. nach dem Sprint Planning Meeting (s. Kapitel 3.3.2). Dieser Zusammenhang wird in Abbildung 22 graphisch dargestellt.

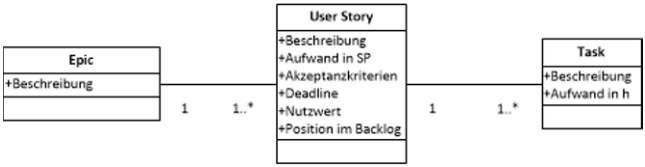


Abbildung 22: Zusammenhang zwischen Epic, User Story & Task (Quelle: Eigene Darstellung)

Ein wichtiges Scrum-Prinzip ist, dass nur fertige, das heißt vollständig abgeschlossene User Stories, einen Wert haben (Stober und Hansmann 2010). Die Akzeptanzkriterien einer User Story legen fest, wann diese abgeschlossen bzw. implementiert ist, so dass diese einen Mehrwert für den Kunden liefert. Die Akzeptanzkriterien werden in der Regel vom Product Owner in Zusammenarbeit mit dem Anforderungssteller erarbeitet und anschließend, falls nötig, für das Entwicklerteam zum besseren Verständnis und zur besseren Nachvollziehbarkeit verfeinert. Aus Sicht des Entwicklerteams ist eine Anforderung also als abgeschlossen anzusehen, wenn sie die allgemeine „Definition von Fertig“ („Definition of Done“) und die spezifischen Akzeptanztests aus Sicht des Kunden erfüllt. Im Rahmen des Sprint Review Meetings werden, wie in Kapitel 3.3.3 erläutert, die Anforderungen dem Product Owner und

Anforderern vorgestellt. Hierbei werden im Allgemeinen auch die Akzeptanzkriterien durch die Anforderer geprüft.

Ausgehend von dieser grundsätzlichen Beschreibung der Verwendung von User Stories wird im Folgenden darauf eingegangen, wie Anforderungen unterschiedlicher Prozessbeteiligter, die Synergiepotenzial beinhalten, innerhalb von Scrum systematisch erfasst und dokumentiert werden können, ohne dass die vorherrschenden Scrum-Prinzipien dabei verletzt werden.

7.2 Synergiepotenziale zwischen User Stories

Dieser Abschnitt befasst sich mit der Definition des Terms „Synergiepotenziale zwischen Anforderungen“ und der Prüfung, inwieweit sich diese mit den bestehenden Artefakten und Rollendefinitionen von Scrum vereinbaren lassen.

Mit „Synergiepotenziale zwischen Anforderungen“ ist nicht das Konzept „Das Ganze ist mehr als die Summe seiner Teile.“ (Aristoteles) gemeint. Es bedeutet somit nicht, dass ein Softwaresystem, welches über mehrere unterschiedliche Funktionen verfügt, als Einheit betrachtet, für den Anwender einen höheren Nutzwert aufweist, als die Summe der Nutzwerte der für sich separat betrachteten Funktionen dieses Softwaresystems.

Im Zusammenhang dieser Arbeit ist mit „Synergiepotenzialen zwischen Anforderungen“ auch nicht die Betrachtungsebene der Enterprise-Architektur (EA) und des Enterprise-Architektur-Managements (EAM) gemeint, die das Ziel verfolgen zu einer konsolidierten IT-Landschaft innerhalb eines Unternehmen oder Konzerns zu gelangen (Barkow 2010). Auf dieser Ebene werden Synergiepotenziale nicht auf Ebene der einzelnen Anforderungen und deren Umsetzung betrachtet. EAM betrachtet Synergien zwischen Softwaresystemen und erarbeitet Strategien für eine zielgerichtete Weiterentwicklung von Systemen. Außerdem wirkt EAM steuernd darauf ein, in welchem System eine Anforderung umgesetzt wird (Bente et al. 2012).

Unter dem Term „Synergiepotenziale zwischen Anforderungen“ ist im Zusammenhang dieser Arbeit das vorhandene Einsparpotenzial bei der Umsetzung von IT-Anforderungen gemeint, welches sich ergibt, wenn nicht jede Anforderung für sich separat betrachtet wird. Durch eine gemeinschaftliche Betrachtung und die Ausnutzung von Überschneidungen auf Projektebene bei der gemeinschaftlichen Betrachtung von Anforderungen soll die Umsetzung der Anforderungen effizienter gestaltet werden. Das Synergiepotenzial zwischen zwei Anforderungen entspricht somit den zu erwartenden Einsparungen bei einer gemeinschaftlichen Umsetzung gegenüber einer separaten Umsetzung je Anforderung. Diese Einsparungen können zum einen durch verringerte Zeiten für die reine Implementierung innerhalb des Projektteams entstehen, zum anderen auch durch einen geringen Umfang an unterstützenden Tätigkeiten in anderen Projektteams, welche z. B. für die Bereitstellung einer Schnittstelle eines Fremdsystems verantwortlich sind.

Bei einer gemeinschaftlichen Betrachtung der beiden fiktiven User Stories aus Kapitel 1 ergibt sich, wie bereits erwähnt, ein Synergiepotenzial im Bereich der Schnittstelle zu einem Fremdsystem. Wird dieses Potenzial bspw. vom Product Owner oder vom Entwicklerteam erkannt, muss diese Erkenntnis auch dokumentiert werden, sodass diese Information auch in späteren Iterationen noch zur Verfügung steht und das Potenzial nicht „verloren“ geht.

Um dieses Potenzial zwischen zwei, in Form von User Stories dokumentierten und verwalteten, Anforderungen auszudrücken, ist eine Art von Zusammenfassung bzw. Bündelung oder auch „Restrukturierung“ dieser beiden Anforderungen notwendig. Hierbei ergibt sich die Herausforderung, dieses mit User Stories und Tasks abzubilden.

7.2.1 Vereinigung von User Stories zur Abbildung von Synergien

Eine simple Möglichkeit wäre eine neue User Story zu formulieren, die die Anforderungen der User Stories umfasst, welche ein gemeinsames Synergiepotenzial aufweisen. Dieser Sachverhalt ist in Abbildung 23 graphisch veranschaulicht. Die Probleme, die sich bei dieser Methode ergeben, sind, dass die neue User Story nicht mehr einem Anforderer zugeordnet werden kann und der Aufwand der neuen User Story bei dieser Art der Bündelung schnell einen innerhalb eines Sprint beherrschbaren Aufwand überschreitet. Außerdem können, je nach Anforderungsinhalten, die Anforderungen selbst thematisch wenig zusammenhängen, sodass es sich bei dieser neu gestalteten User Story nicht mehr um eine User Story im eigentlich Sinn handeln würde. Dies liegt daran, dass es sich bei dieser abgeleiteten User Story nicht mehr um einen zusammenhängenden Bereich handeln muss, der einen Nutzwert für den Anforderer erzeugt, sondern der auch mehrere Bereiche umfassen kann, die mehrere unterschiedliche Nutzeneigenschaften vereinen. In der klassischen Betrachtungsweise müsste diese neue User Story dann wieder in einzelne User Stories unterteilt, bzw. Anforderungen weiter verfeinert werden. Somit wäre an dieser Stelle nicht ersichtlich, ob es sich um eine noch nicht fertig spezifizierte User Story handelt oder um die Zusammenlegung mehrerer User Stories.

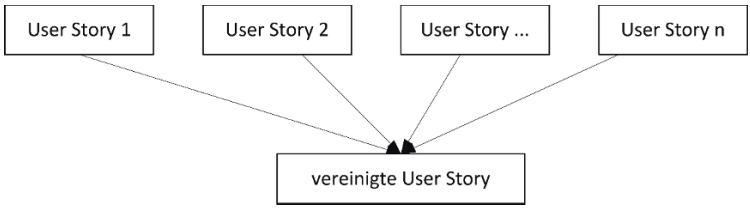


Abbildung 23: Vereinigung von User Stories (eigene Abbildung)

7.2.2 Umverteilung von Arbeitsumfängen auf neue User Stories

Nachdem aufgezeigt wurde, dass eine einfache Zusammenfassung mehrerer User Stories zu einer neuen User Story in vielen Fällen nicht praktikabel handhabbar ist, ist zu untersuchen, ob auf Ebene der User Stories eine Möglichkeit besteht die bestehenden Anforderungen aus unterschiedlichen User Stories auf mehrere neue User Stories zu verteilen. Dieses wird schematisch in Abbildung 24 dargestellt. An dieser Stelle soll also keine Detaillierung oder reine Vereinigung von Anforderungen durchgeführt werden, sondern lediglich eine Umverteilung der bereits identifizierten Arbeitsumfänge. Durch diese Umverteilung sollen Arbeitsumfänge zusammengefasst werden, welche Synergiepotenziale beinhalten. Dabei werden andere Aufwendungen in neue User Stories ausgelagert, sodass es zu keinen Überschneidungen kommt. Der Vorteil dieser Methode ist, dass die so neu erzeugten User Stories vom Arbeitsumfang her bearbeitbar bleiben müssen. Auch sollte eine neu gestaltete User Story weiterhin einen klaren inhaltlichen Fokus besitzen.

Das Problem bei dieser Herangehensweise ist jedoch, dass diese neuen, aus den ursprünglichen User Stories durch Umverteilung abgeleiteten User Stories nicht mehr notwendigerweise einen direkten Nutzwert aufweisen müssen. Dieses ist der Tatsache geschuldet, dass diese durch Umverteilung entstandenen User Stories nicht unbedingt eine für den Anforderer nutzbare Einheit bilden. Im schlechtesten Fall könnte so ein Sprint geplant werden, der keinen Nutzwert für den Anwender erzeugt, da durch die Umverteilung zum einen die direkte Zuordnung zum Kunden verloren geht, weil eine so abgeleitete User Story aus mehreren User Stories unterschiedlicher Kunden entstehen kann. Zum anderen geht auch der Bezug zur ursprünglichen Nutzwertabschätzung verloren. Durch eine reine Umstrukturierung der Arbeitsaufwände auf Ebene der User Stories kann das Problem der im Arbeitsumfang wachsenden User Stories bei einer Vereinigung umgangen werden. Jedoch ergeben sich durch eine ausschließliche Umverteilung Probleme bzgl. der Nutzwertbestimmung und der Nachverfolgbarkeit aus Sicht des Anforderers, da es für ihn nicht mehr vollständig transparent ist, in welche Arbeitspakete seine ursprüngliche Anforderung aufgeteilt wurde und auf welche User Stories seine ursprüngliche User Story verteilt ist. Somit ist es für die Anforderer wesentlich komplexer den Fortschritt der Entwicklung nachzuvollziehen und für den Product Owner wird eine Priorisierung der User Stories und eine Sprint-Planung aufwendiger, da unter Umständen mehrere User Stories erst im Zusammenspiel miteinander einen Nutzwert erzeugen. So müssten z. B. drei User Stories (1*, 2* und m*) implementiert werden, um den Nutzwert der ursprünglichen User Story zu erzeugen (s. Abbildung 24).

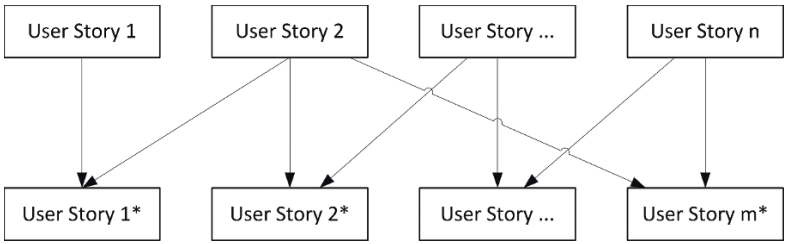


Abbildung 24: Umverteilung von Arbeitsaufwänden zur Abbildung von Synergien zwischen Anforderungen (eigene Abbildung)

7.2.3 Abbildung von Synergiepotenzialen auf Task-Ebene

Wie gezeigt, hat eine reine Umverteilung von Arbeitsumfängen auf Ebene der User Stories einige Vorteile gegenüber der reinen Vereinigung von User Stories, jedoch bringt diese Herangehensweise auch Nachteile mit sich. Dazu gehört z. B. die verringerte Transparenz und Nachvollziehbarkeit aus Sicht der Anforderer. Um diesen Nachteil zu umgehen ist zu prüfen, ob eine Abbildung von Synergiepotenzialen auf Ebene der Tasks möglich ist. Diese Idee wird in Abbildung 25 grob veranschaulicht.

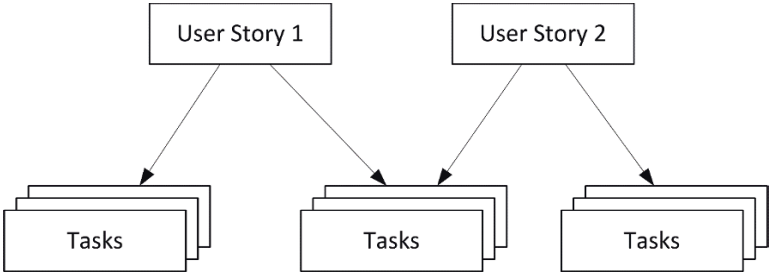


Abbildung 25: Abbildung von Synergien zwischen Anforderungen auf Ebene der Tasks (eigene Darstellung)

Eine Bündelung der Arbeitsumfänge, welche Synergiepotenziale beinhalten, auf Ebene der Tasks hat Vorteile gegenüber einer Abbildung auf Ebene der User Stories. Ein Vorteil ist, dass die User Stories aus Sicht des Kunden unangetastet bleiben und die Nachvollziehbarkeit und die Transparenz innerhalb des Product Backlogs weiterhin gewährleistet bleibt. Auch ist es für den Product Owner weiterhin eindeutig nachvollziehbar, welche User Stories innerhalb des nächsten Sprints umgesetzt werden, unter der Annahme, dass der Sprint komplett abgeschlossen wird. Die User Stories als solche werden dadurch nicht verändert und aus Sicht der Anforderer und Stakeholder bliebe der Prozess nahezu unverändert.

Der Nachteil dieser Herangehensweise ist, dass die Synergiepotenziale nicht vorzeitig dokumentiert werden könnten und die Identifikation, sowie Nutzung der Synergiepotenziale dem Entwicklerteam überlassen bleiben würde. Im Anschluss an das Sprint Planning Meeting müssten die entsprechenden User Stories, deren Umsetzung das Team zugesagt hat, in einzelne

Tasks heruntergebrochen werden. Hierbei läge es dann in der Verantwortung des Teams entsprechend dem Schema aus Abbildung 25 die neuen Tasks zu erstellen. Jedoch kann durch die reine Dokumentation, Identifikation und Nutzung von Synergiepotenzialen auf Ebene der Tasks nicht systematisch sichergestellt werden, dass entsprechende User Stories überhaupt innerhalb eines Sprint Planning Meetings besprochen werden. Bei dieser Vorgehensweise besteht also die Gefahr, dass Synergiepotenziale nicht erkannt und die damit verbundenen User Stories nicht gemeinschaftlich betrachtet werden.

7.2.4 Zusammenfassung

Nachdem drei unterschiedliche Möglichkeiten zur Abbildung von Synergien zwischen Anforderungen im Kontext von agiler Anforderungsdokumentation mit User Stories und Tasks zunächst grob untersucht wurden, zeigt sich, dass keine dieser direkt verwendet werden kann, um Synergien zwischen Anforderungen unterschiedlicher Anforderer abzubilden.

Eine einfache Zusammenlegung von User Stories lässt diese im Umfang zu groß werden und ihren thematischen Fokus verlieren. Eine reine Umverteilung der Arbeitsumfänge verringert die Transparenz des Prozesses. Eine Verarbeitung von Synergiepotenzialen auf Task-Ebene löst zwar die Probleme auf Ebene der User Stories, sie hat jedoch den Nachteil, dass es bei einem rein auf die Task-Ebene konzentrierten Verfahren zu Situationen kommen kann, in denen vorhandene Synergiepotenziale nicht genutzt werden. Diese Erkenntnisse über die Vor- und Nachteile der unterschiedlichen Methoden werden in Tabelle 11 zusammengefasst.

Die für eine Bündelung von Anforderungen und die damit verbundene Nutzung von Synergiepotenzialen wichtigen Aspekte sind somit, dass eine dokumentierte Anforderung, z. B. in Form einer User Story, zu einem Stakeholder assoziiert ist, welcher für die Akzeptanzkriterien und die Abnahme verantwortlich ist. Außerdem ist zu beachten, dass eine User Story immer mit einem bestimmten Nutzwert aus Sicht des Kunden und dem unterstützten Prozess verbunden ist und dieser Nutzen erst erzielt wird, wenn die User Story vollständig implementiert und vom Anforderer abgenommen ist (Bleek und Wolf 2008). Darüber hinaus möchte der Anforderer in der Lage sein, seine Anforderung entlang der Umsetzung nachzuverfolgen, um abschätzen zu können, wann seine Anforderung im System realisiert wird. Außerdem ist bei der Abbildung von Synergiepotenzialen darauf zu achten, dass die abgeleiteten Anforderungen weiterhin in ihrem Arbeitsaufwand team- und sprintabhängige Obergrenzen nicht überschreiten.

Tabelle 11: Vergleich zwischen den Möglichkeiten zur Abbildung von Synergien mit User Stories und Tasks

	Vorteile	Nachteile
Vereinigung von User Stories	• einfache Handhabung • zu erwartende Nutzwerte können aggregiert werden	• Umfang einer User Story kann zu groß werden • User Story nicht mehr thematisch fokussiert
Umverteilung der Arbeitsaufwände auf neue User Stories	• User Stories thematisch fokussiert • Umfang einer User Story bleibt begrenzt	• Nutzwertabschätzung nur noch indirekt möglich • für Stakeholder und Anforderer nicht mehr transparent • Mapping zwischen abgeleiteten User Stories und ursprünglichen Anforderungen
Abbildung der Synergiepotenziale auf Task-Ebene	• User Stories bleiben unangetastet • Nutzwertbestimmung & Transparenz bleiben erhalten	• keine Systematik für eine frühzeitige Erkennung von Synergiepotenzialen • Identifikation & Nutzung von Potenzialen obliegt ausschließlich dem Team

7.3 Abbildung von Synergien mittels eines neuen Produktrückstandselementes

Wie im vorherigen Unterkapitel gezeigt, lassen sich Synergiepotenziale mit den in Scrum bekannten Elementen User Story und Task nur unzureichend abbilden. Es zeigte sich, dass die Abbildung von Synergieeffekten auf Ebene der Tasks den Vorteil hat, dass die User Story als solche nicht angepasst werden muss und der Anforderer weiterhin „seine“ Anforderungen entlang des Umsetzungsprozesses verfolgen kann. Auch hat der Product Owner den Vorteil, dass er die Anforderungen weiterhin direkt Anforderern zuordnen kann. Der Nachteil dieser Methode ist jedoch, dass Tasks innerhalb von Scrum erst zu einem späten Zeitpunkt innerhalb des Prozesses erzeugt werden. Auch werden diese nicht von allen Prozessbeteiligten verwaltet, sondern ausschließlich vom Entwicklungsteam.

Synergien zwischen Anforderungen sollten so dokumentiert und verwaltet werden, dass die einzelnen Anforderer weiterhin in der Lage sind ihre eigenen Anforderungen zu verfolgen. Dar-

über hinaus sollten die Synergiepotenziale frühzeitig innerhalb des Prozesses erkannt und dokumentiert werden, sodass diese einen Einfluss auf die Iterationsplanung haben können. Nur so ist gewährleistet, dass Synergiepotenziale und die vorhandenen Einsparpotenziale z. B. für Innovationen genutzt werden können.

Aus den Vorüberlegungen aus Kapitel 7.2 lässt sich ableiten, dass Synergiepotenziale sich untergeordnet zu User Stories aber oberhalb der Task-Ebene abbilden lassen. Deswegen wird in diesem Kapitel untersucht, wie ein neues Produktrückstandselement zwischen User Stories und Tasks eingebunden werden kann, welche Eigenschaften es besitzen muss und wie es innerhalb der Scrum-Methodik verankert werden kann. Darüber hinaus ist zu klären, in welcher

Art und Weise diese Elemente dokumentiert werden. Außerdem ist zu prüfen, welche Auswirkungen und Implikationen sich für den prozessualen Ablauf von Scrum neu ergeben.

Neben der Frage nach dem Aufbau dieses neuen Produktrückstandselementes ist auch noch ungeklärt, welche Rolle bzw. welche Rollen die Verantwortung für die Identifikation und Nutzung von Synergiepotenzialen übernehmen können. Deswegen wird im folgenden Unterkapitel (7.3.1) zunächst die prozessuale Verantwortung geklärt und daran anschließend wird der Aufbau dieses neuen Elements genauer erörtert, bevor abschließend die Auswirkungen auf den gesamten Prozess betrachtet werden.

7.3.1 Prozessuale Verantwortung

Das Anliegen dieses Unterkapitels ist es zu klären, welche Rolle bzw. welche Rollen geeignet sind, um sich mit der Identifikation und der Verwaltung von Synergiepotenzial zwischen Anforderungen zu beschäftigen. In der klassischen Definition von Scrum existieren neben den Anforderern noch die drei weiteren Rollen, die des Product Owners, des ScrumMasters und des Teams (s. Kapitel 3.2). Die notwendigen Kompetenzen für diese Aufgaben sind der Überblick über alle bestehenden Anforderungen, ein Grundwissen über die Prozesse auf Seiten des Anforderungsstellers sowie auch ein Grundwissen über die Zusammenhänge und Abläufe innerhalb des zu entwickelnden Softwaresystems.

Dabei ist die Rolle des ScrumMasters die Rolle, deren Aufgaben- und Verantwortungsbeschreibung am wenigsten mit den Aufgaben der Identifikation und Nutzung von Synergiepotenzialen übereinstimmt. Die Rolle des ScrumMasters ist die des Coaches für den Prozess und dessen Überwachung. Die Aufgaben, die im Zusammenhang mit der Einführung eines neuen Produktrückstandselementes stehen, sind somit die Überwachung des Prozesses und die Einhaltung der für diesen Prozess definierten Regeln.

Die Delegierung dieser Aufgaben auf einen oder mehrere Anforderer hätte den Vorteil, dass das Team keine zusätzlichen Aufgaben zu erledigen hätte. Vorteilhaft aus Sicht der IT wäre, dass sich die internen Abläufe nicht verändern würden. Ein weiterer Vorteil dieser Herangehensweise wäre, dass Anforderer direkt in den Prozess mit einbezogen werden und auch teilweise für optimierte Abarbeitung mit verantwortlich wären.

Jedoch kann von Anforderern in der Regel nicht verlangt werden, dass sie diese Aufgabe wahrnehmen. Dies ist zum einen in der Tatsache begründet, dass Vertreter aus unterschiedlichen organisatorischen Einheiten meist neben dem Tagesgeschäft keine ausreichenden freien Kapazitäten besitzen, um die Aufgaben vollumfänglich zu erledigen. Auch eine Verteilung der Auf-

gabe auf mehrere Anforderer ist herausfordernd, da sich diese Anforderer zum einen untereinander abstimmen müssten und zum anderen könnten durch ein arbeitsteiliges Vorgehen Synergiepotenziale übersehen werden. Ein weiterer Grund, der ebenfalls gegen die Verantwortung der Anforderer spricht, ist, dass diese meist kein hinreichendes Wissen über das Softwaresystem selbst haben, sodass Synergien, die sich nicht direkt auf einer inhaltlichen, sondern vielmehr auf einer technischen Ebene ergeben, von ihnen nicht erkannt werden könnten. Dadurch würden vorhandene Potenziale verloren gehen. Ein weiterer Grund, der gegen die Übertragung der Aufgabe auf die Anforderer spricht, ist, dass vorhandene Synergiepotenziale auch innerhalb der IT z. B. für Innovation genutzt werden und unter Umständen nicht vollumfänglich an den bzw. die Anforderer weitergegeben werden sollen.

Eine Alternative wäre, dem Entwicklerteam die Verantwortung zu übertragen. Das hätte den Vorteil, dass das Entwicklerteam selbstverantwortlich bestimmen kann, welche Potenziale sinnvoll genutzt werden können und welche vorhandenen Potenziale vielleicht ungenutzt bleiben sollten, da sich andere negative Auswirkungen ergeben. Es ist davon auszugehen, dass innerhalb des Entwicklerteams das größte technische Wissen um das Softwaresystem vorhanden ist. Auch würden Synergiepotenziale, die ausschließlich vom Entwicklerteam verwaltet und genutzt werden würden, ausschließlich innerhalb der IT liegen, sodass es dem Entwickler obliegen würde, die genutzten Potenziale an den Kunden weiter zu geben oder aber diese innerhalb der IT zu nutzen.

Der Nachteil dieser Herangehensweise ist jedoch, dass die Priorisierung der Anforderungen nicht in der Verantwortung des Entwicklerteams liegt. Somit müsste das Team sich darauf beschränken Synergien zu nutzen, die sich durch die Sprintgestaltung des Product Owners ergeben. Das Team wäre somit nicht in der Lage, direkt steuernd auf den Prozess einzuwirken, wenn der Product Owner nicht mit einbezogen wird. Eine weitere Herausforderung ist, dass das Entwicklerteam nicht unbedingt eine vollständige Übersicht über alle Anforderungen aller Anforderungssteller innerhalb des Product Backlogs hat.

Die letzte Rolle, die bzgl. ihrer Eignung untersucht wird, ist die Rolle des Product Owners. Bei der folgenden Bewertung ist zu beachten, dass in diesem Kontext davon ausgegangen wird, dass der Product Owner von der IT gestellt wird und als Bindeglied zwischen den Kunden aus unterschiedlichen Standorten und verschiedenen organisatorischen Einheiten und dem Entwicklerteam fungiert. Der Nachteil, der sich ergibt, wenn der Product Owner für diese Aufgabe eingesetzt wird, ist, dass dieser bei größeren Projekten schnell zum Nadelöhr innerhalb des Prozesses wird und seine Aufgaben nicht mehr vollumfänglich ausführen kann. Wie aber bereits in Kapitel 4.1 erläutert, gibt es Konzepte und Möglichkeiten den Product Owner zu entlasten, sodass durch die Anwendung entsprechender Konzepte eine ausreichende Kapazität erreicht werden kann. Außerdem wird die Priorisierung der Anforderungen zusätzlich durch das in Kapitel 6 beschriebene Vorgehen vereinfacht. Ein weiterer Nachteil ist, dass der Product Owners weniger in die direkte Entwicklung des Produktes eingebunden ist als das Entwicklerteam. Deswegen können dem Product Owner nicht alle technische Details bekannt sein, welche für die Identifikation von Synergiepotenzialen notwendig wären.

Die Vorteile einer Übertragung der Verantwortung für die Identifikation und Dokumentation von Synergiepotenzialen an den Product Owner wäre, dass es bereits in seiner Verantwortung liegt, das Product Backlog zu pflegen und die Priorisierung der Anforderungen innerhalb diesem vorzunehmen. Durch diese Tätigkeit wird gewährleistet, dass der Product Owner eine sehr gute Übersicht über alle bereits dokumentierten Anforderungen besitzt. Darüber hinaus ist der Product Owner dadurch direkt in der Verantwortung Anforderungen innerhalb des Product

Backlogs zusammenzufassen und diese entsprechend zu priorisieren, sodass diese Anforderungen innerhalb eines Sprints betrachtet werden. Außerdem stellt der Product Owner den „Single Point of Contact“ für die unterschiedlichen Anforderungssteller dar. Aufgrund dieser Tatsache hat er zusätzlich auch einen Überblick über die Anforderungen, welche noch wenig detailliert sind und noch nicht als User Stories innerhalb des Backlogs dokumentiert sind. Außerdem ist der Product Owner auch erster Ansprechpartner der Entwicklerteams bei Rückfragen zu Anforderungen.

Zusammenfassend lässt sich feststellen, dass der Product Owner die Rolle innerhalb des Prozesses ist, die sich als die geeignetste Rolle für die Identifikation und Dokumentation von Synergiepotenzialen erweist. Im Kontext dieser Arbeit ist davon auszugehen, dass der Product Owner ein tiefergehendes Wissen über die technischen Zusammenhänge innerhalb des zu entwickelnden Softwaresystems besitzt. Hierbei ist jedoch anzumerken, dass je nach Art und Umfang des technischen Wissens des Product Owner, er bei dieser Tätigkeit durch das Entwicklerteam unterstützt werden sollte.

Aufgrund der Tatsache, dass die Verantwortung mehrheitlich beim Product Owner liegen wird, wird dieses neue Produktrückstandselement Product Owner Story genannt. Dieses geschieht analog zur User Story. Eine User Story beschreibt eine Anforderung aus Sicht eines Anwenders (engl. User). Außerdem sollten User Stories auch von den eigentlichen Anwendern verfasst werden (Bleek und Wolf 2008). Eine Product Owner Story beschreibt ein Synergiepotenzial zwischen mehreren User Stories aus Sicht der Product Owner (Sobiech et al. 2016).

In welcher Form eine Unterstützung durch das Entwicklerteam erfolgen könnte, wird in Kapitel 7.3.3 diskutiert. Zunächst wird jedoch in Kapitel 7.3.2 der Aufbau von Product Owner Stories erläutert und wie mit ihrer Hilfe Synergiepotenziale dokumentiert werden können.

7.3.2 Aufbau von Product Owner Stories

Zunächst ist für Product Owner Stories zu klären, an welcher Stelle und wie sie sich in die bestehende Struktur aus Epics, User Stories und Tasks einfügen lassen (s. Abbildung 22: Zusammenhang zwischen Epic, User Story & TaskAbbildung 22). Wie bereits diskutiert, sollten Product Owner Stories zwischen User Stories und Tasks eingebunden werden, da sie gleichartige Arbeitsumfänge mehrerer User Stories bündeln sollen.

Durch die Notwendigkeit, dass eine Product Owner Story Arbeitsinhalte aus mehreren User Stories bündelt, ergibt sich an dieser Stelle zunächst allgemein eine n:1 Beziehung. Es können also theoretisch Arbeitsumfänge aus n unterschiedlichen User Stories in einer Product Owner Story zusammengefasst werden.

Da in einer Product Owner Story nur gleichartige Arbeitsumfänge unterschiedlicher User Stories gebündelt werden, müssen die verbleibenden voneinander abweichenden Arbeitsumfänge in einer neuen Product Owner Story dokumentiert werden. Somit kann eine User Story als Ergebnis der Bündelung von gleichartigen Arbeitsumfängen auf m unterschiedliche Product Owner Stories verteilt werden. Als Ergebnis dieser Einordnung ergibt sich zwischen User Stories und Product Owner Stories eine m:n-Beziehung.

Eine Task wiederum sollte genau einer Product Owner Story zugeordnet werde, bzw. wird im Rahmen der scrumtypischen Aufgabendefinitionsphase eine User Story bzw. jetzt eine Product

Owner Story in eine beliebige Anzahl an kleineren Aufgaben zerlegt. Abbildung 26 veranschaulicht diesen Zusammenhang.

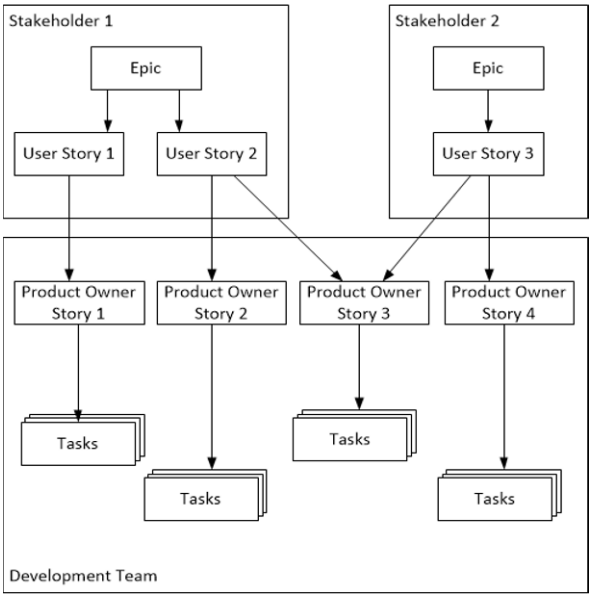


Abbildung 26: Verwendung von Product Owner Stories (eigene Darstellung Quelle: (Sobiech et al. 2016)) [6]

Wie der Abbildung 26 zu entnehmen ist, sind ebenfalls die Bereiche gekennzeichnet, in denen die Verantwortung für den Prozessfortschritt beim jeweiligen Stakeholder der Anforderung liegt, bzw. dass die Verantwortung an der Schnittstelle von User Story hin zu Product Owner Story an die IT übergeht.

Bei der Abbildung von Synergien zwischen User Stories sind zunächst drei unterschiedliche Fälle zu betrachten:

1. Eine User Story weist mit keiner anderen User Story Synergiepotenziale auf (**kein Synergiepotenzial**)
2. Mehrere User Stories sind sich so ähnlich, dass sie in einer neuen Product Owner Story zusammengefasst werden können (**totales Synergiepotenzial**)
3. Mehrere User Stories weisen teilweise einen gemeinsamen Arbeitsumfang auf und jeweils Restaufwände, die nicht miteinander vereinbar sind. (**partielles Synergiepotenzial**)

Diese drei Fälle werden auch in der Abbildung 27 veranschaulicht.

[6] Im Kontext dieser Arbeit wird der Product Owner von der IT gestellt

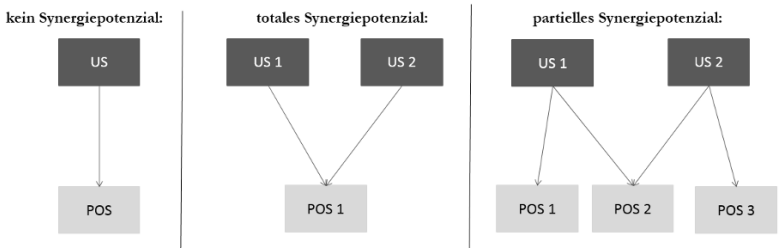


Abbildung 27: Arten von Synergiepotenzialen (eigene Darstellung)

Bei der Verwendung von Product Owner Stories ist davon auszugehen, dass der ursprüngliche Nutzwert, der für die zugehörigen User Stories beschrieben wurde, erhalten bleibt. Da durch Product Owner Stories nur eine Umverteilung der Arbeitsaufwände erfolgt, wird also der Nutzen einer User Story erzielt, wenn alle zugeordneten Product Owner Stories implementiert sind.

Außerdem ist zunächst noch zu klären, wann ein Synergiepotenzial zwischen Anforderungen vorhanden ist. Für die Bewertung sollte dabei folgendes Kriterium gelten:

$$\Sigma\, Aufwand\ User\ Stories \geq \Sigma\, Aufwand\ Product\ Owner\ Stories$$

Diese Formel sagt aus, dass der zu erwartende Aufwand für die Implementierung der korrespondierenden Menge von Product Owner Stories genauso groß oder geringer sein muss, als der zu erwartende Aufwand für die Implementierung der User Stories. Somit soll sichergestellt werden, dass nur Anforderungen gebündelt werden, die ein tatsächliches Sparpotenzial besitzen und nicht jede denkbare Bündelung vollzogen wird. Die Bedingung ist explizit mit $\geq$ belegt, da nicht auszuschließen ist, dass weitere positive Effekte durch Synergiepotenziale entstehen können, welche sich nicht ausschließlich durch Arbeitsaufwandsersparnis bewerten lassen. Der Nutzwert der User Stories spielt bei dieser Betrachtung keine Rolle, da sich dieser indirekt über die User Stories ergibt und nicht verändert wird.

Im Folgenden werden nun die drei Arten von Synergien genauer betrachtet und es wird aufgezeigt, wie sich im jeweiligen Fall die Product Owner Stories aus den zugehörigen User Stories ergeben.

Kein Synergiepotenzial:

Im ersten Fall, in dem eine User Story keine Synergiepotenziale mit einer anderen User Story aufweist, ergibt sich die Product Owner Story trivialerweise nur aus Informationen einer User Story. Dies führt dazu, dass diese User Story direkt zu einer Product Owner Story wird. Änderungen bzw. Ergänzungen sind an dieser Stelle nicht notwendig. Sollte noch keine Aufwandsschätzung für die User Story durchgeführt worden sein, so ist dies nachzuholen, damit die Product Owner Story, basierend auf dem zu erwartenden Nutzwert und ihrem Aufwand, im Product Backlog priorisiert eingeordnet werden kann.

Totales Synergiepotenzial:

Im Fall eines totalen Synergiepotenzials zwischen Anforderungen sind die Inhalte der betrachteten User Stories so ähnlich, dass sie sich in einer Product Owner Story zusammenfassen lassen. Dabei ist zu beachten, dass die resultierende Product Owner Story in ihrem Aufwand nicht

zu groß wird, sodass diese innerhalb eines Sprints nicht mehr sinnvoll bearbeitet werden kann. Bei der Erstellung der Product Owner Story ist ggf. die Beschreibung so anzupassen, dass sie alle relevanten Aspekte der zu verarbeitenden User Stories umfasst.

Bei den Akzeptanzkriterien ist darauf zu achten, dass für die neue Product Owner Story alle Akzeptanzkriterien aller zugeordneten User Stories gelten. Nur so kann sichergestellt werden, dass die entsprechenden User Stories vollumfänglich abgeschlossen werden und der Nutzen jeder User Story erreicht wird. An dieser Stelle ergibt sich somit der Nutzwert der Product Owner Story als Summe der Nutzwerte der zugeordneten User Stories.

Sollte mindestens eine der zugeordneten User Stories eine feste Fertigstellungsfrist (Deadline) besitzen, so erhält auch die resultierende Product Owner Story eine Deadline. Die Frist der resultierenden Product Owner Story ergibt sich dabei als minimale Frist aller Abgabetermine der zugeordneten User Stories. Dieser Sachverhalt kann auch wie folgt definiert werden:

Sei D die Menge aller Abgabefristen der zugeordneten User Stories und $d \in D$. d ist Deadline der resultierenden Product Owner Story: $\leftrightarrow \forall y \in D: (y \leq d \ \rightarrow \ d \leq y)$.

Wie bereits beschrieben ist die Bedingung für die Zusammenfassung mehrerer User Stories zu einer neuen Product Owner Story, dass diese einen geringeren Aufwand für die Umsetzung aufweist, als die separate Bearbeitung der einzelnen User Stories. Hierbei kann es anschließend zu einer Prioritätenverschiebung innerhalb des Product Backlogs kommen.

Wird von einem Product Backlog ausgegangen, in dem User Stories nach dem Verhältnis von Nutzen zu Aufwand priorisiert sind, kann durch eine Bündelung von bspw. zwei User Stories eine solche Verschiebung beobachtet werden. Dieses wird an einem fiktiven Beispiel in Abbildung 28 verdeutlicht.

Abbildung 28 zeigt auf der linken Seite einen Ausschnitt aus einem fiktiven Product Backlog, in dem die User Stories nach ihrem Verhältnis von Nutzwert und Aufwand priorisiert sind. In diesem Beispiel weisen die User Stories US 1 und US 5 ein totales Synergiepotenzial auf. Aus diesen beiden User Stories wird wie bereits beschrieben eine neue Product Owner Story abgeleitet. Wie zu erkennen ist, wird der Nutzwert beider User Stories addiert und der Product Owner Story zugeordnet. Der Aufwand wird an dieser Stelle durch das Entwicklerteam neu mit 5 Storypunkten (SP) bewertet.

Der rechte Ausschnitt aus einem Backlog in Abbildung 28 zeigt die Reihenfolge der User und Product Owner Stories nach der Bündelung von User Story 1 und User Story 5. Diese beiden User Stories sind im Backlog durch die Product Owner Story POS 1 ersetzt worden. Es zeigt sich, dass diese Product Owner Story in diesem fiktiven Beispiel an der dritten Stelle im Backlog einsortiert werden würde. Für User Story US 5 ergibt sich durch diese Bündelung ein Vorteil, da sie im Backlog vor User Story US 4 einsortiert wird. Für User Story US 1 ergibt sich jedoch der Nachteil, dass sie von der ersten, also der höchst priorisierten Anforderung auf den dritten Rang zurückfällt.

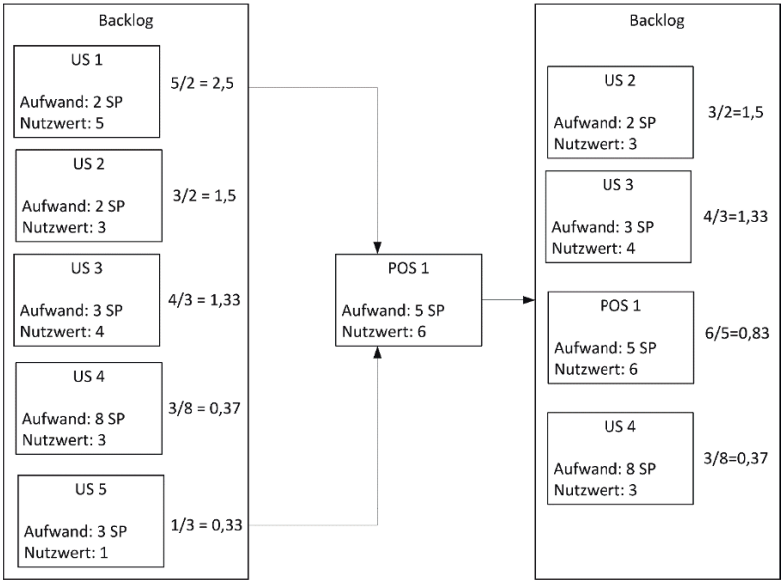


Abbildung 28: Prioritätenverschiebung durch Bündelung von User Stories (eigene Darstellung)

An dieser Stelle kann es somit notwendig werden, einen Trade-Off zwischen der Nutzung von Synergiepotenzialen und der Kundenzufriedenheit zu finden. Durch negative Effekte, wie die spätere Auslieferung eines Features an einen Kunden als ursprünglich angedacht, kann die Zufriedenheit des Anforderers negativ beeinflusst werden. Um diese Effekte zu begrenzen, könnte z. B. das Backlog nur ausschnittsweise betrachtet werden, sodass Verschiebungen nur in einem begrenzten Rahmen erfolgen können. Die Priorisierung und die daraus entstehende Ordnung würde dann jedoch nicht mehr für das gesamte Backlog gelten, sondern jeweils nur für den betrachteten Ausschnitt. Alternativ könnte auch eine abweichende Formulierung für die Synergiebedingung erarbeitet werden. Eine Möglichkeit für Synergiebedingungen, die die negativen Auswirkungen begrenzen, wäre bspw. die folgende Formulierung der Bedingung:

1. $\Sigma\, Aufwand\ User\ Stories \geq \Sigma\, Aufwand\ Product\ Owner\ Stories$
2. $\max\left(\frac{Nutzwert\ US}{Aufwand\ US}\right) \leq \frac{Nutzwert\ POS}{Aufwand\ POS}$

Die erste Bedingung ist die allgemeine und bereits erläuterte Synergiebedingung. Um die negativen Auswirkungen zu begrenzen, könnte eine weitere Forderung sein, dass die resultierende Product Owner Story ein gleiches oder besseres Verhältnis von Nutzwert zu Aufwand aufweist, als das Verhältnis der höchst priorisierten zugeordneten User Story. Somit könnten Anforderungen, welche aus Gründen der Nutzung von Synergien zusammengelegt werden, nur positive Veränderungen bzgl. der Priorisierung innerhalb des Backlogs erhalten. Der Nachteil dieser Formulierung ist, dass nur noch wenige vorhandene Potenziale genutzt werden können. Im Sinne eines Trade-Offs kann die zweite Bedingung durch die Addition eines beliebigen Δ relaxiert werden. Je nach Projektkontext können die Anforderer in diesen Prozess mit einbezogen

werden, um einen sinnvollen Kompromiss zu finden. Sollte jedoch die Nutzung der Potenziale größtenteils in der IT verbleiben, so sollte eine möglichst wenig restriktive Formulierung der Synergiebedingung verwendet werden, um einen maximalen Nutzen innerhalb der IT zu erhalten.

Partielles Synergiepotenzial:

Ein partielles Synergiepotenzial zwischen Anforderungen beschreibt einen Fall, in dem alle beteiligten User Stories einen gemeinsamen Arbeitsumfang aufweisen, einige oder alle User Stories aber noch einen zusätzlichen Arbeitsumfang enthalten, der nicht mehr direkt mit in einer Product Owner Story zusammengefasst werden kann. Dieses wird in Abbildung 29 am Beispiel von zwei User Stories graphisch veranschaulicht.

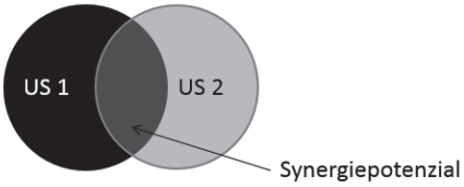


Abbildung 29: Partielles Synergiepotenzial zwischen zwei User Stories (eigene Darstellung)

Aus einem solchen Beispiel könnten bspw. drei Product Owner Stories abgeleitet werden (s. auch Abbildung 27). Eine Product Owner Story würde den Arbeitsumfang beschreiben, den beide Anforderungen gemeinsam aufweisen. Zusätzlich würde jeder User Story eine weitere Product Owner Stroy zugeordnet werden, die die jeweiligen nicht zu vereinbarenden Arbeitsumfänge beschreibt. Hierbei ergeben sich im Wesentlichen zwei Herausforderungen bzgl. der Dokumentation und Verwaltung. Zum einen ist zu klären, wie mit den Akzeptanzkriterien zu verfahren ist, zum anderen ist zu erörtern, wie die durch partielle Synergiepotenziale entstandenen Product Owner Stories im Product Backlog priorisiert werden. Die Besonderheit an dieser Stelle ist, dass nicht jede Product Owner Story einen direkten Nutzwert aufweisen muss. Dieser entsteht ggf. nur in Kombination mit weiteren Product Owner Stories.

Bei der Ableitung von Product Owner Stories, die durch partielle Synergiepotenziale entstanden sind, liegt es in der Verantwortung des Product Owners, zusammen mit dem Entwicklerteam die Akzeptanzkriterien je Product Owner Story zu formulieren, sodass auch die Akzeptanzkriterien der User Stories erfüllt werden. Wird bspw. eine User Story in zwei Product Owner Stories untergliedert, so müssen die Akzeptanzkriterien eine Teilmenge der Akzeptanzkriterien der zugeordneten Product Owner Stories sein. Es handelt sich an dieser Stelle um eine Teilmenge, da durch Verknüpfung einer Product Owner Story mit mehreren User Stories weitere Akzeptanzkriterien hinzukommen können, die diese User Story jedoch nicht betreffen.

Im Zusammenhang mit Product Owner Stories, die aus User Stories mit partiellem Synergiepotenzial entstanden sind, ist darüber hinaus noch die Frage nach dem Nutzwert und der damit verbundenen Priorisierung innerhalb des Backlogs zu klären. In Scrum werden Anforderungen in der Regel nur nach umgesetzten und nicht umgesetzten Anforderungen klassifiziert. Ein Teilnutzen oder eine Teilfertigstellung wird dabei nicht berücksichtigt. Dieses Konzept wird auch in der Berechnung der „Velocity“ berücksichtigt. Auch hier werden nur vollständig umgesetzte

Anforderungen mit in die Berechnung einbezogen (vgl. Cohn 2004). Ist nun eine User Story in mehrere Product Owner Stories untergliedert, so müssen alle diese Product Owner Stories implementiert werden, um den Nutzen der ursprünglichen User Story zu erzielen. Im Umkehrschluss muss somit nicht jede Product Owner Story einen expliziten Nutzwert aufweisen. Der Nutzwert ist an dieser Stelle nur noch für eine Menge von Product Owner Stories definiert. Der Nutzwert einer Menge von Product Owner Stories wäre dann definiert als Summe der Nutzenwerte der User Stories, die vollumfänglich implementiert werden. Aufgrund dieser Definition ist es jedoch nicht mehr möglich einzelne Product Owner Stories innerhalb des Product Backlogs zu priorisieren, da diese nur noch einen Aufwand aufweisen müssen und keinen direkten Nutzen mehr.

Um diesen Sachverhalt besser zu veranschaulichen wird davon ausgegangen, dass zwei User Stories mit einem partiellen Synergiepotenzial mit drei Product Owner Stories abgebildet werden (s. Abbildung 30). Außerdem sei der Nutzwert jeder dieser beiden User Stories gleich fünf.

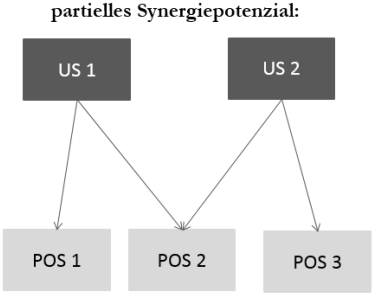


Abbildung 30: Abbildung von User Stories mit partiellem Synergiepotenzial auf Product Owner Stories (eigene Darstellung)

Aus den drei abgeleiteten Product Owner Stories lassen sich theoretisch acht unterschiedliche Mengen erzeugen, die unterschiedliche Nutzwerte besitzen. Diese werden in Tabelle 12 dargestellt.

Wie Tabelle 12 zu entnehmen ist, gibt es drei Teilmengen der Product Owner Stories, die unterschiedliche Nutzwerte größer als null erzeugen. Dementsprechend ergeben sich aus dieser Zerlegung mehrere unterschiedliche Priorisierungen innerhalb des Backlogs. Zum einen könnte die komplette Menge an Product Owner Stories innerhalb des Backlogs einsortiert werden. Somit würden die beiden ursprünglichen Anforderungen dieselbe Position innerhalb des Backlogs einnehmen. Zum anderen könnten auch die jeweiligen nutzwertbehafteten Teilmengen einsortiert werden ({POS 1, POS 2}, {POS 2, POS 3}). Die Teilmengen weisen einen geringeren Aufwand auf, als die Betrachtung aller assoziierten Product Owner Stories. Dadurch ist es, je nach Umfang, wahrscheinlicher, dass sie innerhalb eines Sprints umgesetzt werden. Die verbleibende Teilmenge würde dann in der Priorisierung für den nächsten Sprint einen Boost erfahren. Wird in einem Sprint bspw. POS 1 und POS 2 implementiert, so wird ein Nutzwert von 5 erzeugt (vgl. Tabelle 12). Wie ebenfalls aus Tabelle 12 zu entnehmen ist, würde die Menge {POS 2, POS 3} ebenfalls einen Nutzwert von fünf erzeugen. Da aber in diesem Beispiel POS 2 bereits in einem vorherigen Sprint implementiert wurde, würde die Umsetzung von POS 3

dazu führen, dass ein Nutzwert von fünf erzeugt wird. Der Nutzwert der Menge bleibt erhalten, jedoch hat sich der Aufwand verringert, da POS 2 bereits umgesetzt ist und sich somit das Verhältnis von Nutzwert und Aufwand verbessert. Dies führt wiederum dazu, dass die Anforderung innerhalb des Backlogs in der Priorisierung nach oben steigen kann.

Tabelle 12: Beispielhafte Darstellung für die Nutzwertverteilung unterschiedlicher Mengen von Product Owner Stories

Menge non Product Owner Stories	Nutzwert
{POS 1}	0
{POS 2}	0
{POS 3}	0
{POS 1, POS 2}	5
{POS 1, POS 3}	0
{POS 2, POS 3}	5
{POS 1, POS 2, POS 3}	10

Nachdem drei mögliche Ausprägungen von Synergiepotenzialen zwischen Anforderungen erläutert und deren Abbildung durch Product Owner Stories verdeutlicht wurden, kann nun analog zu Abbildung 22 das neue Rückstandselement der Product Owner Story mit in diese Hierarchie aufgenommen werden. Dieses wird in Abbildung 31 dargestellt. Hierbei ist zu beachten, dass es sich bei der Priorisierung innerhalb des Product Backlogs nicht mehr um eine totale Ordnung handelt, da die neue Ordnung, die durch die Einführung von Product Owner Stories entsteht, nicht mehr antisymmetrisch ist. Die nachfolgende Bedingung für Antisymmetrie gilt nicht mehr, da die Implikation x = y nicht mehr erfüllt wird. Mehrere Product Owner Stories können dieselbe Position innerhalb des Backlogs belegen und somit können auch transitiv mehrere unterschiedliche User Stories dieselbe Position innerhalb des Backlogs einnehmen.

$$x \leq y \wedge y \leq x \rightarrow x = y$$

Es zeigt sich, dass eine m:n-Beziehung zwischen den User Stories und Product Owner Stories besteht. Product Owner Stories unterscheiden sich dabei fast nicht von User Stories. Lediglich die inhaltliche Ausrichtung ist eine andere. Da Iterationen nicht mehr mit User Stories, sondern mit Product Owner Stories gefüllt werden, ist es notwendig, dass der Aufwand nicht mehr für User Stories, sondern für Product Owner Stories geschätzt wird.

Eine Abschätzung des Aufwandes für User Stories und den daraus abgeleiteten Product Owner Stories ist nur dann notwendig, wenn nicht direkt ersichtlich ist, ob eine Zusammenlegung mehrerer User Stories mittels Product Owner Stories tatsächlich einen wirtschaftlichen Vorteil bringt.

User Stories behalten jedoch weiterhin ihre Beschreibung, ihre Akzeptanzkriterien, ihre Abgabefristen und auch ihre Positionierung innerhalb des Backlogs, auch wenn sich diese nicht mehr direkt und als totale Ordnung ergibt, sondern nur noch indirekt durch die Positionierung der Product Owner Stories.

Abbildung 31: Zusammenhang zwischen Epic, User Story, Product Owner Story & Task (eigene Darstellung)

Es ist zusätzlich festzuhalten, dass User Stories, welche keine Synergiepotenziale mit anderen User Stories aufweisen, direkt auch als Product Owner Stories verwendet werden können. Weitere Anpassungen oder Erweiterungen sind somit nicht notwendig. Dies ist ein wesentlicher Aspekt bei der Dokumentation, da so die systematische Abbildung von Synergiepotenzialen je nach Projektsituation dynamisch ins Backlog mit aufgenommen und auch wieder entfernt werden kann, ohne größere strukturelle Änderungen. Nachfolgend werden nun die prozessualen Auswirkungen erläutert, die sich durch die Einführung von Product Owner Stories ergeben.

7.3.3 Prozessuale Auswirkungen

Nachdem Product Owner Stories als neues Produktrückstandselement eingeführt wurden, wird in diesem Unterkapitel darauf eingegangen, welche Auswirkungen sich auf Prozessebene ergeben. Hierbei stehen zwei Aspekte im Fokus der Betrachtungen. Einerseits ist zu klären, wie und in welchem Rahmen Synergiepotenziale zwischen Anforderungen sinnvoll identifiziert werden können. Andererseits ist zu klären, ob und inwieweit sich Änderungen bzgl. der Sprintplanung ergeben, da, wie festgestellt, keine totale Ordnungsrelation mehr innerhalb des Product Backlogs besteht und der Nutzwert nicht mehr für jedes Objekt separat definiert ist, sondern teilweise nur noch für Mengen von Product Owner Stories.

Wie bereits bei der Diskussion zur prozessualen Verantwortung in Kapitel 7.3.1 erläutert, ist der Product Owner der Verantwortliche für die Dokumentation, Verwaltung und Priorisierung von Product Owner Stories. Es wurde ebenfalls erläutert, dass der Product Owner nicht alleine in der Lage ist, alle Synergiepotenziale zu identifizieren, da Wissen und Informationen zu technischen Details innerhalb des Softwareproduktes nicht vorhanden sind. Deswegen ist eine Zusammenarbeit mit dem Entwicklerteam an dieser Stelle unerlässlich.

Um diese Zusammenarbeit sicherzustellen, sollte sie als essenzieller Bestandteil von Scrum verstanden werden. Deswegen wird geprüft, ob die Identifikation von Synergiepotenzialen als weiterer Punkt einem bereits bestehenden Meeting (s. Kapitel 3.3) zugeschlagen werden kann oder ob ein neues Meeting eigens zum Zweck der Prüfung von Anforderungen auf Potenziale mit in den Prozess aufgenommen werden sollte.

Im Folgenden wird untersucht, ob sich eines der vier standardmäßigen Meetings innerhalb von Scrum für eine nähere Betrachtung eignet.

Im Sprint Review Meeting werden die Ergebnisse des abgeschlossenen Sprints vor den Prozessbeteiligten vorgestellt. Da hier auch Vertreter der Kunden anwesend sind und Product Owner Stories mit dem Ziel verbunden sind Einsparungen zu tätigen und diese nicht zwangsweise vollumfänglich an den Kunden weiter zu reichen, um bspw. die Rolle des Innovators besser wahrnehmen zu können, eignet sich dieses Meeting nicht.

Beim Sprint Retrospektive Meeting soll der gelebte Prozess vom Team kritisch reflektiert werden und Verbesserungen in Bezug auf den angewendeten Prozess erzielt werden. Somit hat dieses Meeting einen anderen thematischen Fokus, welcher sich mit der Analyse von Anforderungen nicht vereinen lässt.

Das Daily Scrum Meeting ist ein täglich stattfindendes Meeting, mit dem Fokus, den aktuellen Arbeitsfortschritt innerhalb des Teams zu besprechen, Wissen auszutauschen und über aufgetretene Probleme zu informieren. Da auch dieses Meeting nicht den richtigen Fokus besitzt, zu kurz ist und zu häufig stattfindet, eignet sich auch dieses Meeting nicht, um die Identifikation von Synergiepotenzialen zu unterstützen.

Ein weiteres Meeting, welches zu betrachten ist, ist das Sprint Planning Meeting. Der Hauptaspekt dieses Meetings ist es den nachfolgenden Sprint mit User Stories bzw. mit Product Owner Stories zu füllen. Anwesend sind meistens der Product Owner, der die Anforderungen vorstellt und das Team, das diese bzgl. ihres Aufwands bewertet und sich abschließend auf einen Umfang für den nächsten Sprint verständigt. Obwohl die Teilnehmer und auch der Inhalt des Ereignisses zutreffend sind, ist dieses Meeting jedoch auch nicht geeignet, da es innerhalb des Ablaufs zu spät stattfindet. Hier können nur Synergien erkannt und verarbeitet werden, die sich innerhalb dieses Sprints ergeben. Dieses ist analog zur diskutierten Dokumentation von Synergiepotenzialen mit Tasks.

Eine Alternative wäre es zunächst zu Beginn des Meetings Synergiepotenziale innerhalb des Backlogs zu identifizieren und zu dokumentieren. Anschließend müsste der Product Owner ggf. die Priorisierung anpassen und danach könnte mit der Sprintplanung begonnen werden. Der Nachteil ist, dass der Product Owner spontan reagieren und die Priorisierung ad hoc anpassen muss. Da dieses jedoch durch die Verwendung von Product Owner Stories komplexer ist als die reine Priorisierung von User Stories, ist zumindest fraglich, ob dieses möglich ist. Außerdem sollte berücksichtigt werden, dass das Sprint Planning Meeting dadurch in seinem Umfang weiter wachsen würde. Je nach Projektsituation kann dieses Ereignis ohne einen Synergieteil bereits mehrere Stunden in Anspruch nehmen.

Die letzte Alternative ist das Backlog Grooming. Die Identifikation von Synergiepotenzialen würde thematisch gut zu diesem Meeting passen, da es ebenfalls eine vorbereitende Maßnahme für das nächste Sprint Planning Meeting ist. Außerdem ist auch die Erstellung und Verwaltung von Product Owner Stories aus User Stories ein essenzieller Bestandteil der Backlogpflege, damit Synergiepotenziale systematisch bestimmt und genutzt werden können. Dennoch kommt auch dieses Meeting für die Identifikation von Synergiepotenzialen und die Erstellung von Product Owner Stories nicht in Betracht, da zumeist auch Vertreter aus unterschiedlichen Fachbereichen mit an dem Meeting teilnehmen. Da, wie bereits erwähnt und in Abbildung 26 dargestellt, die Nutzung von Synergiepotenzialen innerhalb der IT geregelt werden soll und entstehende Potenziale auch innerhalb der IT z. B. für die Evaluation neuer Technologien oder aber auch für den Aufbau neuer Mitarbeiter genutzt werden sollen, scheidet eine Beteiligung der Anforderer aus den Fachbereichen bei der Erstellung und Verwaltung von Product Owner Stories aus.

Nach einer Analyse aller Scrum Meetings und der Prüfung, ob diese geeignet wären die Identifikation von Synergiepotenzialen innerhalb dieses Treffens zu vollziehen, zeigt sich, dass lediglich das Sprint Planning Meeting die Voraussetzungen erfüllt. Jedoch würde die Dauer deutlich steigen, sodass das Meeting ggf. in zwei Meetings unterteilt werden müsste.

In einem Szenario, in dem es bspw. zwei unterschiedliche Entwicklerteams gibt und zusätzlich ein Team, welches für die Erhebung und die Definition von Anforderungen verantwortlich ist (vgl. Vlaanderen et al. 2011), ergibt sich z. B. eine Sprintfolge wie in Abbildung 32.

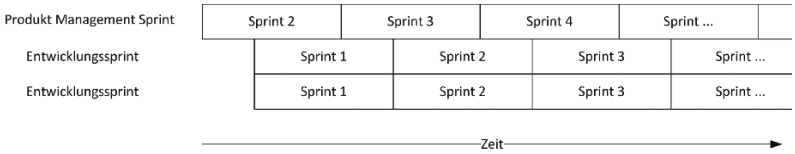


Abbildung 32: Schematische Darstellung des zeitlichen Versatzes von mehreren Sprints (in Anlehnung an (Sobiech et al. 2014))

Es ist zu erkennen, dass es einen zeitlichen Versatz zwischen dem Produkt Management Sprint und den eigentlichen Entwicklungssprints der unterschiedlichen Teams gibt. Dieser Versatz ist notwendig um sicherzustellen, dass die Anforderungen, welche innerhalb des folgenden Entwicklungssprints umgesetzt werden sollen, für diese hinreichend genau spezifiziert sind. Somit würde das Ende des jeweiligen Produkt Management Sprints, bzw. ein Termin kurz vor Ende, einen guten Zeitpunkt für die Identifikation von Synergiepotenzialen markieren. Dies ist in der Tatsache begründet, dass zu diesem Zeitpunkt ein gewisser Umfang an Anforderungen dem Backlog neu hinzugefügt wurde und bestehende Anforderungen weiter detailliert wurden. Diese Anforderungen können dann als Basis für die Identifikation von Synergiepotenzialen genutzt werden. In einem weiteren Schritt können dann bis zum definitiven Ende des Produkt Management Sprints entsprechende Product Owner Stories erstellt und das Backlog entsprechend der sich neu ergebenden Kennwerte priorisiert werden. Dies hat den Vorteil, dass das Sprintplanungstreffen weiterhin den „normalen“ Ablauf aufweisen kann und zu diesem Meeting bereits die User Stories bzw. Product Owner Stories fertig beschrieben sind. Nachdem ein möglicher Zeitpunkt für die Identifikation von Synergiepotenzialen identifiziert ist, ist zu klären, wie die vorhandenen Potenziale effizient erkannt werden können.

Kurz vor Beendigung des Product Management Sprints sollten bei diesem Meeting der Product Owner, der durch die IT gestellt wird, und das Entwicklerteam anwesend sein. In einer Projektsituation mit mehreren Entwicklerteams sollten ausgewählte Vertreter der unterschiedlichen Teams an dem Meeting teilnehmen. Dies dient dem Zweck, dass nur die notwendigen Ressourcen für dieses Meeting gebunden werden und zielorientiert gearbeitet werden kann.

Um sicherzustellen, dass alle vorhandenen Potenziale zwischen Anforderungen erkannt werden können, müsste jede dokumentierte Anforderung in Beziehung mit jeder anderen dokumentierten Anforderung auf ein vorhandenes Potenzial hin überprüft werde. Die für die Beurteilung benötigte Anzahl der Paarvergleiche berechnet sich bei einer Matrixdarstellung der Anforderungen mit n zu betrachtenden Anforderungen wie folgt (vgl. Saaty 1987):

$$\frac{n * (n-1)}{2}$$

Die Grundannahmen hierbei sind, dass eine Anforderung nicht mit sich selbst auf Synergiepotenziale hin untersucht wird und dass, wenn Anforderung A in Bezug auf Anforderung B geprüft wurde, im nächsten Schritt B nicht mehr in Bezug auf A hin überprüft wird.

Bei einem Projekt mit bspw. 100 Anforderungen innerhalb des Product Backlogs wären somit

$$\frac{100 * (100 - 1)}{2} = 4950$$

Vergleiche notwendig, um sicherzustellen, dass alle Potenziale erkannt werden können. Unter der Annahme, dass eine durchschnittliche Bewertung eines Synergiepotenzials 10 Sekunden dauert, so würde das Meeting in diesem Fall 825 Minuten bzw. 13h 45 min dauern. Die Kosten, die durch diese Herangehensweise entstünden, wären so hoch, dass diese durch die Nutzung von identifizierten Synergiepotenzialen kaum ausgeglichen werden könnten oder sich bestenfalls nur eine geringe Ersparnis ergeben würde. Deswegen besteht die Notwendigkeit die Anzahl der benötigten Vergleiche deutlich zu reduzieren, um auch tatsächlich Einsparungen erzielen zu können.

Die Anforderungen sollten im Vorfeld des Meetings durch den Product Owner gefiltert werden und in Gruppen, in denen Synergiepotenziale auftreten können, eingeteilt werden. Je nach Architektur und Aufbau des Systems können nur zwischen bestimmten Anforderungen Synergiepotenziale vorhanden sein. Ein möglicher Ansatzpunkt ist, dass die Anforderungen nach zuvor mit dem Entwicklerteam festgelegten Regeln gruppiert werden. Eine solche Regel könnte bspw. besagen, dass User Stories und ggf. bereits bestehende Product Owner Stories zunächst nach bestimmten, z. B. durch die Softwarearchitektur vorgegebenen, Bereichen untergliedert werden. Im Kontext von Managementinformationssystemen könnten Anforderungen z. B. zunächst in die Bereiche „Datenbeschaffung“, „Datenbewirtschaftung“ und „Datenvisualisierung“ untergliedert werden. Dabei ist darauf zu achten, dass eine Anforderung auch mehreren Bereichen zugeordnet werden kann, da eine User Story aus Sicht des Kunden eine Anforderung beschreibt, die mehrere oder alle diese Bereiche betrifft.

In einem weiteren Schritt könnten dann die Anforderungen aus der Gruppe „Datenbeschaffung“ weiter untergliedert werden. Hierbei würde eine Gruppierung nach Vorsystemen erfolgen, die die benötigten Daten bereitstellen. Eine solche Untergliederung ist insoweit vorteilhaft, da Schnittstellen für unterschiedliche Berichte teilweise Daten aus identischen Vorsystemen verwenden. Somit müssten diese Schnittstellen nicht mehr nachträglich geändert werden oder eine weitere Schnittstelle zum identischen System implementiert werden.

Im Bereich der „Datenvisualisierung“ könnte ebenfalls eine weitere Untergliederung erfolgen, diese würde sich bspw. an der Art der Datendarstellung orientieren. Hierbei ist zu prüfen, ob sich unterschiedliche Anforderungen bzgl. der Visualisierung von Daten mit einem gemeinsam genutzten und zu entwickelnden Template erledigen lassen.

Solche Richtlinien für eine Gruppierung und Untergliederung können jedoch nicht allgemein festgelegt werden. Sie beziehen sich dabei auf ein Projekt in Abhängigkeit der Architektur und der eingesetzten Technologien. Sie sollten somit zu Projektbeginn vom Product Owner in Zusammenarbeit mit dem Entwicklerteam bzw. den Entwicklerteams erarbeitet werden.

Wird davon ausgegangen, dass die zunächst vorhandenen 100 User- bzw. Product Owner Stories durch eine solche Vorklassifizierung in acht unterschiedliche nicht leere Bereiche unterteilt werden können, so ergäbe sich bei einer Gleichverteilung der Produktrückstandselemente, dass

jeder Bereich ca. dreizehn Elemente enthält. Eine weitere Annahme ist, dass ca. 15 % der Anforderungen vom Product Owner direkt aus der Betrachtung entfernt werden können. Gründe hierfür sind, dass wegen der Einhaltung von Abgabefristen eine weitere Betrachtung dieser Anforderung nicht mehr als vorteilhaft erachtet werden kann oder dass diese Anforderung bei der Vorklassifizierung nicht mit anderen Anforderungen in Verbindung gebracht werden konnte. Darüber hinaus wurde bereits festgestellt, dass eine User Story oftmals nicht exklusiv einem Bereich zugeordnet werden kann, deswegen wird davon ausgegangen, dass jede User Story durchschnittlich in 1,5 Bereichen vorhanden ist. Somit würde sich die Anzahl der notwendigen Paarvergleiche wie folgt bestimmen:

- User Stories ohne Potenziale entfernen: $100 * 0{,}85 = 85$
- Unterteilung in nicht disjunkte Mengen:
  - $85 * 1{,}5 \approx 128$
  - $\frac{128}{8} = 16$
- Anzahl der notwendigen Vergleiche als Summe der Vergleiche je Gruppe: $\sum_{i=1}^{n} \frac{(x_i * (x_i - 1))}{2} = 960$

Analog zur Ausgangssituation, in der alle Anforderungen als eine Gruppe vorlagen, kann auch für diese, durch Vorklassifizierung optimierte Bewertung der Zeitaufwand grob approximiert werden. Wird der identische Zeitfaktor je notwendigem Vergleich angewendet, so reduziert sich die benötigte Zeit auf 160 Minuten bzw. auf 2 Stunden und 40 Minuten.

Durch dieses Vorgehen kann der notwendige Ressourceneinsatz, der für die Identifikation von Synergiepotenzialen notwendig ist, verringert werden. Außerdem steigt die Wirtschaftlichkeit des Verfahrens, bzw. kann nur durch eine entsprechende Vorarbeit des Product Owners überhaupt eine entsprechende Wirtschaftlichkeit erzielt werden.

7.4 Zusammenfassung

In diesem Kapitel wurde zunächst erläutert, wie aktuell Anforderungen in Form von User Stories verwaltet werden. Anschließend wurde gezeigt, dass die zurzeit bestehende Hierarchie aus Epics, User Stories und Tasks keine ausreichenden Möglichkeiten bietet, um Synergiepotenziale zwischen Anforderungen abzubilden und in der Entwicklung nutzbar zu

machen. Es wurde gezeigt, dass sich durch die vorhandenen Elemente vorhandene Potenziale zwischen Anforderungen unterschiedlicher Anforderer nur unzureichend abbilden lassen.

Dadurch wurde die Einführung eines neuen Produktrückstandselementes motiviert. Die Verantwortung für dieses Element liegt beim Product Owner, (welcher von der IT gestellt wird). Es liegt in seiner Verantwortung, aus User Stories entsprechende Product Owner Stories abzuleiten und diese innerhalb des Product Backlogs zu verwalten. Obwohl der Product Owner der Verantwortliche ist, so ist er jedoch nicht in der Lage diese Aufgabe vollumfänglich durchzuführen. Ihm fehlt unter Umständen nötiges Detailwissen bzgl. der genauen Implementierung, um eine tatsächliche Bündelung durchführen zu können.

Um diese Notwendigkeit zur Zusammenarbeit auch im Ablauf von Scrum zu reflektieren wurde geprüft, ob die Identifikation von Synergiepotenzialen in einem bestehenden Meeting innerhalb

von Scrum mit abgehandelt werden kann. Es zeigte sich, dass an dieser Stelle die Notwendigkeit besteht, ein neues Meeting in den Ablauf zu integrieren. Innerhalb dieses Meetings soll der Product Owner in Zusammenarbeit mit Mitgliedern des Entwicklerteams entsprechende Synergiepotenziale identifizieren. Im Zusammenhang mit diesem Meeting wurde gezeigt, dass der Product Owner in der Verantwortung ist, eine gewisse Vorgruppierung bzw. Vorklassifizierung durchzuführen, damit das Meeting innerhalb eines begrenzten zeitlichen Rahmens durchgeführt werden kann. Dies ist wichtig, da durch dieses Meeting das Entwicklerteam zusätzlich belastet wird. Es könnte passieren, dass die identifizierten Potenziale nicht ausreichen, um einen tatsächlichen Benefit zu erbringen. Dies würde im schlechtesten Fall durch den mit der Identifikation verbundenen Mehraufwand wieder aufgehoben. Die Einordnung dieses neuen Meetings wird in der folgenden Abbildung in einen theoretischen Ablauf für den Sprint eines Entwicklerteams eingeordnet.

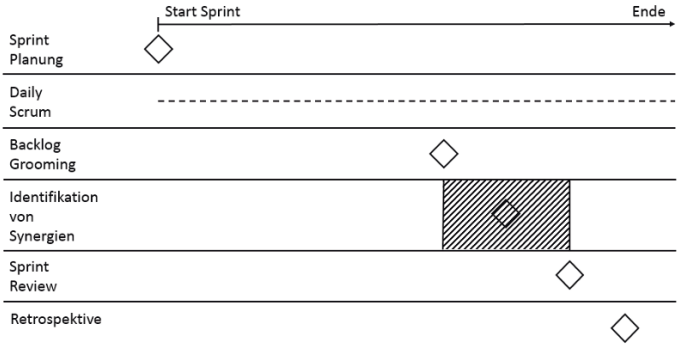


Abbildung 33: Einordnung des Meetings zur Identifikation von Synergien in den Sprintablauf (eigene Darstellung)

Es ist vorteilhaft, dass das neue Synergieidentifikations-Meeting zum Ende des PM-Sprints stattfindet, kurz vor Ende der Iteration. Somit wird sichergestellt, dass die Anzahl an zur Verfügung stehenden Anforderungen, in denen Potenziale erkannt werden können, möglichst „vollständig“ ist und somit Potenziale, welche sich erst innerhalb der weiteren Detaillierung ergeben, mit berücksichtigt werden können.

Die in Kapitel 2.3 identifizierten Forschungsfragen bzgl. der Abbildung und Nutzung von Synergien können hiermit beantwortet werden.

Für die Abbildung von Synergien innerhalb von Scrum ist zum einen ein neues Produktrückstandselement notwendig, welches Synergiepotenziale zwischen User Stories abbildet und aus denen anschließend, scrumtypisch, Tasks erstellt werden. Außerdem ist ein separates Meeting notwendig, um entsprechende Synergiepotenziale zu identifizieren. Die restlichen Abläufe werden durch diese Methodik nur minimal beeinflusst. Als eine mögliche Bedingung für die Nutzung von Synergiepotenzialen wurde die Aufwandsreduktion identifiziert.

Die Forschungsfrage 2.2, welche sich mit den Implikationen durch eine geänderte Betrachtungsweise bei der Sprintplanung befasst, kann ebenfalls beantwortet werden. Product Owner

Stories weisen keinen direkten Nutzwert mehr auf. Dieser ist nur noch für eine Menge von Product Owner Stories transitiv durch die damit verbundenen User Stories zu ermitteln. Dadurch ergeben sich, je nach Zerlegung, unterschiedliche Priorisierungen innerhalb des Product Backlogs (s. Tabelle 12 & Abbildung 28). Eine sich ergebende Implikation ist, dass die Priorisierung innerhalb des Backlogs zeitaufwendiger wird und ein Product Owner somit eine Nutzwertoptimierung nur noch mit einem nicht mehr realistischen Zeitaufwand durchführen kann. Deswegen wird im sich anschließenden Kapitel erläutert, wie die Iterationsplanung teilweise automatisiert werden kann. Das Ziel ist den Product Owner zu entlasten und gleichzeitig die zuvor identifizierten Synergiepotenziale zu nutzen.

8 Bestimmung valider und optimaler Iterationen

Eines der klassischen Ziele in Scrum ist mit jeder Iteration bzw. mit jedem Sprint möglichst viel Nutzwert für den Kunden zu erzeugen. Dieses bedeutet für den Product Owner die Anforderungen zu identifizieren, welche gemeinsam als eine neue, potenziell auslieferbare Version den größten Gesamtnutzen unter Berücksichtigung der aktuellen Gegebenheiten erzeugt. Bei einer größeren Anzahl von User Stories bzw. Product Owner Stories innerhalb des Product Backlogs ist es für den Product Owner nicht mehr möglich, mit einem realistischen Zeitaufwand eine „optimale" Zusammenstellung festzulegen. Bei größeren Projekten, an denen mehr als ein Team gleichzeitig arbeitet, steigt der Aufwand weiter. Hier kommt es zu Situationen, in denen der Product Owner zum „Flaschenhals" für den Entwicklungsprozess wird (vgl. Daut 2013 oder Sobiech et al. 2014).

Das folgende Kapitel wird eine mengentheoretische Beschreibung einer optimalen Iteration erläutern. Durch den Einsatz von Algorithmen sollen Vorschläge für den Product Owner erzeugt werden, um diesen in seiner Arbeit zu entlasten und trotzdem die Potenziale durch die Ausnutzung vorhandener Synergieeffekte sinnvoll nutzen zu können.

8.1 Problemdefinition für die Sprintoptimierung

Bei der Planung von Iterationen mit der Berücksichtigung von Synergieeffekten ergibt sich die Herausforderung, wie in Kapitel 1 erwähnt, dass die auf das Team bzw. die Teams zu verteilenden Elemente keinen direkten Nutzwert mehr aufweisen. Der Nutzwert kann also nur noch indirekt bestimmt werden. Der Nutzwert einer User Story kann erst dann erreicht werden, wenn alle zugehörigen, noch nicht bearbeiteten Product Owner Stories abgeschlossen sind. Demzufolge kann also nur dann der Nutzwert einer User Story erzielt werden, wenn alle zugehörigen, noch zu bearbeitenden Product Owner Stories mit in die Iteration aufgenommen werden. Somit werden für die Nutzwertbestimmung einer Menge von Product Owner Stories mehrere Informationen benötigt. Zum einen muss der Nutzwert der User Stories bekannt sein, zum anderen müssen Informationen darüber vorliegen, welche User Stories und welche Product Owner Stories miteinander in Beziehung stehen.

Mittels des grafischen Beispiels in Abbildung 34 wird die Nutzwertberechnung beispielhaft verdeutlicht.

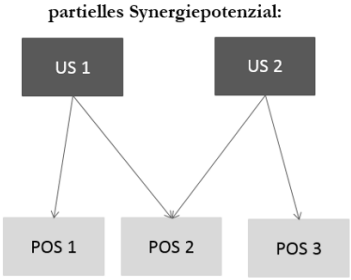


Abbildung 34: Partielles Synergiepotenzial zur Veranschaulichung der Nutzwertberechnung (eigene Darstellung)

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0_8

- US := {US1, US2}, Menge der User Stories
- POS:={POS1, POS2, POS3}, Menge der Product Owner Stories
- Menge K aus Teilmengen von POS, die für jedes u aus US eine Menge K_u enthält, die alle p aus POS enthält, die für die Umsetzung von u notwendig sind
    - K={{POS1, POS2}, {POS2,POS3}}

Durch die Menge K wird somit der Zusammenhang zwischen User Stories und Product Owner Stories beschrieben. Diese Informationen können im nächsten Schritt dazu verwendet werden, den Nutzwert für eine Menge von Product Owner Stories zu berechnen. Der Nutzwert für eine Menge P ⊆ POS lässt sich wie folgt bestimmen:

$$\sum_{u \in US} Nutzwert(u) * x(u)$$

$$\text{mit } x(u) = \begin{cases} 1 \; wenn \; K_u \subseteq P \\ 0 \; sonst \end{cases}$$

Für das Sprint-Optimierungsproblem wird also für jedes Entwicklerteam d ∈ D eine Teilmenge an Product Owner Stories ($P_d \subseteq$ POS) gesucht, sodass der gemeinsame Nutzwert ($P = \bigcup_{d \in D} Pd$) entsprechend der genannten Formel maximiert wird. Eine Iteration kann dann als optimal angesehen werden, wenn es keine andere Zuteilung von Product Owner Stories auf die vorhandenen Teams gibt, die einen höheren Nutzwert erzeugt. Hierbei ist zu beachten, dass es je nach Problemstellung mehr als eine optimale Lösung geben kann.

Ohne weitere Einschränkungen, wäre also die Abarbeitung aller Product Owner Stories innerhalb der nächsten Iteration immer die optimale Lösung, da somit alle User Stories implementiert werden und der komplett vorhandene Nutzwert erzielt werden würde. Dieses ist jedoch nicht realistisch, da weitere Beschränkungen und Einschränkungen gemacht werden müssen, um sicherzustellen, dass die angestrebten Ziele auch eingehalten werden können.

Bei der Zuteilung der Arbeitspakete in Form von Product Owner Stories auf Teams ist darauf zu achten, dass das Team diese Aufgabe erledigen kann. Dies beinhaltet zwei Aspekte. Zum einen muss sichergestellt werden, dass das Team ausreichende technische Fähigkeiten aufweist, um die jeweilige Product Owner Story, mit den ihr zugeordneten Aufgaben, erledigen zu können. Zum anderen ist z. B. auf Grundlage der Menge an abgeschlossenen Product Owner Stories aus den letzten Sprints zu prüfen, ob das Team zum aktuellen Zeitpunkt noch die nötige Restkapazität besitzt, um die Aufgabe innerhalb der nächsten Iteration zu bewältigen.

Nach der Zuteilung aller Product Owner Stories muss neben der Nutzwertoptimierung auch noch sichergestellt werden, dass alle User Stories, die eine Deadline aufweisen, deren Ablauf vor dem Ende der nächsten Iteration liegt, vollständig innerhalb dieser Iteration implementiert werden. Da die Deadline, analog zum Nutzwert, nur aus Sicht des Kunden definiert ist und nicht auf Ebene der Product Owner Stories, ergibt sich die Fertigstellungsfrist als minimale Frist aller Fristen auf User Story Ebene, die mit der jeweiligen Product Owner Story assoziiert sind.

Neben dieser Tatsache ist auch zu prüfen, ob bestehende Abhängigkeiten zwischen den Product Owner Stories beachtet sind. Hierbei ist sicherzustellen, dass keine Product Owner Story in eine Iteration eingeplant und einem Team zugewiesen wird, wenn die zuvor benötigte Grundlage, welche z. B. durch eine andere Product Owner Story geschaffen wird, nicht vorhanden ist. Deswegen müssen also Product Owner Stories, welche vorbereitende Aufgaben für eine andere

Product Owner Story mit beinhalten, auch mit in diese Iteration aufgenommen werden oder die aufbauenden Tätigkeiten müssen auf eine spätere Iteration verschoben werden.

Ein Sprint-Problem, wie es in diesem Kapitel zunächst in natürlicher Sprache modelliert und definiert wurde, kann somit wie folgt beschrieben werden:

Definition:

Ein Sprintproblem (U, P, K, D, CD, n, a, d, k, G, dl) hat als Eingaben:

- eine Menge U, als Menge der User Stories
- eine Menge P, als Menge der Product Owner Stories
- eine Menge K aus Teilmengen von P, die für jede $u \in U$ eine Menge K_u enthält, die alle $p \in P$ enthält, die für die Umsetzung von u notwendig sind
- eine Menge D, als Menge der Entwicklerteams
- eine Menge CD aus Teilmengen von P, die für jede $d \in D$ eine Menge CD_d enthält, die alle $p \in P$ enthält, die Team d bearbeiten kann
- eine Funktion n, die jeder $u \in U$ einen Nutzwert zuordnet
- eine Funktion a, die jeder $p \in P$ einen Aufwand zuordnet
- einer Funktion d, die jeder $p \in P$ eine Deadline zuordnet
- einer Funktion k, die jedem $d \in D$ die Kapazität in der nächsten Iteration zuordnet
- einen Abhängigkeitsgraphen G = (V, E), mit V = P und E ={(p, p‘) | p, p‘∈ V und p‘ ist abhängig von p}
- ein Datum dl, welches das Ende des nachfolgenden Sprints angibt

Das Ziel ist, eine Menge $P_d \subseteq P$ für jedes Entwicklerteam $d \in D$ zu finden, so dass

- $P_d \subseteq CD_d$, als Ausdruck der Möglichkeit, dass das entsprechende Team die ihm zugedachten Aufgaben bearbeiten kann
- $\sum_{p \in Pd} a(p) \leq k(d)$, die Summe der Aufwände liegt innerhalb der Kapazität des Teams
- Es gilt für alle d, d‘ mit d $\neq d'$ dass $P_d \cap P_{d'} = \emptyset$, keine Product Owner Story wird mehrfach von unterschiedlichen Teams umgesetzt
- $(p' \in \bigcup_{d \in D} P_d) \wedge (\,(p, p') \in V) \rightarrow$ p $\in \bigcup_{d \in D} P_d$, alle Vorbedingungen sind ebenfalls in dieser Iteration eingeplant
- $\neg \exists\, p((p \in P) \wedge (\mathrm{d(p)} < \mathrm{dl}\,) \wedge (p \notin \bigcup_{d \in D} Pd))$, es sind in der Sprintplanung alle Product Owner Stories enthalten, deren Abgabefrist vor der des nachfolgenden Sprints liegen
- $\sum_{u \in U} n(u) * x(u)$ wird maximiert

  mit $x(u) = \begin{cases} 1\ wenn\ K_u \subseteq \bigcup_{d \in D} P_d \\ 0\ sonst \end{cases}$

Da das Konzept des Nutzwertes nur aus der Sicht des Kunden definiert ist und somit nur für User Stories und nicht für Product Owner Stories existiert, wird der Nutzwert einer kompletten Iteration für alle Teams über die Summe der Nutzwerte der komplett implementierten User Stories definiert. Diese ergeben sich durch die vollständig umgesetzten Product Owner Stories, welche in den unterschiedlichen Teams bearbeitet werden.

Durch diese Definition des Nutzwertes wird implizit auch davon ausgegangen, dass User Stories, die nur teilweise implementiert sind, keinen Nutzwert für den Kunden besitzen oder nicht bereitgestellt werden können.

8.2 Komplexitätsbetrachtung für das Sprintproblem

Ein wichtiger Aspekt von Optimierungs-, wie auch Entscheidungsproblemen ist die Untersuchung ihrer Komplexität. Je nach Komplexitätsklasse kann entsprechend entschieden werden, ob ein Algorithmus existiert, der das Problem mit einem realistischen Zeitaufwand optimal lösen kann. Wenn kein solcher Algorithmus aufgrund der theoretischen Untersuchungen existieren kann[7], müssen alternative Lösungsmöglichkeiten evaluiert werden.

Ein möglicher Weg, um die Komplexität eines gegebenen Problems näherungsweise zu bestimmen, ist die Reduktion eines Problems, dessen Komplexität bereits bekannt ist, auf das zu untersuchende Problem (Hopcroft et al. 2002; Wegener 2003). Somit kann sichergestellt werden, dass das zu untersuchende Problem mindestens genauso schwer ist wie das Problem, welches auf das zu untersuchende Problem reduziert wurde.

Für die Reduktion wird das Rucksackproblem (engl. „Knapsack Problem" (KP)) verwendet (s. Kapitel 4.3.2 bzw. Kapitel 12.2). Das Rucksackproblem ist eins von Karps 21 klassischen NP-vollständigen Problemen (Karp 1972). Im Folgenden wird dieses Problem auf das Sprintproblem reduziert, um zu zeigen, dass es sich bei diesem Problem auch um ein NP-schweres Problem handelt.

Das Rucksackproblem existiert sowohl als Optimierungsproblem als auch als Entscheidungsproblem. Beim Optimierungsproblem ist es das Ziel, eine Menge von Gegenständen oder allgemeinen Objekten auszuwählen und in einen Rucksack zu legen, sodass der Wert dieser ausgewählten Teilmenge maximiert wird. Die Beschränkung für die Auswahl der Objekte stellt die Gewichtsobergrenze des Rucksacks dar. Dieser darf nicht überladen werden, da sonst der Rucksack reißen könnte oder vom Träger nicht mehr getragen werden kann.

Bei der Entscheidungsvariante des Rucksackproblems wird gefragt, ob es eine Belegung des Rucksacks gibt, deren Gewicht unterhalb der maximalen Gewichtsschranke liegt und eine gegebene Nutzwertschranke erreicht oder übersteigt. Diese Variante gehört zu den 21 NP-vollständigen Problemen, von denen Richard Karp 1972 die Zugehörigkeit zu dieser Klasse zeigen konnte. (Dem Anhang kann eine detailliertere Formulierung des Rucksackproblems, wie auch die Definition unterschiedlicher Variationen des Problems, entnommen werden (s. Kapitel 12.2).

[7] Außer P = NP

Sei (I, v‘, w‘, c‘) eine Instanz des Rucksackproblems. I stellt hierbei die Menge der Objekte oder Gegenstände dar. v‘ ist in diesem Zusammenhang eine Funktion, welche jedem Gegenstand $i \in I$ einen Wert zuordnet. Die Funktion w‘ ordnet jedem $i \in I$ ein Gewicht zu. Außerdem sei c‘ die maximale Kapazität des Rucksacks. Das Ziel ist somit eine Teilmenge $R \subseteq I$ zu finden, sodass:

1. $\sum_{i \in R} w'(i) \leq c'$, Gewichtsgrenze wird eingehalten
2. $\sum_{i \in R} v'(i)$ wird maximiert

Im nächsten Schritt wird die Eingabe des Rucksackproblems zu einer Eingabe in das Sprintproblem transformiert. Hierzu sei zunächst $D = \{d\}$, außerdem wird $k(d) = c'$ gesetzt. Somit wird sichergestellt, dass das Entwicklerteam d die identische Kapazität im nächsten Sprint besitzt, wie der Rucksack aus der Probleminstanz des Rucksackproblems. Außerdem sei $I = U = P = CD_d$. Gleichermaßen wird für jedes $u \in U$ eine Menge $M_u = \{u\}$ erzeugt. Dieses ist möglich, da $U = P$. Ebenso sei definiert, das $a(u) = w'(u)$ und $n(u) = v'(u)$. Durch diese Definition wird sichergestellt, dass jedes Objekt aus der initialen Probleminstanz des Rucksackproblems genau auf ein Paar von User Story und Product Owner Story abgebildet wird, bei denen die User Story den entsprechenden Nutzwert des Objektes erhält. Die zu dieser User Story assoziierte Product Owner Story erhält den korrespondierenden Aufwand bzw. das zugehörige Gewicht. Dies ist ebenfalls nur im Spezialfall von $I = U = P$ möglich. Dieses wird in Abbildung 35 verdeutlicht.

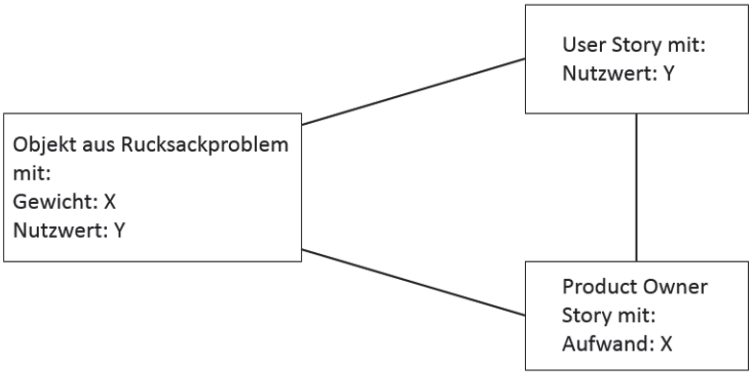


Abbildung 35: Zerlegung von Objekten aus KP in User Story und Product Owner Story (eigene Darstellung)

Außerdem sei der Abhängigkeitsgraph G als Graph ohne Kanten definiert, da Abhängigkeiten in der klassischen Variante des Rucksackproblems keine Berücksichtigung finden. Somit sei $G = (P, \emptyset)$. Darüber hinaus kann für das Datum dl ein beliebiges Datum gewählt werden, bei der Erzeugung der User Stories ist dann lediglich darauf zu achten, dass alle User Stories eine

Deadline nach diesem Datum erhalten. Eine mögliche Wahl für die Funktion d ist somit $\forall u \in U: d(u) = dl + 1\,Tag$[8]

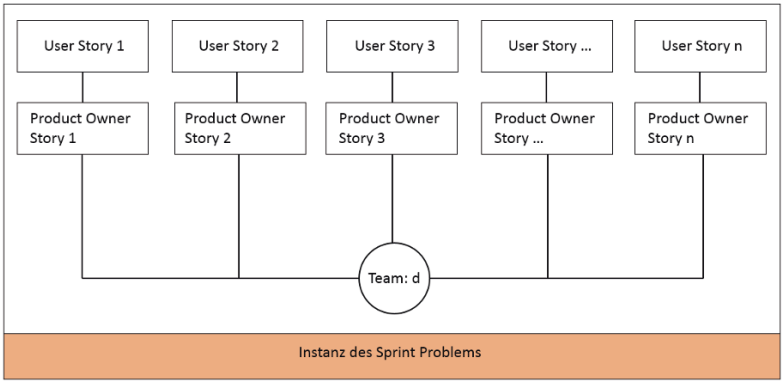


Abbildung 36: Alternative Repräsentation der Instanz des Sprint Problems (eigene Darstellung)

Durch das Lösen dieser Probleminstanz des Sprint-Optimierungs-Problems wird offensichtlich auch die Probleminstanz des Rucksackproblems gelöst. Dies ist begründet in der Tatsache, dass das Entwicklerteam d als Rucksack interpretiert werden kann, da es die gleichen kapazipativen Eigenschaften aufweist. Da die so erzeugte Instanz immer nur ein Team besitzt, ist sichergestellt, dass die dritte Bedingung immer erfüllt ist. Dieses gilt ebenfalls für die erste Einschränkung, da $CD_d = I$. Dieses kann so interpretiert werden, dass das Team in der Lage ist alle Product Owner Stories zu bearbeiten, wie auch der Rucksack tendenziell zunächst jeden Gegenstand beherbergen kann. Aufgrund der besonderen Konstruktion dieser Probleminstanz gilt, dass ein optimales $P_d \subseteq P$ auch eine Teilmenge von U sein muss, da $P = U$. Außerdem muss $P_d \subseteq I$ gelten, da $U = I$. Deswegen wird, wenn und nur wenn $u \in P_d$, der Aufwand aufsummiert und mit der Gewichts- bzw. Aufwandsgrenze, welche identisch sind, verglichen. Gleiches gilt für den Nutzwert der zugehörigen User Story, welcher ebenfalls identisch ist mit dem korrespondierenden Objekt aus der Instanz des Rucksackproblems, da, wenn $u \in P_d$ ist, dann muss u auch ein Element in U sein. Weil $M_u = \{u\}$ und $u \in I$ wird also ein Paar aus User Story und Product Owner Story identisch zu einem Objekt aus dem Rucksackproblem behandelt. Wegen dieser Tatsache gilt auch, dass ein optimales P_d auch automatisch die optimale Lösung für die Probleminstanz des Rucksackproblems darstellt.

Dadurch ist gezeigt, dass das hier formulierte Sprint-Optimierungsproblem mindestens so schwer ist wie das Rucksackproblem. Außerdem ist eine Abbildung von einer Instanz des Rucksackproblems auf eine Instanz des Sprintproblems in polynomiellI beschränkter Zeit möglich. Dadurch ergibt sich folgendes Theorem:

Theorem: Das Sprint-Optimierungsproblem ist NP-schwer, auch im Fall von $|D| = 1$.

[8] Eine äquivalente Formulierung in SQL wäre : DATEADD(day,1,@dl)

Durch die Feststellung, dass es sich bei dem Sprint-Optimierungsproblem um ein NP-schweres Problem in der Optimierungsvariante handelt, bzw. um ein NP-vollständiges in der Entscheidungsvariante, muss davon ausgegangen werden, dass das Problem vor allem für größere Probleminstanzen nicht mehr effizient und optimal durch einen Algorithmus gelöst werden kann.

Auf Grund dieser Tatsache werden in den folgenden Kapiteln unterschiedliche Methoden diskutiert, um dennoch eine zeiteffiziente Lösung und Berücksichtigung der Ergebnisqualität zu erzeugen.

8.3 Verfahren für die Vorschlagsgenerierung

In diesem Kapitel werden unterschiedliche Strategien für die Vorschlagsgenerierung zur Lösung des zuvor vorgestellten Sprint-Optimierungsproblems erarbeitet. Der Fokus wird dabei sowohl auf heuristischen Verfahren, als auch auf naturanalogen Verfahren liegen. Diese beiden unterschiedlichen Strategien haben für andere Probleme dieser Komplexitätsklasse gezeigt, dass sie gute Näherungslösungen erzeugen können.

Aus den theoretischen Vorüberlegungen im vorherigen Abschnitt wurde bereits gefolgert, dass größere Probleminstanzen vermutlich nicht mehr zeiteffizient optimal gelöst werden können. In Abbildung 37 wird der Zeitaufwand für eine vollständige Prüfung aller Möglichkeiten in Bezug auf die Problemgröße dargestellt. Es zeigt sich, dass der benötigte Zeitaufwand bei zwei komplett cross-funktionalen Teams und lediglich zehn Product Owner Stories gering ist. Für eine komplette Prüfung aller Möglichkeiten werden deutlich weniger als 30 Sekunden benötigt[9]. Bei einer Problemgröße von 13 Product Owner Stories braucht eine Prüfung aller Varianten bereits ca. 10 Minuten.

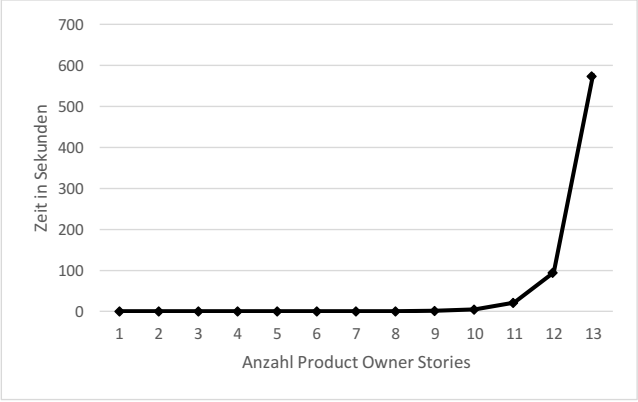


Abbildung 37: Zusammenhang zwischen Problemgröße und Zeitaufwand (2 Entwicklerteams) (Quelle: Sobiech et al. 2015)

[9] Windows 8.1, Intel Core i7-3770 @ 3.4 GHz, 16 GB Arbeitsspeicher, Algorithmus in C# implementiert

Bei typischen Problemgrößen von ca. 30 bis 100 Product Owner Stories bei größeren Entwicklungsprojekten mit einer Vielzahl an Anforderern muss davon ausgegangen werden, dass eine erschöpfende Suche zur Problemlösung ausscheidet (Brute-Force-Methode).

Deswegen werden im Folgenden Lösungsmöglichkeiten aufgezeigt und hinsichtlich der Ergebnisqualität und des Zeitaufwandes bewertet.

Diesen Untersuchungen sei vorweggestellt, dass die Verwendung des Nemhauser/Ullman-Algorithmus, welcher in Kapitel 4.3.2.3 bereits vorgestellt wurde, für die Lösung des Problems ebenfalls nicht in Betracht kommt. Durch theoretische Überlegungen können schnell Beispiele gefunden werden, bei denen zunächst dominierte Lösungen abschließend doch zu optimalen Ergebnissen führen können. Dieses ist im Wesentlichen in der Tatsache begründet, dass durch die Verwendung von Product Owner Stories keine direkte Beziehung zwischen den einzelnen Product Owner Stories und Nutzwerten existiert, sondern dieser nur für eine Menge von Product Owner Stories definiert ist.

8.3.1 Erzeugung von Testfällen zur Konfiguration und Bewertung

In diesem Abschnitt wird dargestellt, wie die in den nachfolgenden Kapiteln verwendeten Test- und Konfigurationsinstanzen erzeugt werden. Für diesen Zweck wurde ein Programm zur Generierung entwickelt. Als Ergebnis eines Durchlaufs erzeugt das Programm eine Datei, deren Inhalt dem definierten Input für das Sprintproblem entspricht. Nachfolgend wird eine solche Datei beispielhaft gezeigt (s. Abbildung 38).

```
a       5       N
b       9       Y
---------
x       2
y       5
z       3
---------
a       x,y
b       y,z
--------

--------
t1      12      x,y,z
t2      12      x,y,z
```

Abbildung 38: Beispieldatei für das Iterationsproblem

In den obersten beiden Zeilen werden zunächst die vorhandenen User Stories angegeben. Jede Zeile entspricht dabei einer User Story. Die ersten Zeichen geben den Namen der User Story an, in diesem Fall „a“ und „b“. Dem Namen folgt ein Tabulator-Zeichen. Anschließend wird der Nutzwert der User Story angegeben, gefolgt von einem weiteren Tabulatorzeichen und einem Indikator. Dieser zeigt an, ob diese User Story bei der folgenden Planung als eine User Story betrachtet werden soll, deren Abgabefrist bei Nichteinplanung verletzt werden würde. Bei der Erzeugung der Instanzen wurde eine zufällige ganze Zahle zwischen 1 und 10 für jede User Story gewählt. Die Wahrscheinlichkeit, dass eine User Story mit Abgabefrist erzeugt wird, beträgt 10 %.

Im zweiten Abschnitt der Datei sind die Product Owner Stories definiert. Analog zum ersten Abschnitt wird zunächst der Name der Product Owner Story angegeben, gefolgt von einem

Tabulatorzeichen. Anschließend wird der Aufwand genannt. Für die Erstellung dieser Testfälle wurde der Aufwand für jede Product Owner Story als zufälliges Element aus der Fibonacci-Folge im Intervall von 1-13 bestimmt (n=2 bis n=7).

Anschließend wird für jede User Story definiert, welche Product Owner Stories umzusetzen sind. Sollte mehr als eine Product Owner Story notwendig sein, so werden diese durch ein Komma voneinander getrennt. Bei der Erzeugung wird zunächst jeder User Story eine zufällige Product Owner Story zugewiesen. Mit einer Wahrscheinlichkeit von 30 % wird einer User Story eine weitere Product Owner Story zugeordnet. Dieser Vorgang wird im Erfolgsfall ein weiteres Mal wiederholt, sodass einer User Story maximal drei Product Owner Stories zugewiesen werden können.

Im nächsten, hier leeren Abschnitt werden ggf. vorhandene Abhängigkeiten eingetragen. Dieses geschieht in der Form:

1. Name der Product Owner Story
2. Tabulatorzeichen
3. Name der Product Owner Story die Vorbedingung zur Abarbeitung ist

Die Wahrscheinlichkeit für jede Product Owner Story beträgt 10 %. Als Vorbedingung wird eine zufällige vorhandene Product Owner Story gewählt, die ungleich sich selbst ist.

Im letzten Abschnitt werden die Entwicklerteams beschrieben. Der Satzaufbau ist: Name des Teams, Tabulatorzeichen, Kapazität des Teams in Storypunkten, Tabulatorzeichen, Liste der Product Owner Stories, die bearbeitet werden können.

Für die Test- und Konfigurationsfälle wurden zwei Teams verwendet. Als Kapazität wird ein Wert von 30 Storypunkten für beide Teams angenommen. Im Fall von nicht cross-funktionalen Teams wird die erste Hälfte der vorhandenen Product Owner Stories dem ersten Team zugeordnet, die verbleibenden Stories dem zweiten Entwicklerteam.

Nach der automatisierten Erstellung der Test- und Konfigurationsinstanzen wurden alle Instanzen manuell kontrolliert, ob sie eine triviale Lösung erlauben. Dieser Fall liegt vor, wenn die optimale Lösung durch das Hinzufügen aller Product Owner Stories erreicht wird, ohne dass die Rahmenbedingung bzgl. der Teamkapazitäten verletzt wird.

Die durch dieses Vorgehen erzeugten Instanzen werden in den folgenden Abschnitten verwendet. Für die Konfiguration wurden andere Instanzen verwendet, als für die spätere Auswertung.

8.3.2 Heuristische Lösungsverfahren

Ein Lösungsverfahren für ein Problem wird als Heuristik bezeichnet, wenn mit unvollständigen Informationen und geringem Zeitaufwand eine Lösung generiert wird (Gigerenzer und Todd 1999). Heuristische Algorithmen verwenden dabei sogenannte „Daumenregeln“, um für viele praktisch relevante Problemstellungen brauchbare Lösungen innerhalb eines vertretbaren Zeitaufwandes zu generieren (Gumm und Sommer 2006).

Die Verwendung von Heuristiken für die Lösung eines solchen NP-schweren Problems ist damit begründet, dass diese aller Vermutung nach nicht mit einem Algorithmus in polynomieller Zeit gelöst werden können. Deswegen leisten Heuristiken die gute, nah optimale Lösungen erzeugen einen wichtigen Beitrag zur Lösung von Problemen dieser Komplexitätsklasse (Martello und Toth 1981).

In den nachfolgenden Unterkapiteln werden zunächst unterschiedliche heuristische Algorithmen erläutert. Anschließend werden die unterschiedlichen Algorithmen in Bezug auf ihre Leistungsfähigkeit hin verglichen. Abschließend werden die Ergebnisse dieser Evaluation gegenübergestellt.

8.3.2.1 First Fit Heuristik

Der heuristische Ansatz, der in diesem Abschnitt vorgestellt wird, verwendet einen Greedy-Algorithmus. Diese Algorithmen zeichnen sich dadurch aus, dass in jeder Iteration bzw. in jedem Schritt der Zustand angestrebt wird, der zum Zeitpunkt der Wahl das beste Ergebnis verspricht. Das Grundschema wird in diesem Abschnitt ebenfalls genauer erläutert und auch für die folgenden beiden heuristischen Algorithmen verwendet. Deswegen werden in diesen Abschnitten nur noch die Abweichungen zu dem hier vorgestellten Schema dargestellt.

Das Sprint-Problem hat, wie beschrieben, fünf einschränkende Bedingungen, die bestimmen ob es sich bei einer Lösung um eine valide Lösung handelt. Folgende Bedingungen müssen erfüllt sein:

1. Es müssen alle Owner Stories eingeplant werden, deren Abgabefrist verletzt werden würde
2. Bei mehreren Teams sollen einem Team nur Aufgaben zugewiesen werden, die dieses auch bearbeiten kann (notwendige Kompetenzen sind vorhanden)
3. Kein Team wird überlastet. Die Summe der Aufwände für die Umsetzung der Product Owner Stories ist ≤ der Teamkapazität
4. Besitzt eine Product Owner Story Vorbedingungen, so müssen auch diese erfüllt sein oder innerhalb des Sprints erfüllt werden
5. Bei mehreren Teams soll keine Product Owner Story durch zwei oder mehr unterschiedliche Teams gleichzeitig umgesetzt werden

Um diese Rahmenbedingungen einzuhalten empfiehlt sich ein zweistufiger Ablauf innerhalb des Algorithmus. In einer ersten Phase werden zunächst alle Product Owner Stories eingeplant, deren Abgabefrist sonst verletzt werden würde. Erst in einem weiteren Schritt ist es möglich, weitere Product Owner Stories einzuplanen und den Nutzwert durch diese zu steigern.

Nachdem alle benötigten Variablen initialisiert sind, muss also in der ersten Phase für jede Product Owner Story geprüft werden, ob die Abgabefrist verletzt wird, wenn diese nicht innerhalb dieser Iteration eingeplant werden würde. Ist dies der Fall, so muss diese Product Owner Story einem der vorhandenen Entwicklerteams zugeordnet werden. Nachfolgendes Quellcodebeispiel stellt diesen Ablauf dar:

```
…
for p in P:
       if date(p) < dl:
              doFirstFitAssignment(p)
…
```

Die eigentliche Allokation der Product Owner Story zu einem Team wird dabei in der Methode „doFirstFitAssignment(ProductOwnerStory p)“ durchgeführt. Innerhalb dieser Methode wird

jedoch nicht ausschließlich eine Zuweisung durchgeführt. Es wird ebenfalls geprüft, ob entsprechende Rahmenbedingungen eingehalten werden. Innerhalb dieser Funktion ergibt sich folgender Ablauf:

1) Rekursive Bestimmung aller ggf. vorhandenen Vorbedingungen zur Product Owner Story p und der vorhandenen Vorbedingungen der Vorbedingungen usw.
2) Zuweisung jeder so bestimmten Product Owner Story zu einem Team
   a) Prüfe nacheinander für jedes Team d, ob p von diesem Team bearbeitet werden kann, wenn nicht, prüfe nächstes Team, sonst weiter mit b)
   b) Prüfe, ob die verbleibende Kapazität des Teams d ≤ a(p) ist. Wenn ja, plane Product Owner Story p für Team d ein und gehe zu 2, wenn nein, gehe zu a
3) Sind alle Product Owner Stories einem Team zugewiesen, entferne jede dieser Product Owner Stories aus allen Mengen K_u; andernfalls beende den Algorithmus

Im ersten Schritt wird zunächst sichergestellt, dass die Bedingung bzgl. der Vorbedingungen eingehalten wird. Hierbei ist zu beachten, dass sowohl die Vorbedingungen der Product Owner Story, die eingeplant werden soll, beachtet werden, als auch etwaige vorhandene Vorbedingungen der Vorbedingungen. In diesem Schritt wird somit eine Menge an Product Owner Stories erzeugt, die sicherstellt, dass diese Bedingung erfüllt wird, wenn alle Product Owner Stories einem Team zugewiesen werden.

Im nächsten Schritt (2) wird für jede der zuvor ermittelten Product Owner Stories ein Team bestimmt, dass diese Product Owner Story bearbeiten kann. Bei diesem einfach heuristischen Algorithmus werden zunächst keine weiteren Rahmenparameter betrachtet. Es wird das erste Team ausgewählt, dass die jeweilige Product Owner Story bearbeiten kann und eine ausreichende verbleibende Kapazität aufweist. Dieser Mechanismus wird auch „First Fit“ genannt, da das erste passende Element verwendet wird, in diesem Fall das erste Team, das die Product Owner Story bearbeiten kann. Durch die einmalige Zuordnung und dem anschließenden Übergang zur nächsten Product Owner Story wird an dieser Stelle auch verhindert, dass eine Product Owner Story von mehr als einem Team umgesetzt wird.

Im letzten Schritt wird geprüft, ob alle Product Owner Stories einem Team zugewiesen werden konnten. Ist dies nicht Fall, so beendet der Algorithmus und gibt eine Fehlermeldung aus. Dies ist insofern notwendig, als dass in diesem Fall kein valider Sprint mehr entstehen kann und eine weitere Verarbeitung der Daten keine weitere Relevanz mehr besitzt. Im Fall einer erfolgreichen Allokation aller Product Owner Stories werden diese im letzten Schritt der ersten Phase dieses heuristischen Algorithmus aus jeder Menge K_u entfernt. Dies ist eine notwendige Vorbereitung für die nächste Phase, denn der Aufwand einer User Story u ist bestimmt durch die Summe der Aufwände der assoziierten Product Owner Stories in der Menge K_u. Dadurch, dass bereits eine gewisse Menge an Product Owner Stories alloziert ist, spielen diese für die Erzielung von Nutzwerten durch User Stories keine weitere Rolle mehr und müssen aus der Betrachtung ausgeschlossen werden. Dieses wird durch folgendes Beispiel, welches analog zum Beispiel aus Kapitel 8.1 ist, veranschaulicht.

- US := {US1, US2, US4}
- POS:={POS1, POS2, POS3, POS4}
- Menge K aus Teilmengen von POS, die für jedes u aus US eine Menge K_u enthält, die alle p aus POS enthält, die für die Umsetzung von u notwendig sind
  - K={$_{US1}${POS1, POS2}, $_{US2}${POS2,POS3}, $_{US4}${POS4}}

- Es gelten die folgenden Abgabefristen:
  - d(US 1) ist kleiner als das Enddatum des nachfolgenden Sprints
  - d(US 2) ist größer als das Enddatum des nachfolgenden Sprints
  - d(US 4) ist größer als das Enddatum des nachfolgenden Sprints

Zwecks einer besseren Nachvollziehbarkeit wird in diesem Beispiel davon ausgegangen, dass es genau ein Team t gibt, welches die notwendigen Kompetenzen besitzt, alle Product Owner Stories zu bearbeiten. Dieses Beispiel ist in der nachfolgenden Abbildung 39 veranschaulicht.

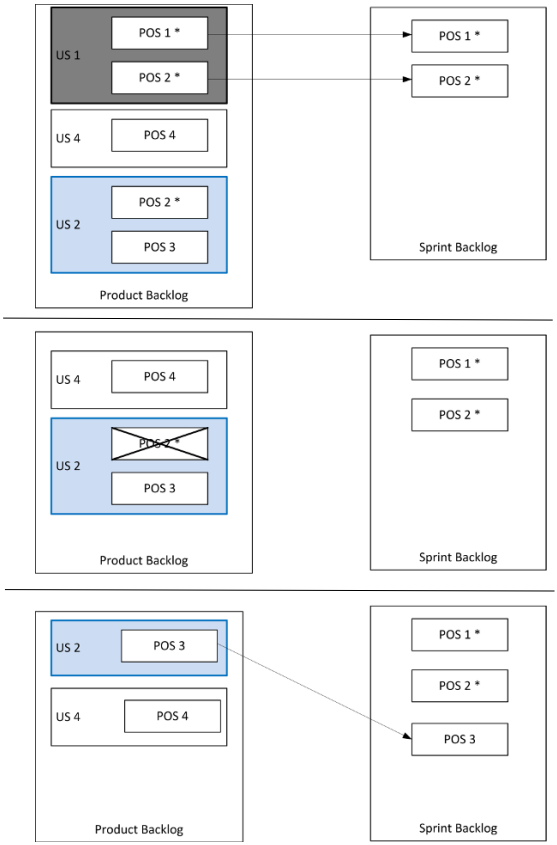


Abbildung 39: Ablauf der Allokation von Product Owner Stories mit Abgabefrist (eigene Darstellung)

Im ersten Schritt wird dargestellt, dass die Product Owner Stories, welche zur Umsetzung von US 1 notwendig sind, dem Sprint Backlog hinzugefügt werden. Dieser Schritt dient nur zur Sicherstellung von Rahmenbedingungen, jedoch nicht zur eigentlichen Nutzwertoptimierung.

Im zweiten Schritt werden alle Product Owner Stories, welche bereits eingeplant sind, aus dem Product Backlog entfernt, um sicherzustellen, dass keine Product Owner Story doppelt bearbeitet wird. Anschließend ist es notwendig, das Product Backlog erneut nach dem Verhältnis von Nutzen zu Aufwand zu sortieren. Der Nutzwert einer jeweiligen User Story bleibt unverändert, jedoch kann sich der noch erforderliche Aufwand ändern, da der Aufwand einer User Story durch die Aufwände der assoziierten Product Owner Stories bestimmt wird. Durch Allokationen im ersten Schritt kann es an dieser Stelle zu Verschiebungen im Backlog kommen.

Diese Repriorisierung wird im unteren Teil der Abbildung dargestellt. US 2 befindet sich nach der Repriorisierung des Backlogs vor US 4, weil die Product Owner Story POS 2 bereits im ersten Schritt aufgrund der Abgabefrist von US 1 eingeplant wurde. Deswegen ist diese Product Owner Story nicht mehr notwendig für die Betrachtung des verbleibenden Aufwandes von User Story US 2. Diesem Umstand wird durch die Repriorisierung Rechnung getragen. Daher besteht nicht die Notwendigkeit, dass bei der Eingabe des Backlogs bereits eine sortierte Liste vorliegt, da sich die Sortierung durch jede Allokation einer Product Owner Story ändern kann. Sind innerhalb des Backlogs User Stories bzw. Product Owner Stories vorhanden, welche eine Abgabefrist besitzen, so werden diese bekanntlich außerhalb der eigentlichen Priorisierung, welche auf dem Verhältnis von Aufwand und Nutzen basiert, in eine Iteration eingeplant.

Innerhalb des Abschnitts des Algorithmus, welcher die eigentliche „Optimierung“ vollzieht, werden zwei Annahmen getroffen, die zur Verringerung der Komplexität angewendet werden. Zum einen wird davon ausgegangen, dass die User Story, die die größte Nutzwertdichte aufweist, die optimale User Story ist, um weiter zu verfahren. Zum anderen wird unterstellt, dass die Zuteilung zu einem Team keine Auswirkungen hat, sodass das erste Team für die Allokation einer Product Owner Story verwendet wird, dass die entsprechenden Kompetenzen besitzt.

Der grobe Ablauf wird im folgenden Quellcodebeispiel dargestellt:

```
uList.sort(key = lambda t:dense(t), reverse = True)
while (uList):
    u = uList[0]
    doFirstFitAssignment(u)
    uList.remove(u)
    uList.sort(key = lambda t:dense(t), reverse = True)
```

Wie zu ersehen ist, wird zunächst die Liste der User Stories entsprechend der Dichte sortiert. Anschließend wird in einer Schleife jeweils die User Story mit der größten Dichte aus der Liste entfernt. Anschließend werden innerhalb der Funktion „doFirstFitAssignment(UserStory u)“ die folgenden Schritte ausgeführt:

1) Bestimmung aller notwendigen Product Owner Stories, welche für die Umsetzung dieser User Story notwendig sind
2) Rekursive Bestimmung aller ggf. vorhandenen Vorbedingungen zu allen in 1) ermittelten Product Owner Stories und der vorhandenen Vorbedingungen der Vorbedingungen
3) Zuweisung jeder so bestimmten Product Owner Story p zu einem Team
    a. Prüfe nacheinander für jedes Team d, ob p von diesem Team bearbeitet werden kann, wenn nicht, prüfe nächstes Team, sonst weiter mit b)
    b. Prüfe, ob die verbleibende Kapazität des Teams d ≤ a(p) ist. Wenn ja, plane Product Owner Story p für Team d ein und gehe zu 2, wenn nein, gehe zu a
4) Sind alle Product Owner Stories einem Team zugewiesen, entferne jede dieser Product Owner Stories aus allen Mengen K_u; andernfalls alle Zuweisungen aus 3) rückgängig machen

Unabhängig davon, ob die Zuweisung erfolgreich war oder nicht, wird im nächsten Schritt diese User Story aus der Liste der User Stories entfernt. Danach wird innerhalb dieser Schleife die Liste neu sortiert.

Dieser Ablauf wiederholt sich solange, bis alle User Stories entsprechend diesem Vorgehen geprüft wurden. Als Ergebnis ergibt sich für jedes Team eine mögliche Menge an Product Owner Stories für den nächsten Sprint. Über die qualitativen Eigenschaften dieses heuristischen Algorithmus wird an dieser Stelle noch keine Aussage getroffen. Eine erste qualitative Bewertung erfolgt in Kapitel 8.3.2.4.

8.3.2.2 Best Fit Heuristik

Der heuristische Ansatz, der in diesem Unterkapitel erläutert wird, unterscheidet sich im allgemeinen Ablauf nicht von dem in Kapitel 8.3.2.1 vorgestellten Vorgehen. Deswegen wird an dieser Stelle nur auf die Veränderungen innerhalb der Verarbeitung eingegangen und nicht mehr detailliert auf den allgemeinen Ablauf.

Diese Variante eines heuristischen Algorithmus betrachtet die Allokation der Product Owner Stories in Bezug auf die Auswahl des Entwicklerteams genauer. Hierbei wird versucht durch die Verwendung und Verarbeitung von weiteren Informationen die Ergebnisqualität zu verbessern. Wie beschrieben wird in der Grundvariante für eine Product Owner Story immer das zuerst gefundene Team ausgewählt, dass die entsprechenden Rahmenbedingungen erfüllt.

Eine mögliche Verbesserung dieser Auswahl könnte sein, dass nicht das erste Team ausgewählt wird. Bei der Best Fit Heuristik werden zunächst alle Teams geprüft, ob sie die entsprechende Product Owner Story umsetzen können. Aus diesen Teams, die die notwendigen Kompetenzen besitzen, wird das Team mit der größten verbleibenden Kapazität ausgewählt.

Durch diese Herangehensweise sollen zwei mögliche Probleme der simplen Heuristik behoben werden. Zum einen sollen die Entwicklerteams gleichmäßiger ausgelastet werden und zum anderen soll verhindert werden, dass ein Team bereits „zu früh“ komplett ausgeplant wird, sodass ggf. andere User Stories nicht mehr bearbeitet werden können, da unter Umständen notwendige Product Owner Stories exklusiv von dieser Teamressource bearbeitet werden können.

Die Verarbeitung wird ausschnittsweise im folgenden Quellcode dargestellt.

```
def doBestFitAssignment(u):
...
    productOwnerStories = getStories(u)
...
   for p in productOwnerStories:
        assignmentTeamCap = 0
        for t in team.keys():
            cap, caplist = team.get(t)
            if p in caplist and cap >= a(p) and cap > assignmentTeamCap:
                teamfound = True
                assignmentTeam = t
                assignmentTeamCap = cap

...
```

Wie erläutert, werden zunächst alle benötigten Product Owner Stories bestimmt. Das Entwicklungsteam je Product Owner Story wird durch eine Iteration über alle vorhandenen Teams vollzogen. Für jedes Team wird die verbleibende Kapazität bestimmt sowie die Kompetenz für die

Umsetzung der Product Owner Story geprüft. Anschließend wird begutachtet, ob dieses Team diese Product Owner Story umsetzen kann und ob das Team noch über eine ausreichende Kapazität verfügt und die verbleibende Kapazität größer als die eines ggf. bereits gefundenen Teams ist. Ist dies der Fall, wird diese Product Owner Story diesem Team zugeordnet.

Auch hier gilt, dass eine Allokation nur vorgenommen wird, wenn alle Product Owner Stories zugeteilt werden können. Andernfalls werden alle vorläufigen Allokationen wieder zurückgenommen.

8.3.2.3 Future Fit Heuristik

Auch bei dieser Variante bleibt der allgemeine Ablauf unverändert, auch hier wird lediglich das Allokationsmodell weiter verfeinert. In Kapitel 8.3.2.2 werden Informationen über die „Vergangenheit", also bereits getätigte Allokationen bei der Zuteilung weiterer Product Owner Stories mit berücksichtigt. Aus den Eingaben der Probleminstanz können aber auch Informationen über die „Zukunft" abgeleitet werden. Neben dem Wissen über bereits verarbeitete Product Owner Stories besteht auch die Möglichkeit, die Product Owner Stories mit zu betrachten, die in den folgenden Verarbeitungsschritten noch verarbeitet werden müssen. Dieser Ablauf wird an folgendem Beispiel illustriert.

Annahme: Es gibt zwei Entwicklerteams mit einer freien Kapazität für Team 1 von 8 SP und für Team 2 von 16 SP. Es soll eine Product Owner Story mit dem Aufwand 8 einem Team zugeteilt werden. Beide Teams sind potenziell in der Lage diese Product Owner Story zu bearbeiten. Ohne weitere Informationen würde die zuvor erläuterte Heuristik diese Product Owner Story Team 2 zuweisen, da dieses über die höhere verbleibende Kapazität verfügt. Im Fall, dass bspw. noch zwei weitere User Stories im Backlog verarbeitet werden müssen und diese jeweils mit nur einer Product Owner Story mit dem Aufwand 8 assoziiert sind, welche ausschließlich von Team 2 bearbeitet werden kann, würde diese Heuristik kein optimales Ergebnis liefern. Dies hätte zur Folge, dass Team 1 nicht vollständig ausgelastet wäre. Außerdem würde eine User Story nicht bearbeitet werden, obwohl diese mit einer anderen Zuteilung noch im Sprint hätte untergebracht werden können.

Deswegen berücksichtigt diese heuristische Variante nicht nur verbleibende Kapazitäten der Teams, sondern auch die verbleibenden Aufwände innerhalb des noch nicht verarbeiteten Backlogs, um die Zuteilung für ein Team zu bestimmen. Der entsprechende Indikator i ergibt sich als:

$$i = verbleibende\ Kapazität - \sum_{p \in POS} a(p) * x_p$$

$$\text{mit } x_p = \begin{cases} 1\ wenn\ p \in \bigcup_{K_u \in K} K_u\ \ und\ p \in CD_d \\ 0\ sonst \end{cases}$$

Dieser Indikator wird in der weiteren Verarbeitung anstelle der verbleibenden Kapazität verwendet. Die restliche Abarbeitung bleibt unverändert.

8.3.2.4 Evaluation der Verfahren

Für die Evaluation der vorgestellten heuristischen Verfahren wurden zunächst 100 Probleminstanzen mit zwei x-funktionalen Entwicklerteams und 20 Product Owner Stories zufällig generiert.

Anschließend wurde mittels einer naiven Brute-Force-Methode die optimale Lösung für jede der zuvor erzeugten Probleminstanzen erzeugt. Mit diesen Ergebnissen wurden dann die Ergebnisse des jeweiligen heuristischen Vorgehens verglichen. Die Ergebnisse dieser Untersuchung sind in Abbildung 40 dargestellt.

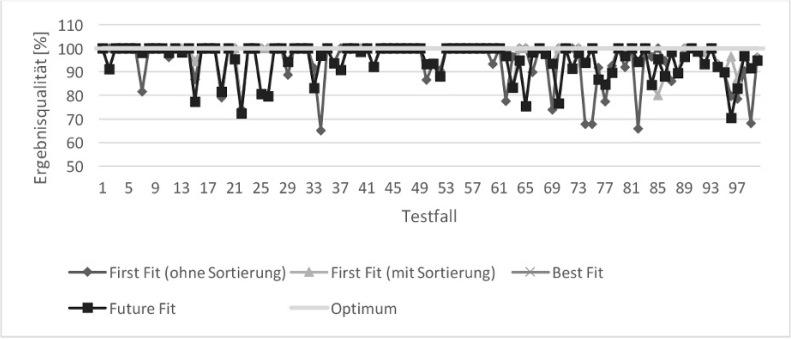


Abbildung 40: Vergleich der Heuristiken bei komplett cross-funktionalen Teams

Dem Diagramm ist zu entnehmen, dass das Best Fit- und das Future Fit Verfahren identische Werte aufzeigen. Hierbei ist zu beachten, dass dieses das zu erwartende Ergebnis ist, da es sich bei den zufällig erzeugten Probleminstanzen um komplett x-funktionale Teams handelt. Somit ist die verbleibende Belastung für beide Teams immer identisch und das verbleibende Unterscheidungskriterium sind die bereits einem Team zugeteilten Product Owner Stories. Dadurch erzielt dieses Verfahren die identischen Werte wie das erwähnte Best Fit Verfahren.

Wie der Abbildung ebenfalls zu entnehmen ist, gelingt es, den beschriebenen Verfahren in manchen Fällen sogar eine 100 %-Lösung anzubieten. Im Mittel erreichen alle vorgestellten Verfahren eine Lösungsqualität von ca. 94 %. Im schlechtesten Fall erreicht die First Fit Heuristik ohne zwischenzeitliches Neusortieren nur eine Ergebnisqualität von ca. 55 %, die anderen Verfahren liefern Ergebnisqualitäten von mehr als 70 %.

In Abbildung 41 werden die summierten erzeugten Nutzenwerte über alle 100 Testfälle je Verfahren dargestellt. Es zeigt sich, dass die First Fit Heuristik in diesem Fall sogar minimal bessere Ergebnisse erzeugt als die anderen Verfahren. Es ist auch festzuhalten, dass sich die Ergebnisse aller Verfahren bei komplett x-funktionalen Teams auf einem relativ ähnlichen Niveau befinden und sich keine gravierenden Unterschiede ergeben.

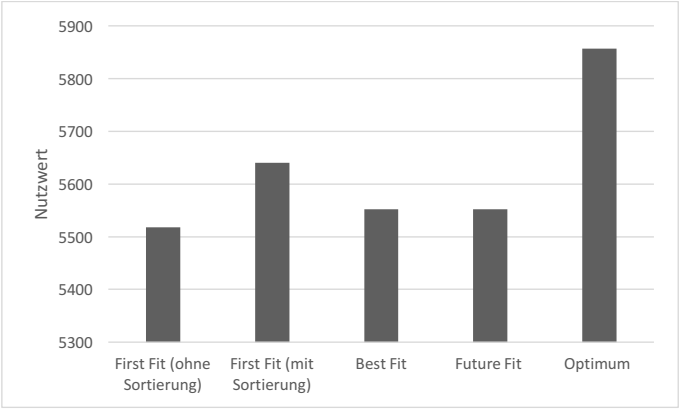


Abbildung 41: Erzeugter Nutzwert über 100 Testfälle durch heuristische Verfahren (x-funktional)

Abbildung 42 zeigt die Verarbeitungszeiten, die das jeweilige Verfahren benötigt, um alle 100 Instanzen des Problems zu bearbeiten und eine Lösung zu generieren. Das First Fit- wie auch das Best Fit Verfahren benötigen weniger als 6 Sekunden, um alle 100 Instanzen mit jeweils 20 Product Owner Stories zu bearbeiten. Das Future Fit Verfahren benötigt für die identischen Testfälle ca. 10,5 Sekunden. Dies entspricht auch dem erwarteten Zeitverhalten, da die Ermittlung der verbleibenden für ein Team theoretisch zuteilbaren Product Owner Stories aufwendiger ist als die Bestimmung der verbleibenden Kapazität.

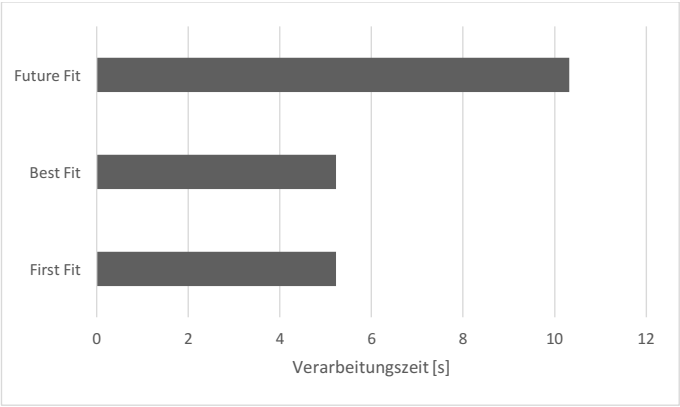


Abbildung 42: Verarbeitungszeit von 100 Instanzen mit 20 POS, 2 Teams, x-functional Teams, in Sekunden

Da es sich bei dem Fall von komplett cross-funktionalen Entwicklerteams zunächst um einen Sonderfall handelt, wurden in einem zweiten Testlauf 100 zufällige Probleminstanzen erstellt, bei denen die Teams nicht mehr komplett cross-funktional sind. In diesen Probleminstanzen ist eine Teilmenge der Product Owner Stories von beiden Teams gleichermaßen bearbeitbar. Ein

weiterer Teil der Product Owner Stories ist jedoch jeweils nur exklusiv von einem Team zu bearbeiten.

Die Erwartung ist, dass in diesem Fall die Future Fit Heuristik im Durchschnitt die besten Ergebnisse erzeugen müsste, gefolgt von der Best Fit Heuristik und der First Fit Heuristik.

Abbildung 43 zeigt die Ergebnisqualität in Prozent zum jeweiligen Testfall bzgl. des eingesetzten Verfahrens. Es ist festzustellen, dass die maximale Güte, die jeweils erreicht wurde, bei 100 % liegt. Dies bedeutet, dass alle drei hier untersuchten Verfahren potenziell optimale Lösungen erzeugen können. Die durchschnittlichen Ergebnisqualitäten über alle 100 Testinstanzen mit 20 Product Owner Stories und zwei nicht komplett cross-funktionalen Entwicklerteams liegen für das First Fit Verfahren bei 89 %, für das Best Fit Verfahren bei 90 % und für das Future Fit Verfahren bei ca. 92 %. Die minimale Ergebnisqualität liegt bei allen drei Verfahren bei ca. 52 %.

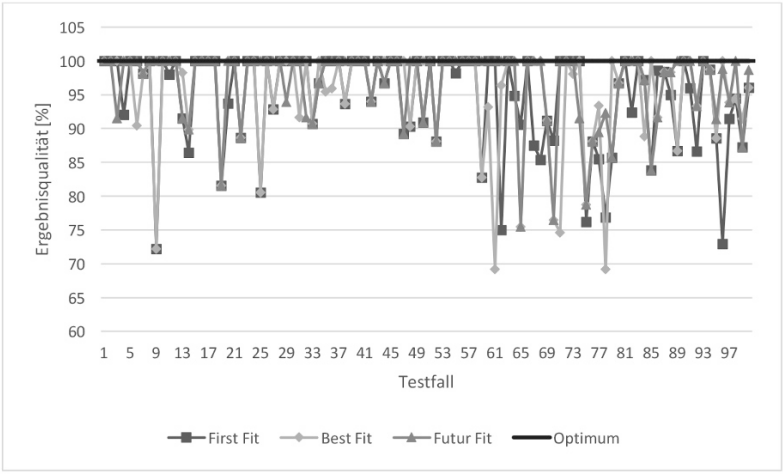


Abbildung 43: Vergleich der Heuristiken bei nicht komplett cross-funktionalen Teams

In Abbildung 44 sind die summierten Nutzwerte für alle 100 zufällig erzeugten Probleminstanzen dargestellt.

Aus dieser Abbildung wird ersichtlich, dass sich die vor dieser Untersuchung formulierte These bestätigt hat. Über alle zufällig erzeugten Probleminstanzen ist das Future Fit Verfahren das Verfahren, welches den höchsten Gesamtnutzen erzielt hat. Den zweitbesten Gesamtnutzen hat das Best Fit Verfahren erzeugt. Das einfachste dieser drei Verfahren, das First Fit Verfahren, erreicht in diesem Szenario den geringsten summierten Nutzenwert.

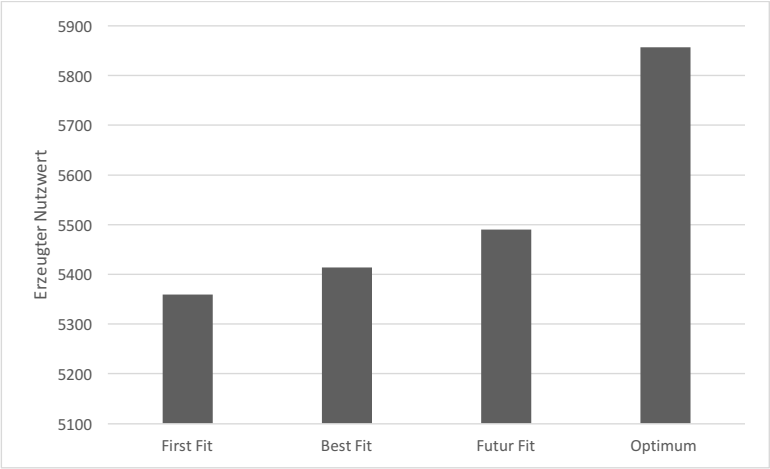


Abbildung 44: Erzeugter Nutzwert über 100 Testfälle durch heuristische Verfahren (nicht komplett x-funktional)

8.3.2.5 Diskussion der Ergebnisse

Der Vergleich der Ergebnisse der vorgestellten heuristischen Verfahren in Kapitel 8.3.2.4 zeigt, dass alle drei Verfahren potenziell optimale Lösungen generieren können. Eine detaillierte Auswertung der Ergebnisse hat ergeben, dass die durchschnittliche Ergebnisqualität bei ca. 90 % liegt. Nichtsdestoweniger sind die Vorschläge, welche im schlechtesten Fall erzeugt werden, bei ca. 66 %. Das heißt, in ungünstigen Konstellationen können keine zufriedenstellenden Ergebnisse erreicht werden.

Ergebnisse, die eine Qualität von 90 % oder mehr aufweisen, sind qualitativ hinreichend gut, da zum einen der Aufwand und zum anderen der Nutzenwert nur als grobe Schätzungen eingehen. Ergebnisse im Bereich einer Ergebnisqualität von lediglich 70 % weisen jedoch ein deutliches Verbesserungspotenzial auf.

Deswegen wird im folgenden Abschnitt die Leistungsfähigkeit naturanaloger Verfahren untersucht.

8.3.3 Naturanaloge Lösungsverfahren

Bei naturanalogen Lösungsverfahren handelt es sich im Kontext dieser Arbeit um Algorithmen, die aus der Natur inspiriert sind. Es wird versucht, die Mechanismen der Natur nachzubilden und auf neue oder ähnliche Probleme anzuwenden.

8.3.3.1 Binäre Repräsentation einer Lösungsmöglichkeit

Da sowohl für einen genetischen Algorithmus, als auch für die Anwendung von Simulated Annealing eine Repräsentation für einen Lösungsvorschlag erarbeitet werden muss, welche einfach bewertet und anschließend weiterverarbeitet werden kann, wurde für beide Verfahren eine

gemeinschaftliche Repräsentation für einen entsprechenden Kandidaten erarbeitet, die an dieser Stelle allgemein für die folgenden Verfahren vorgestellt wird.

Für die Repräsentation eines Lösungskandidaten wird eine binäre Darstellung verwendet. Der Vorteil ist zum einen, dass entsprechende Operationen eines genetischen Algorithmus einfach umgesetzt werden können und zum anderen ist die Repräsentation immer noch vom Menschen nachvollziehbar, sowie auch wieder zurückführbar auf eine vereinfachte Repräsentation für die Darstellung.

Jede Product Owner Story kann entweder in einen Sprint eingeplant werden (1) oder nicht Bestandteil des Sprint Backlogs werden (0). Das bedeutet, dass die Länge einer binären Repräsentation eines Lösungskandidaten mindestens so groß ist wie die Anzahl an Product Owner Stories. Um beschreiben zu können, welchem Team eine Product Owner Story zugeteilt wird, wird für jedes Team eine entsprechend lange binäre Repräsentation erzeugt. Dieses wird in nachfolgender Abbildung 45 schematisch dargestellt. Es werden zwei Entwicklerteams und fünf Product Owner Stories abgebildet.

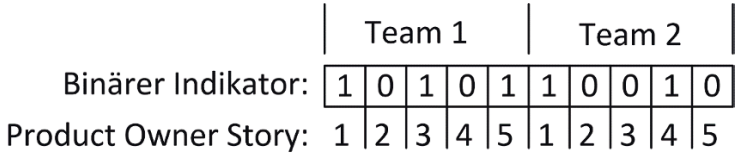


Abbildung 45: Binäre Repräsentation eines Lösungskandidaten (eigene Darstellung)

In diesem Beispiel sind Team 1 die erste, dritte und fünfte Product Owner Story zugeteilt. Team 2 sind die Product Owner Story eins und vier zugeordnet. Um bspw. die Bedingung zu prüfen, ob eine Product Owner mehreren Teams zeitgleich zugeordnet ist, reicht es an dieser Stelle aus, den binären Vektor an der Schnittstelle der Teams aufzuspalten und die entstehenden Vektoren mit einem bitweisen logischen „und“ zu verbinden. Ist der entstehende Vektor abweichend vom 0-Vektor, so ist mindestens eine Product Owner Story mehr als einem Team zugeordnet.

Die Rahmenbedingung bzgl. der Abgabefristen kann auf ähnliche Art und Weise geprüft werden. Hierzu wird zunächst ein Referenzvektor erzeugt, der Einsen an den Stellen enthält, die die notwendigen Product Owner Stories repräsentieren, ansonsten Nullen. Anschließend wird der Lösungskandidat wieder entlang der Teamgrenzen gespalten. Abweichend vom vorherigen Vorgehen werden dieses Mal die Vektoren mit einem bitweisen binären „oder“ verbunden. Dieser resultierende Vektor wird dann mittels bitweisem binären „und“ mit dem zuvor erzeugten Vektor der Abgabefristen verrechnet. Entspricht der Ergebnisvektor dieser Operation dem initialen Abgabefristvektor, so ist diese Bedingung erfüllt, andernfalls ist mindestens eine benötigte Product Owner Story nicht Teil der Lösung.

8.3.3.2 Simulated Annealing

Simulated Annealing besitzt Analogien zur statistischen Thermodynamik. Das zu optimierende System wird bei einer hohen Temperatur „geschmolzen“. Somit liegt zunächst ein hohes Energieniveau vor, sodass ein hoher „Bewegungsfreiraum“ vorherrscht. Dadurch hat das System viel Spielraum bei der Erzeugung initialer Lösungen. Das System wird anschließend schrittweise abgekühlt bis es erstarrt und keine Änderungen mehr auftreten können. Bei jeder Temperatur muss die Simulation lange genug auf dieser Temperatur verweilen und unterschiedliche

Lösungskandidaten erzeugen, um einen neuen stabilen Lösungskandidaten zu generieren. Dabei werden die Temperaturverringerung und die Neuanordnung als Abkühlschema bezeichnet. (Ruigies 1998)

Die Vorteile dieses Verfahrens sind, dass das Verfahren nicht „stecken“ bleiben kann, da eine Verbesserung aus einem lokalen Optimum immer möglich ist (bis zum kompletten Erstarren). Außerdem ergibt sich eine Art von adaptivem „Teile-und-herrsche“. Grobe Lösungsstrukturen entwickeln sich dabei bei höheren Temperaturen, während bei niedrigeren Temperaturen nur noch eine Detail-Verbesserung erfolgt. (Ruigies 1998)

Das allgemeine Schema dieses Algorithmus wird nachfolgend grob dargestellt:

```
Wähle eine Anfangskonfiguration
Wähle Temperatur t > 0

while (t > 0):
            for i in range(Anzahl zu prüfender Konfigurationen):
                konfiguration_neu = veraendere(konfiguration)
                dE = nutzwert(konfiguration_neu)-nutzwert(konfiguration)
                if(dE < 0):
                    pAccept = math.exp(-dE/(temp*1.0))
                    if (random.random() < pAccept):
                        konfiguration = konfiguration_neu
                else:
                        konfiguration = konfiguration_neu
            verringere(t)
```

Auf jeder Temperaturstufe wird eine bestimmte Anzahl an neuen Konfigurationen geprüft und bewertet. Ist im Fall des Sprint-Optimierungsproblems der erzeugte Nutzwert der neuen Lösung besser als der der zuvor fixierten Lösung, wird diese neue Lösung als Referenzlösung übernommen. Ist das Ergebnis nicht besser, so wird mit einer gewissen Wahrscheinlichkeit in Abhängigkeit des Betrags der Verschlechterung und der Temperatur zufällig bestimmt, ob diese Lösung dennoch als neue Lösung fixiert werden soll.

Der Erfolg dieses Verfahrens ist zum einen vom Abkühlschema und zum anderen von der Bewertungsfunktion abhängig. Um den allgemeinen Ablauf nicht zu verändern, wurde zur Bewertung eine Funktion ermittelt, die zum einen den Nutzwert der Lösung bestimmt und zum anderen Strafen für Verletzungen von Rahmenbedingungen gegen diesen Nutzwert gegenrechnet, um diesen zu reduzieren. Dieses Verfahren wurde bspw. erfolgreich beim Rucksackproblem eingesetzt und angewendet vgl. z. B. (Olsen 1994).

Bei dem hier angepassten Verfahren für die Bewertung eines Zustandes bzw. eines Lösungskandidaten wird zunächst der Nutzwert, der durch die eingeplanten Product Owner Stories erzielt wurde, aufsummiert. Anschließend werden Nutzwertreduzierungen für Verletzungen der Bedingungen vom Algorithmus bestimmt. Dabei wird so vorgegangen, dass für jede nicht erfüllte Bedingung der Nutzwert der höchstwertigsten User Story vom Ergebnis abgezogen wird. Dies ist in der Tatsache begründet, dass bspw. bei einer Überplanung eines Teams im schlechtesten Fall eine Product Owner Story zur Erfüllung eben dieser User Story nicht umgesetzt werden kann. Die Ausnahme von dieser Regel bildet ein Verstoß gegen die Rahmenbedingung, dass alle Product Owner Stories, die mit einer Deadline belegt sind, eingeplant werden müssen. In diesem Fall wird der Nutzwert pauschal auf 0 gesetzt. Eine Verletzung dieser Bedingung kann im schlechtesten Fall dazu führen, dass das System nach Verstreichen dieser Abgabefrist nicht betriebsfähig ist. Dieser Fall muss unter allen Umständen vermieden werden.

Nachdem das Bewertungsschema eines Zustandes grob skizziert ist, muss noch ein entsprechendes Abkühlschema definiert werden. Die Stellgrößen, die auf die Güte des Ergebnisses Einfluss haben, sind zum einen die Starttemperatur und zum anderen die Verweildauer auf einem bestimmten Temperaturniveau, wie auch die Abkühlrate.

In Abbildung 46 sind zunächst die aufsummierten Nutzwerte über 100 Testinstanzen mit nicht komplett x-funktionalen Teams in Abhängigkeit der Starttemperatur und der Verweildauer dargestellt. Die Abkühlrate liegt bei dieser Untersuchung konstant bei 1 %.

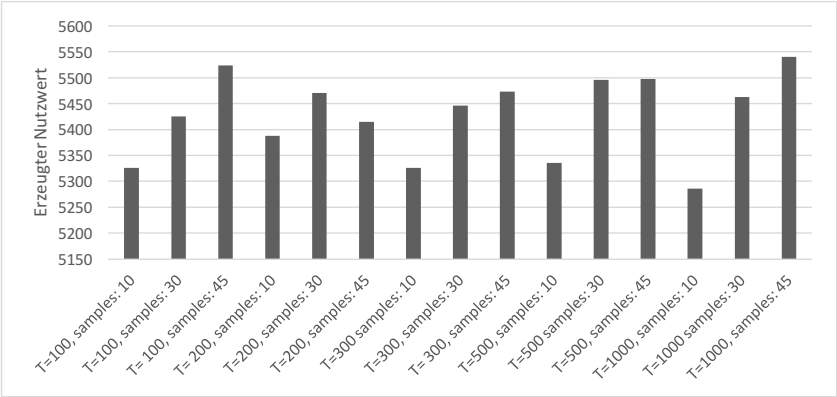


Abbildung 46: Erzeugter Nutzwert über 100 Testinstanzen in Abhängigkeit von Starttemperatur und Samples je Temperaturschritt

Abbildung 46 ist zu entnehmen, dass die Wahl der Starttemperatur bei diesem Problem eine eher untergeordnete Rolle spielt. Die erzeugten Nutzwerte bei identischer Verweildauer auf einer Temperaturstufe sind hierbei relativ ähnlich. Unabhängig von der Wahl der Starttemperatur ergibt sich, mit Ausnahme der Starttemperatur von 200, ein wiederkehrendes Muster. Der erzeugte Nutzwert über alle 100 Testinstanzen steigt mit der Verweildauer auf einer Temperaturstufe an.

Deswegen wurde in einem zweiten Schritt die Verweildauer auf einer Temperaturstufe erhöht, um zu prüfen, wie weit sich durch eine Anpassung dieses Parameters weitere Verbesserungen bzgl. der Ergebnisqualität erzielen lassen.

In Abbildung 47 wird dargestellt, wie sich der über 100 Instanzen erzeugte Nutzwert verändert, wenn bei Starttemperatur 100 die Verweildauer schrittweise verlängert wird.

Es ist zu ersehen, dass der über alle Instanzen erzeugte Nutzwert trotz einer längeren Verweildauer auf jeder Temperaturstufe bis zu 45 samples je Stufe ansteigt, bei einer weiteren Verlängerung keine weiteren positiven Effekte mehr eintreten und keine weitere Verbesserung erreicht wird. Wie in der nachfolgenden Abbildung 48 für zehn ausgewählte Testfälle zu erkennen ist, kann die Ergebnisqualität je Testfall unterschiedlich ausfallen. In der Summe über alle Testfälle kann jedoch keine absolute Verbesserung erzielt werden.

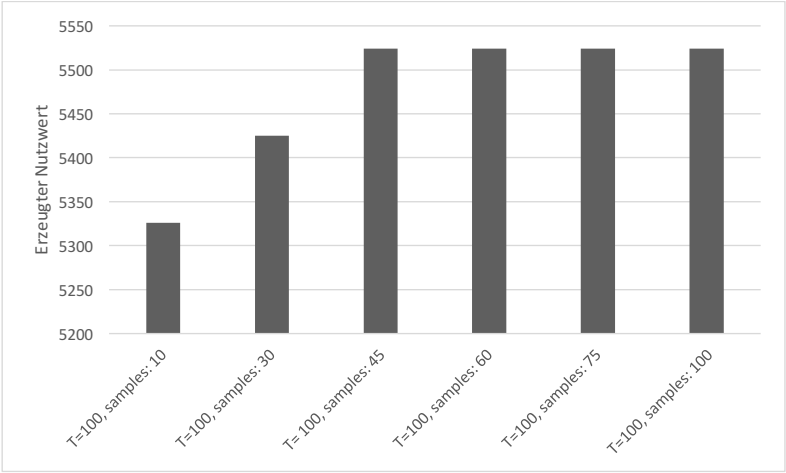


Abbildung 47: Summierte Nutzwerte bei unterschiedlichen Verweildauern, konstanter Starttemperatur und konstanter Abkühlrate

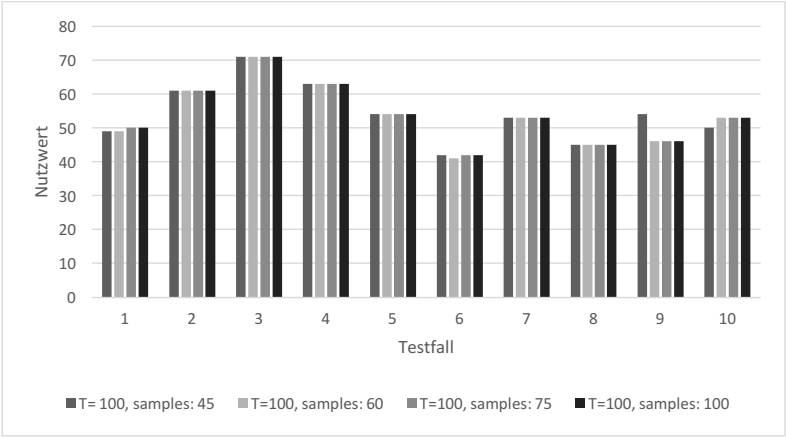


Abbildung 48: Vergleich unterschiedlicher Verweildauern je Temperaturstufe bei zehn zufällig ausgewählten Testfällen

Aufgrund dieser Befunde wird zunächst eine Starttemperatur von 100 bei einer Verweildauer von 45 Zuständen je Temperaturstufe fixiert. Im Folgenden wird nun mit diesen Parametern versucht die Abkühlrate so anzupassen, dass der Algorithmus die bestmöglichen Ergebnisse liefern kann.

In Abbildung 49 ist zunächst der erzeugte Gesamtnutzwert über 100 zufällig erzeugte Probleminstanzen mit nicht komplett cross-funktionalen Teams in Abhängigkeit der Abkühlrate dargestellt.

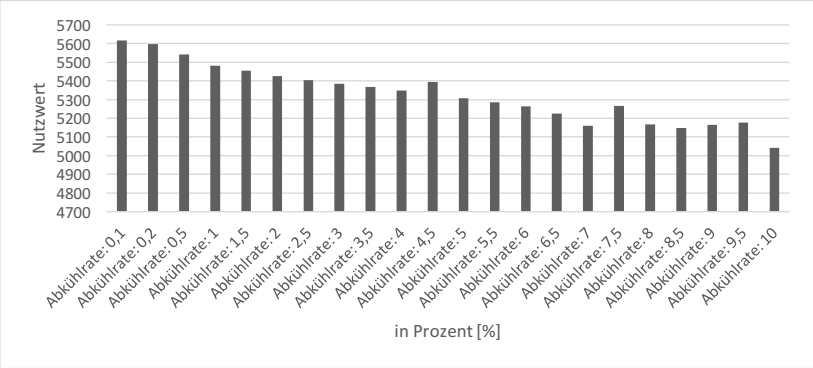


Abbildung 49: Veränderung der Ergebnisqualität unter Berücksichtigung unterschiedlicher Abkühlraten

Der Abbildung ist zu entnehmen, dass der gesamt erzeugte Nutzwert durch eine schnelle Abkühlung verringert wird. Da jedoch neben der Ergebnisqualität auch der Zeitaufwand für die Erzeugung der Ergebnisse mit berücksichtigt werden muss, wird im Folgenden der Zeitaufwand für die Lösungsgenerierung in Abhängigkeit der Abkühlrate analysiert.

Abbildung 50 zeigt den summierten Zeitaufwand in Sekunden, der für die Lösungsgenerierung von 100 Testinstanzen in Abhängigkeit der Abkühlrate benötigt wird. Es zeigt sich, dass der Zeitaufwand bei Abkühlraten kleiner 1 % deutlich ansteigt. Der absolute Zuwachs an erzeugtem Nutzwert korreliert jedoch nicht mit dem zusätzlichen Zeitaufwand. Deswegen muss an dieser Stelle ein Trade-Off zwischen Ergebnisqualität und Zeitaufwand gefunden werden.

Abbildung 51 zeigt das Verhältnis aus erzeugtem Nutzwert über 100 Testinstanzen zum Zeitaufwand unter Berücksichtigung der Abkühlrate. Wie diesem Diagramm zu entnehmen ist, liegt das optimale Verhältnis zwischen Zeitaufwand und Nutzwert bei einer Abkühlrate von 9 %.

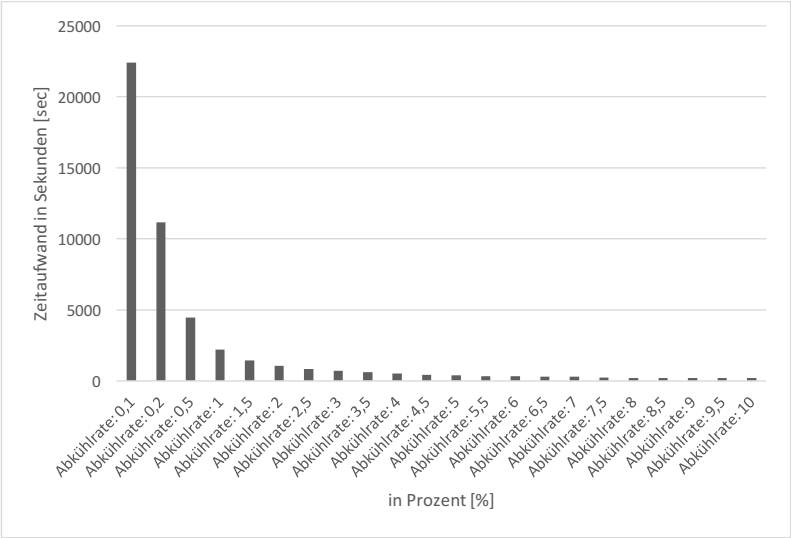


Abbildung 50: Zeitaufwand in Abhängigkeit der Abkühlrate (100 Instanzen)

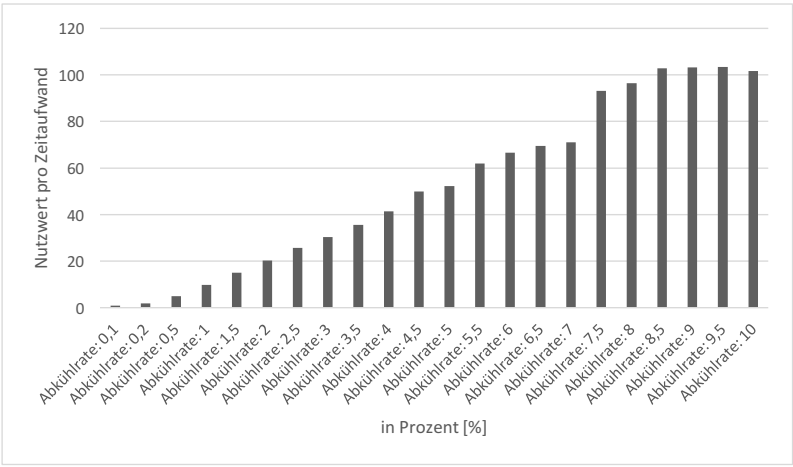


Abbildung 51: Verhältnis von erzeugtem Nutzwert zu Zeitaufwand bei unterschiedlichen Abkühlraten

Zusammenfassend ist an dieser Stelle festzustellen, dass für die hier beschriebenen Probleminstanzen mit 20 Product Owner Stories und 2 Teams eine Starttemperatur von 100 bei einer Samplingrate von 45 je Temperaturstufe und einer Abkühlrate von 9 % die besten Ergebnisse bzgl. des Rechenaufwandes erwarten lässt.

Sollte sich Simulated Annealing als geeignetstes Verfahren herausstellen, so müssten vor einem Einsatz die hier aufgezeigten Untersuchungen zur Bestimmung eines optimierten Abkühlschemas erneut durchgeführt werden, da nicht sichergestellt werden kann, dass diese Parameter auch für eine steigende Anzahl an Teams und Product Owner Stories weiterhin als geeignet angesehen werden können.

8.3.3.3 Genetischer Algorithmus

Genetische Algorithmen sind in ihrer Funktionsweise an die Mechanismen der Evolution des natürlichen Lebens angelehnt. Es handelt sich hierbei um ein naturanaloges metaheuristisches Optimierungsverfahren.

Genetische Algorithmen wurden vor allem durch die Arbeiten J. Hollands bekannt (Holland 1992a). Für die Darstellung eines Lösungskandidaten wird dabei häufig eine binäre Repräsentation gewählt, da diese effizient verarbeitet werden kann. Für die Bewertung eines Lösungskandidaten durch eine sogenannte Fitnessfunktion wird meist ein Genotyp-Phänotyp-Mapping benötigt. Dieses bedeutet, dass die binäre Darstellung zunächst in eine andere Repräsentation des Problems übersetzt werden muss, um eine Bewertung durchführen zu können. Die genaue Übersetzungsvorschrift ist dabei für jedes Problem spezifisch.

Die binäre Repräsentation kann auch als Erbgut verstanden werden. Durch Funktionen, welche an den natürlichen Evolutionsprozess angelehnt sind, sollen somit durch „Evolution“ qualitativ hochwertige Ergebnisse erzeugt werden. Zu diesen aus der Evolution abgeleiteten Funktionen gehören z. B. Mutation und Rekombination.

Mutation kann innerhalb eines genetischen Algorithmus z. B. durch das zufällige invertieren einzelner Bits der binären Repräsentation nachgebildet werden. Rekombination und Fortpflanzung erfolgen bei einem genetischen Algorithmus mit fitnessproportionaler Selektion. Dies bedeutet, dass für die „Fortpflanzung“ als Eltern die Lösungskandidaten, die eine bessere Bewertung durch die Fitnessfunktion erfahren haben, mit einer größeren Wahrscheinlichkeit ausgewählt werden, als jene, welche eine schlechte Bewertung erfahren haben. Dieses bildet in gewissen Maßen das Prinzip des Überlebens der Stärksten nach. Grundsätzlich werden bei der „Fortpflanzung“ die binären Repräsentationen der Eltern an einem (1-Punkt-Crossover) oder mehreren Punkten (n-Punkt-Crossover) geteilt und anschließend neu zusammengesetzt, sodass die „Nachfahren“ jeweils Teileigenschaften der „Vorfahren“ übernehmen. Auch die Rekombination von mehr als zwei Lösungskandidaten ist theoretisch in einem Algorithmus möglich und führt in manchen Fällen zu besseren Ergebnissen (Ting 2005) .

Für andere bekannte NP-schwere Probleme zeigen genetische Algorithmen, dass sie in der Lage sind gute Näherung zu erzeugen (s. Kapitel 4.3.2.2 und 4.3.3.2). Aufgrund dieser guten Ergebnisse bei anderen Problemen dieser Klasse wurde im Rahmen dieser Arbeit ein genetischer Ansatz erarbeitet.

Der allgemeine Ablauf eines genetischen Algorithmus ist im Folgenden als Pseudo-Code dargestellt:

```
Gegeben: Suchraum S und Fitnessfunktion f
generiere zufällige Startpopulation P
WHILE(Abbruchbedingung nicht erfüllt)
{
        Berechne f für jedes Element aus P
        Selektion von Individuen für nächste Generation
        Anwendung genetischer Operatoren
        neue Population erzeugen
}
Ergebnis: bestes Individuum aus der letzten Population P
```

Der Suchraum ergibt sich hierbei aus der binären Repräsentation eines Lösungskandidaten. Als Fitnessfunktion dient die beschriebene Funktion zur Ermittlung des Nutzwertes einer Lösung. Auch bei diesem Verfahren wird wie bei Simulated Annealing mit Strafen für Verletzungen der Rahmenbedingungen gearbeitet.

Bei der Startpopulation kann es sich bspw. um 100 zufällig erzeugte Lösungskandidaten handeln. Die Größe der Startpopulation hat dabei Auswirkungen auf die Ergebnisqualität und auch auf die Laufzeit des Algorithmus.

Beim Schritt der Selektion werden „zufällig" Elemente aus der Population ausgewählt. Diese dienen dann in den folgenden Schritten als „Elternelemente". Bei der Auswahl werden Individuen (Lösungskandidaten) mit einem hohen Nutzwert, also einer hohen Fitness, bevorzugt. Dieses bildet die Systematik des „survival of the fittest" nach.

Im Schritt der Anwendung der genetischen Operatoren werden zum einen neue Individuen durch ein sogenanntes Crossover erzeugt, zum anderen werden Veränderungen durch Mutation vorgenommen. Dabei stellt das Crossover sowohl den fundamentalen Mechanismus für genetische Reorganisation und Neuordnung für lebende Organismen als auch für genetische Algorithmen dar (Holland 1992b). Dieser Mechanismus wird in Abbildung 52 schematisch dargestellt. Zunächst werden zwei Individuen (Eltern) bestimmt. Anschließend werden die Gensequenzen an einer Stelle (einfaches Crossover) getrennt und anschließend die Sequenzen rekombiniert, sodass die neu entstehenden Individuen (Kinder) jeweils einen Teil der Gene vom Vater als auch einen Teil der Gene von der Mutter übernehmen.

Beim Crossover kann, wie bereits erwähnt, sowohl das veranschaulichte 1-Punkt-Crossover implementiert werden als auch ein n-Punkt-Crossover, bei dem die Sequenzen an mehreren Stellen gespalten und anschließend neu zusammengesetzt werden. Eine Stellgröße für die Konfiguration eines genetischen Algorithmus ist die Bestimmung der Häufigkeit mit der ein Crossover stattfindet bzw. mit welcher Wahrscheinlichkeit die ausgewählten Eltern keine Nachkömmlinge hervorbringen und selbst in die nächste Generation übergehen.

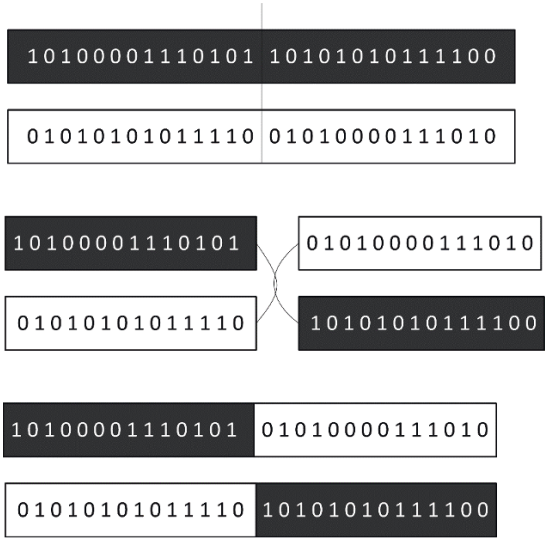


Abbildung 52: Schematische Darstellung Crossover (eigene Darstellung in Anlehnung an (Holland 1992b))

Ein weiterer genetischer Operator ist die Mutation. Bei der Mutation werden zufällig einzelne Bits entlang der Sequenz invertiert. Die Mutationswahrscheinlichkeit eines Bits ist ebenfalls eine Stellgröße für die Anpassung an ein bestimmtes Problem. Ist die Mutationswahrscheinlichkeit zu gering, wird ein lokaler Zustand mit zu geringer Wahrscheinlichkeit verlassen und es wird nicht der „komplette“ Suchraum erschlossen. Ist die Mutationswahrscheinlichkeit zu hoch, so werden gute Lösungen zu schnell wieder verworfen, da sie durch Mutation verändert werden.

Im Folgenden wird nun versucht, vor der eigentlichen Evaluation des Verfahrens in dem Vergleich mit dem Verfahren des Simulated Annealings eine bestmögliche Konfiguration auf Basis der 100 Testinstanzen zu generieren.

Zunächst wird die Anzahl an Generationen auf 500 sowie die Populationsgröße auf 100 festgesetzt. In Abbildung 53 wird dargestellt, wie sich der erzeugte Nutzwert über 100 zufällige Testinstanzen mit 20 Product Owner Stories in Abhängigkeit der Crossover-Wahrscheinlichkeit und der Mutations-Wahrscheinlichkeit verändert. Die jeweiligen Werte sind auf der x-Achse angefügt. Die obere Zahl gibt dabei die Crossover- und die untere die Mutations-Wahrscheinlichkeit an. Die Crossover-Wahrscheinlichkeit wird dabei von 50 % - in 10 %-Schritten auf 100 % Prozent je Mutations-Wahrscheinlichkeit erhöht. Die Mutationswahrscheinlichkeit erstreckt sich von 2 % bis 9 % in 1 %-Schritten.

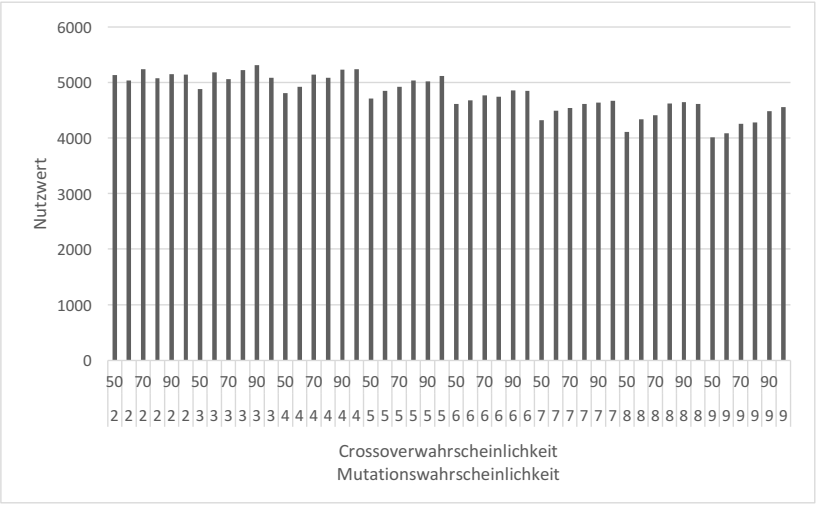


Abbildung 53: Erzeugte Nutzwerte eines genetischen Algorithmus in Abhängigkeit von Crossover- & Mutationswahrscheinlichkeit. (eigene Darstellung)

Wie in der Abbildung 53 dargestellt, ergibt sich ein wiederkehrendes Muster. Bei einer fixierten Mutationswahrscheinlichkeit steigt der Nutzwert mit steigender Crossover-Wahrscheinlichkeit bis 90 % an. Darüber hinaus ist festzustellen, dass der gesamte erzeugte Nutzwert ab einer Mutationswahrscheinlichkeit von 4 % und mehr schrittweise abfällt. Das beste Gesamtergebnis wird bei einer Mutationswahrscheinlichkeit von 3 % und einer 90- prozentigen Crossover-Wahrscheinlichkeit erzielt.

Im nächsten Schritt werden diese Konfigurationsparameter fixiert und die Populationsgröße schrittweise nach oben gesetzt. Da die Verarbeitungsgeschwindigkeit abhängig von der Populationsgröße ist, wird es an dieser Stelle ggf. wieder notwendig, einen Trade-Off zwischen Ergebnisqualität und Populationsgröße zu finden, wenn sich herausstellen sollte, dass die Ergebnisse mit einer steigenden Populationsgröße qualitativ gesteigert werden können.

In Abbildung 54 ist die Veränderung des Nutzwertes in Bezug auf den zuvor festgelegten Bereich der Crossover- und Mutationswahrscheinlichkeit sowie der Populationsgröße dargestellt.

Es ist zu erkennen, dass der Bestwert mit einer Crossover-Wahrscheinlichkeit von 90 %, einer Mutationswahrscheinlichkeit von 3 % sowie einer Populationsgröße von 300 Individuen erreicht wird. Dieser liegt bei knapp über 5400 Nutzwertpunkten.

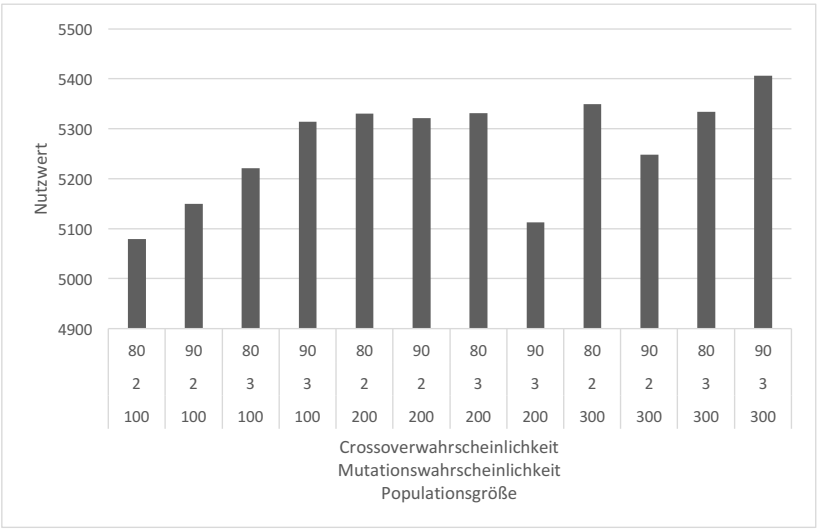


Abbildung 54: Auswertung über den generierten Nutzwert in Abhängigkeit der Populationsgröße (eigene Darstellung)

In Abbildung 55 ist der aufsummierte Zeitaufwand für die Generierung der 100 Lösungskandidaten dargestellt. Es wird deutlich, dass der Zeitaufwand größtenteils durch die Populationsgröße beeinflusst wird und die weiteren Parameter nur eine untergeordnete Rolle bzgl. des Zeitaufwandes spielen. Wie dem Diagramm (Abbildung 55) zu entnehmen ist, benötigt der genetische Algorithmus bereits bei einer Populationsgröße von 300 Individuen länger als das Simulated Annealing Verfahren bei einer Abkühlrate von 0,5 % (vgl. Abbildung 51). Deswegen wird an dieser Stelle auf eine weitere Steigerung der Populationsgröße verzichtet, auch wenn die Ergebnisse theoretisch noch verbessert werden könnten. Es ist jedoch nicht davon auszugehen, dass der Trade-Off zwischen Zeitaufwand und Ergebnisqualität gesteigert werden kann.

Diese Annahme wird auch durch die dargestellten Ergebnisse in Abbildung 56 unterstützt. Aus dem Diagramm geht hervor, dass das beste Nutzen-Aufwandsverhältnis mit einer Populationsgröße von 100 Individuen bei einer Crossover-Wahrscheinlichkeit von 90 % respektive einer Mutationswahrscheinlichkeit von 3 % erzielt wird.

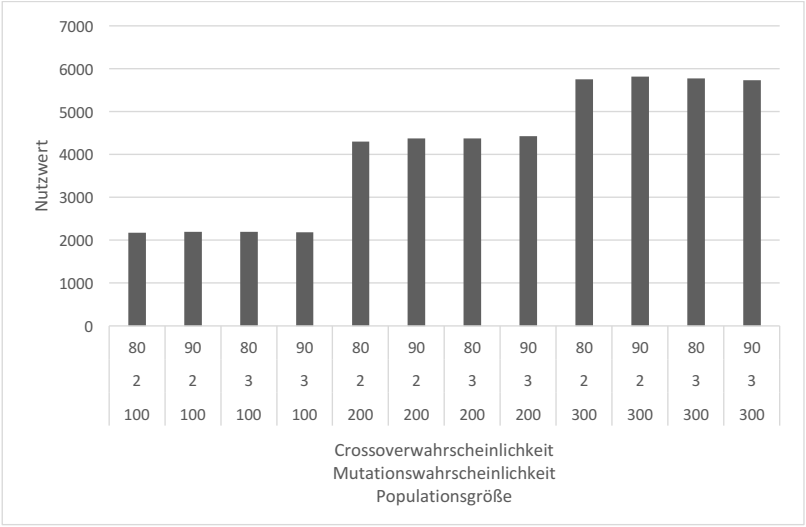


Abbildung 55: Zeitaufwand in Abhängigkeit der Populationsgröße (100 Instanzen, eigene Darstellung)

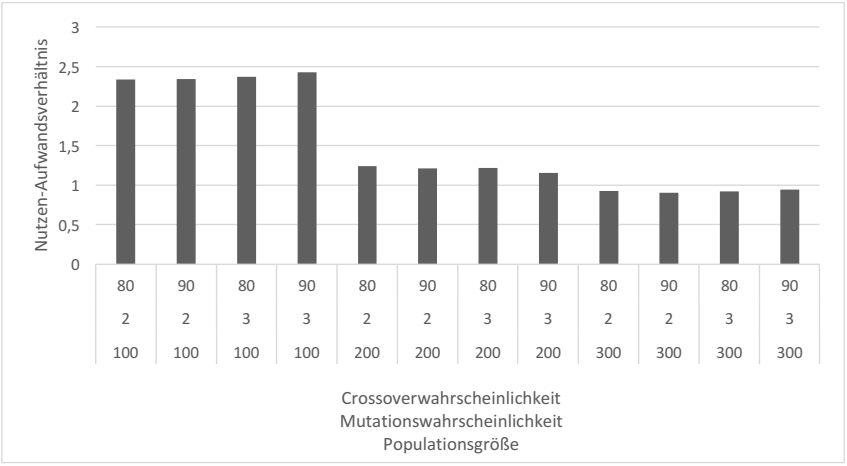


Abbildung 56: Verhältnis von Ergebnisqualität zu Zeitaufwand beim genetischen Algorithmus (eigene Darstellung)

Nachdem die Konfiguration des genetischen Algorithmus abgeschlossen ist, werden im folgenden Unterkapitel die Ergebnisse der hier vorgestellten naturanalogen Verfahren verglichen und bewertet.

8.3.3.4 Evaluation der Verfahren

Als Grundlage für die im Folgenden ausgeführte Evaluation der beiden vorgestellten naturanalogen Verfahren werden die in den jeweiligen Unterkapiteln beschriebenen Konfigurationen verwendet. Dies bedeutet im Falle von Simulated Annealing eine Starttemperatur von 100, eine Abkühlrate von 9 % und eine Verweildauer je Temperaturstufe von 45 samples. Beim genetischen Algorithmus werden folgende Parameter verwendet:

- Populationsgröße: 100
- Mutationswahrscheinlichkeit: 3 %
- Crossover-Wahrscheinlichkeit: 90 %

Für die Evaluation der naturanalogen Verfahren werden die identischen Testfälle, wie bei der Bewertung der heuristischen Ansätze in Kapitel 8.3.2.4, verwendet, um die Vergleichbarkeit sicherzustellen.

Zunächst werden 100 Testinstanzen mit jeweils 20 Product Owner Stories und zwei Entwicklerteams mit einer Kapazität von jeweils 30 Storypunkten bewertet. Bei diesen Testinstanzen wird davon ausgegangen, dass jedes Team theoretisch in der Lage ist je eine Product Owner Story zu bearbeiten. Die Ergebnisse dieser Untersuchung sind im Diagramm in Abbildung 57 dargestellt.

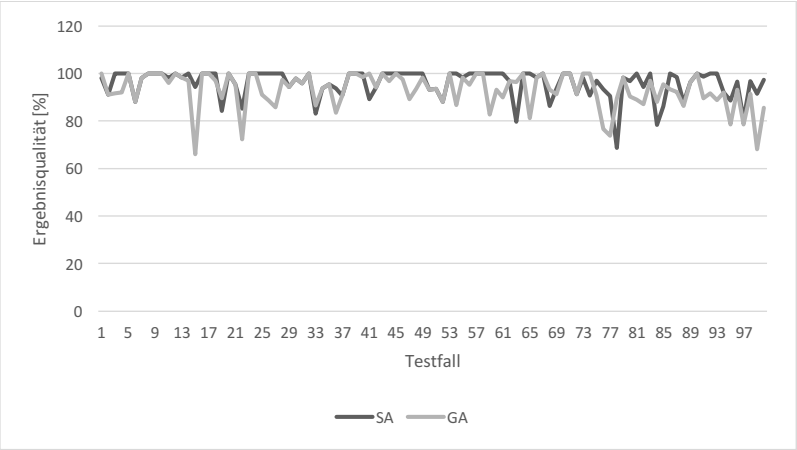


Abbildung 57: Vergleich Simulated Annealing (SA) mit einem gentischen Algorithmus (GA) bei cross-funktionalen Teams

Die Auswertung zeigt, dass beide Verfahren potenziell in der Lage sind optimale Ergebnisse zu erzeugen. Die minimale Ergebnisqualität beim Simulated Annealing liegt bei 68,7 %. Die minimale Ergebnisqualität bei der Nutzung eines genetischen Algorithmus liegt lediglich bei 66,0 %. Die mittlere Ergebnisqualität liegt bei beiden Verfahren oberhalb von 93 %, beim genetischen Algorithmus bei 93,1 % und beim Simulated Annealing bei 95,9 %. Ergebnisse im Be-

reich von 90 % oder mehr können als hinreichend gut angesehen werden, da, wie bereits erörtert, sowohl die Aufwandsschätzung, als auch die Nutzwertbestimmung lediglich Schätzergebnisse sind und der tatsächliche Nutzwert und der tatsächliche Aufwand abweichen können.

In Abbildung 58 wird der aufsummierte Nutzwert über alle 100 Testinstanzen bei komplett cross-funktionalen Entwicklerteams dargestellt und mit dem optimalen Ergebnis verglichen. Es zeigt sich, dass das Simulated Annealing ca. 180 Nutzwertpunkte mehr erzeugt als der vergleichbare genetische Algorithmus.

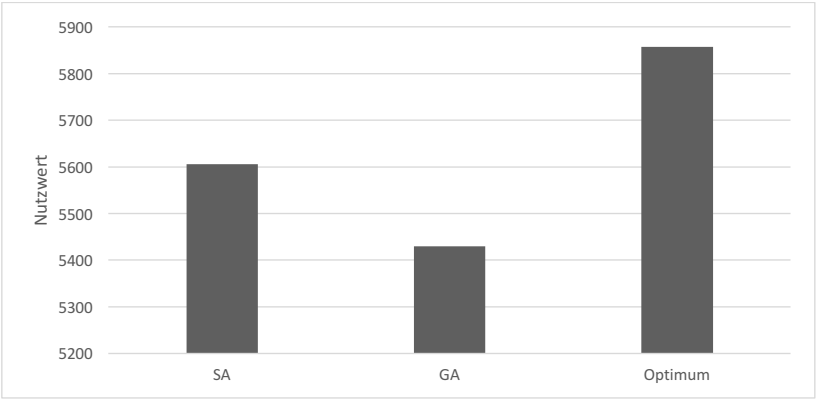


Abbildung 58: Summierter Nutzwert über 100 Testinstanzen, cross-funktional, Vergleich SA mit GA

Im nächsten Schritt der Bewertung der beiden Verfahren werden die Ergebnissqualitäten bei nicht vollständig cross-funktionalen Teams untersucht. Auch bei dieser Auswertung werden die identischen Testfälle verwendet, die bereits bei der Evaluation der heuristischen Verfahren verwendet wurden. Die Ergebnisse sind in Abbildung 59 dargestellt.

Die Ergebnisse zeigen, dass beide Verfahren auch bei nicht komplett cross-funktionalen Entwicklerteams optimale und nahezu optimale Ergebnisse liefern können. Auffällig ist jedoch, dass der genetische Algorithmus in zwei Fällen nicht in der Lage war mit der entsprechenden Konfiguration eine valide Lösung zu finden, somit verbucht dieses Verfahren bei zwei Testinstanzen eine Ergebnisqualität von 0 %. Hier zeigt sich eine Schwäche von naturanalogen Verfahren. Bei diesen Metaheuristiken kann im Regelfall keine Qualitätsgarantie gewährleistet werden. Dementsprechend liegt die durchschnittliche Ergebnisqualität beim genetischen Algorithmus nur noch bei 76,1 %. Auch Simulated Annealing hat Einbußen bei der Ergebnisqualität zu verzeichnen. Die durchschnittliche Ergebnisqualität liegt im Fall von nicht komplett cross-funktionalen Entwicklerteams bei 85,9 %. Dies bedeutet eine Verschlechterung von 7,7 Prozentpunkten.

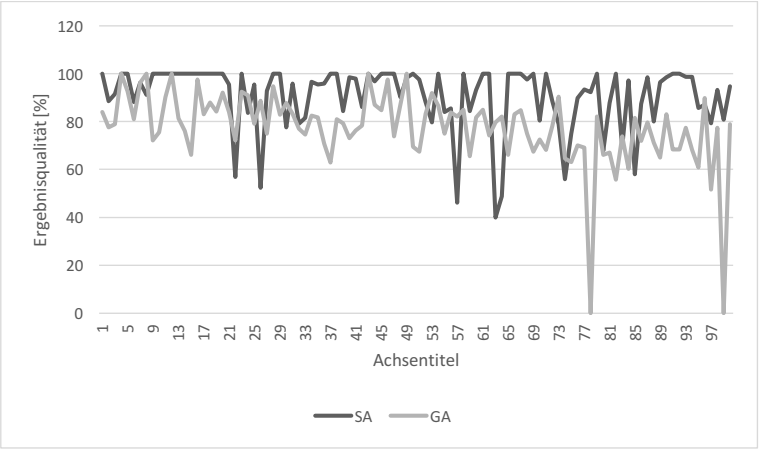


Abbildung 59: Vergleich von SA und GA bei nicht komplett cross-funktionalen Entwicklerteams

In Abbildung 60 wird der summierte Nutzwert über die 100 Testinstanzen analog zu Abbildung 58 dargestellt. Der Abbildung ist zu entnehmen, das beide Verfahren im Vergleich zum Optimum an Qualität verlieren. Der Verlust an Ergebnisqualität ist jedoch beim genetischen Algorithmus deutlich ausgeprägter als bei Simulated Annealing.

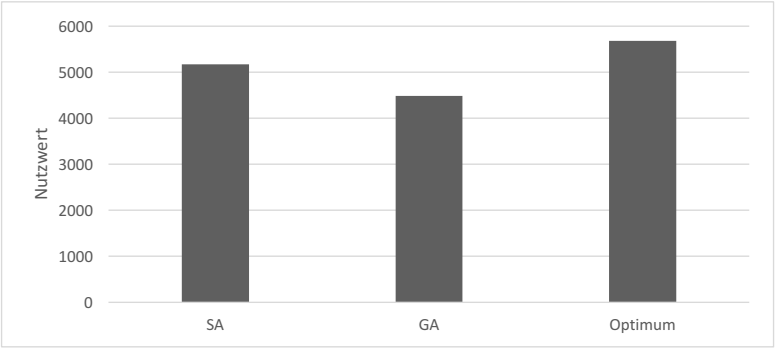


Abbildung 60: Summierter Nutzwert über 100 Testinstanzen, nicht cross-funktional, Vergleich SA mit GA

8.3.3.5 Diskussion der Ergebnisse

Die Evaluation der ausgewählten naturanalogen Verfahren hat gezeigt, dass Simulated Annealing bei den hier durchgeführten Tests bessere Resultate liefert als der hier betrachtete genetische Algorithmus. Sowohl bei cross-funktionalen als auch bei nicht cross-funktionalen Entwicklerteams werden mit Simulated Annealing konstantere und qualitativ hochwertigere Lösungsvorschläge generiert. Dieses spiegelt sich sowohl im aufsummierten erzeugten Nutzwert wieder als auch beim Vergleich der Ergebnisqualitäten je Testfall. Es ist festzustellen, dass der

genetische Algorithmus mit der zuvor festgelegten Konfiguration nur in 98 % der Testfälle überhaupt eine valide Lösung finden konnte.

Auch beim zeitlichen Aufwand zeigt sich, dass Simualted Annealing deutliche Vorteile gegenüber dem hier verwendeten genetischen Algorithmus besitzt. Die durchschnittliche Verarbeitungszeit je Testfall liegt bei ca. 2 Sekunden. Der genetische Algorithmus hingegen benötigt durchschnittlich 14,7 Sekunden je Testinstanz.

Somit kann als Ergebnis der Auswertung der reinen naturanalogen Verfahren festgestellt werden, dass bzgl. der hier durchgeführten Untersuchungen Simulated Annealing als der vielversprechendere Ansatz für die Erzeugung von Lösungskandidaten für das in dieser Arbeit definierte Sprintproblem hervorgeht. Simulated Annealing ist in der Lage bei einem geringeren zeitlichen Aufwand bessere Ergebnisse zu erzeugen als der hier erprobte genetische Algorithmus. Die Ergebnisse bei komplett cross-funktionalen Entwicklerteams liegen oberhalb der Ergebnisse der heuristischen Verfahren. Bei nicht cross-funktionalen Teams zeigt sich jedoch, dass die „Future Fit"-Methode die besten Ergebnisse erzielt. Deswegen wird im nachfolgenden Kapitel 8.3.4 untersucht, ob durch eine Verbindung beider Verfahren eine weitere Verbesserung der Ergebnisse bzw. eine größere Stabilität und Unabhängigkeit der Ergebnisqualität vom Testfall geschaffen werden kann.

8.3.4 Hybride Lösungsansätze

Als hybride Algorithmen werden im Kontext dieser Arbeit Algorithmen beschrieben, die in ihrem Ablauf aus zwei Phasen bestehen. In der ersten Phase werden klassische heuristische Algorithmen durchlaufen, wie sie bspw. in Kapitel 8.3.2 vorgestellt sind. Die Lösungskandidaten werden anschließend in der zweiten Phase als Eingangsgrößen für naturanaloge Verfahren verwendet. Das Ziel der Verbindung beider Verfahren ist, dass die Ergebnisse, die mittels einer Heuristik erzeugt werden, noch verbessert werden können. Zum anderen können so Iterationen innerhalb der naturanalogen Verfahren reduziert werden, da die Suche einer Lösung nicht bei zufällig erzeugten Lösungskandidaten beginnt, sondern bereits ab der ersten Iteration gute Lösungskandidaten zur Verfügung stehen, die lediglich verbessert werden sollen und nicht nach einem kompletten Muster gesucht werden muss.

Bei dieser Synthese von zwei Verfahren wird im Zusammenspiel mit Simulated Annealing zunächst eine Lösung mittels der „Future Fit" Heuristik erzeugt und dieses Ergebnis wird als initialer Input verwendet.

Auch wenn der genetische Algorithmus wegen der stark schwankenden Ergebnisqualität noch nicht überzeugen konnte, wird untersucht, ob durch eine Zuführung der Ergebnisse aus den heuristischen Verfahren in die Startpopulationen die Schwankungen verringert werden und die Ergebnisse generell verbessert werden können, um abschließend doch kompetitiv im Vergleich zu den anderen Methoden zu sein.

8.3.4.1 Evaluation der Ergebnisse

Auch bei dieser Auswertung werden wieder die identischen Testfälle wie in den vorherigen Auswertungen verwendet, um eine Vergleichbarkeit der Ergebnisse sicherzustellen. In Abbildung 61 werden die qualitativen Ergebnisse je Testfall dargestellt.

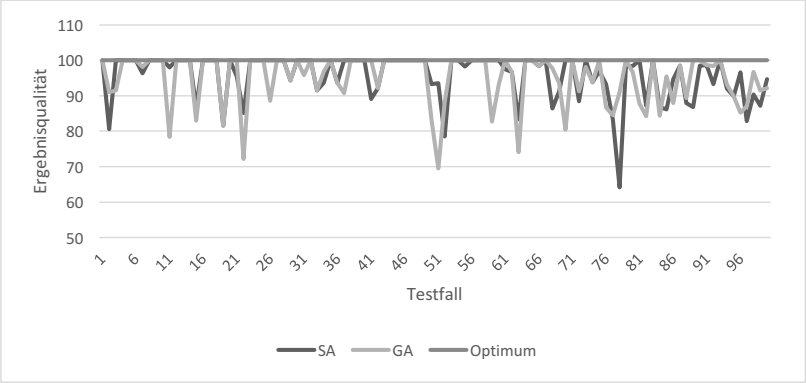


Abbildung 61: Vergleich von 100 Testinstanzen, cross-funktional, hybride Ansätze (eigene Darstellung)

Dem Diagramm ist zu entnehmen, dass die minimale Ergebnisqualität des hybriden genetischen Algorithmus, wie auch die des hybriden Simulated Annealings oberhalb von 62 % liegen. Es zeigt sich, dass die minimale Ergebnisqualität im Vergleich zu den nicht hybriden Ansätzen gesteigert werden konnte (vgl. Abbildung 57).

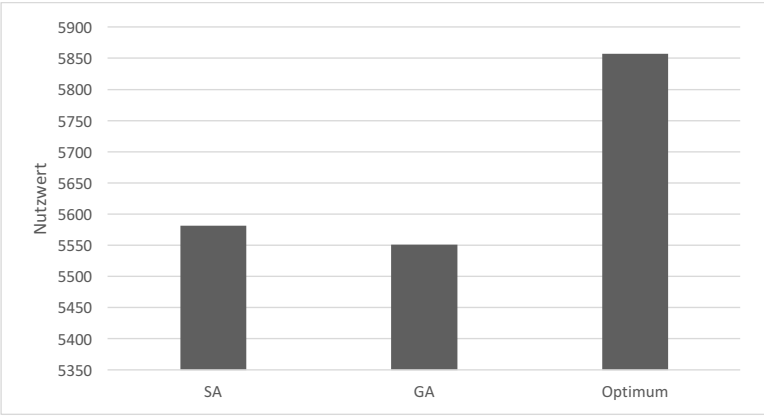


Abbildung 62: Vergleich der aufsummierten Nutzwerte der hybriden Ansätze zum Optimum, cross-funktional (eigene Darstellung)

Abbildung 62 ist zu entnehmen, dass der hybride genetische Algorithmus eine durchschnittliche Ergebnisqualität von ca. 94 % über alle 100 Testinstanzen mit komplett cross-funktionalen Entwicklerteams erreicht. Der hybride Simulated-Annealing-Ansatz weist insgesamt eine Zielerreichung von ca. 95 % auf.

In Abbildung 63 werden die qualitativen Ergebnisse bei nicht komplett cross-funktionalen Teams dargestellt. Hierbei ist festzustellen, dass die minimale Ergebnisqualität des hybriden genetischen Algorithmus lediglich bei knapp über 45 % liegt. Somit ergibt sich bei diesem Testfall eine deutliche Verschlechterung im Vergleich zu den reinen heuristischen Verfahren (vgl. Abbildung 43). Die minimale Ergebnisqualität vom hybriden Simulated Annealing liegt bei den hier bewerteten Testfällen bei ca. 60 %. Auch dieses stellt eine kleinere Verschlechterung gegenüber den qualitativen Ergebnissen der heuristischen Verfahren dar.

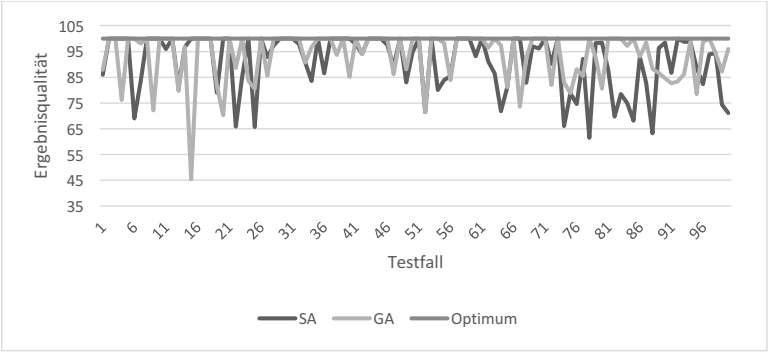


Abbildung 63: Vergleich von 100 Testinstanzen, nicht cross-funktional, hybride Ansätze (eigene Darstellung)

In der nachfolgenden Abbildung 64 werden die summierten Nutzwerte über alle Testinstanzen mit nicht komplett cross-funktionalen Teams dargestellt. Bei diesen Testfällen ist eine Umkehrung der Ergebnisse im Vergleich zu den cross-funktionalen Testfällen zu beobachten. Die Ergebnisqualität des hybriden genetischen Algorithmus übertrifft die Ergebnisqualität des hybriden Simulated Annealing Ansatzes um ca. 2,5 %.

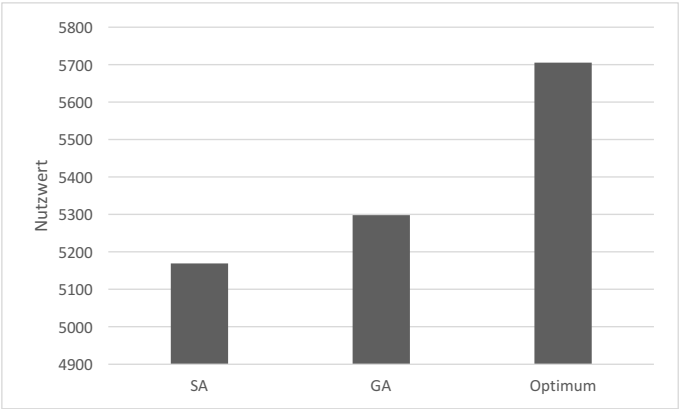


Abbildung 64: Vergleich der aufsummierten Nutzwerte der hybriden Ansätze zum Optimum, nicht cross-funktional (eigene Darstellung)

8.3.4.2 Diskussion der Ergebnisse

Die Ergebnisse aus den Untersuchungen der hybriden Ansätze zeigen, dass kein klarer Favorit zwischen den beiden Verfahren auszumachen ist. Bei komplett cross-funktionalen Teams erzeugt der hybride Simulated Annealing insgesamt mehr Nutzwert als der hybride genetische Algorithmus. Dieses Verhältnis kehrt sich jedoch bei nicht komplett cross-funktionalen Teams um, sodass der hybride genetische Ansatz bessere Lösungsvorschläge liefert. Im Vergleich zu den reinen naturanalogen Verfahren kann jedoch festgestellt werden, dass durch die Verwendung der Lösungskandidaten, die durch die heuristischen Verfahren erzeugt werden, als initiale Startparameter die Stabilität der Ergebnisse verbessert werden kann. Die minimale Ergebnisqualität konnte gesteigert werden, sodass von beiden Verfahren immer ein möglicher Lösungskandidat mit einem Nutzwert größer 0 erzeugt werden konnte. Gleiches gilt für den summierten Nutzwert über alle Instanzen.

Im nachfolgenden Kapitel 8.4 werden alle Untersuchungsergebnisse der unterschiedlichen untersuchten Verfahren zusammengefasst und miteinander verglichen.

8.4 Zusammenfassung

In Kapitel 1 wurde zunächst das Problem der Nutzwertoptimierung für Scrum Sprints mit mehreren Entwicklerteams unter der Verwendung von Product Owner Stories beschrieben. Anschließend wurde aus diesen Vorüberlegungen und Beobachtungen aus dem betrieblichen Kontext ein mathematisches Modell zur Beschreibung valider Sprints abgeleitet und das Optimierungskriterium bestimmt.

In einem weiteren Schritt wurde gezeigt, dass es sich bei dem beschriebenen Problem um ein NP-schweres bzw. NP-vollständiges Problem handelt, je nachdem ob geprüft werden soll ob eine Sprintplanung existiert, die einen Mindestnutzwert erreicht oder nach dem tatsächlichen Optimum gesucht wird. Da Probleme aus dieser Komplexitätsklasse nur für kleine Probleminstanzen durch einen Algorithmus innerhalb eines begrenzten Zeitaufwandes optimal gelöst werden können, wurden daran anschließend unterschiedliche Methoden für die Vorschlagsgenerierung von Iterationen erarbeitet und evaluiert.

Es wurden zunächst heuristische Verfahren erarbeitet und deren Ergebnisqualität an 100 zufällig erzeugten Testinstanzen gemessen. Anschließend wurden zwei naturanaloge Verfahren auf ihre Eignung hin untersucht. Beim Simulated Annealing, wie auch beim genetischen Algorithmus wurden zunächst 100 zufällige Testinstanzen verwendet, um die Stellgrößen, wie die Abkühlrate beim Simulated Annealing oder die initiale Populationsgröße beim genetischen Algorithmus, unter Berücksichtigung der Laufzeit für dieses Problem zu kalibrieren. Anschließend wurden die identischen 100 Testfälle für die Auswertung der Ergebnisqualität dieser so konfigurierten Algorithmen evaluiert.

In einem weiteren Schritt wurden die zuvor erarbeiteten heuristischen und naturanlogen Verfahren kombiniert, sodass die Heuristiken den initialen Input für das naturanaloge Verfahren liefern. Das Anliegen war es, durch die so geschaffenen hybriden Algorithmen eine weitere Ergebnisverbesserung zu erzielen und die stärkeren Schwankungen der naturanalogen Verfahren bei der Ergebnisqualität zu verringern.

In Tabelle 13 werden die Ergebnisse der einzelnen Untersuchungen zusammengefasst.

Tabelle 13: Zusammenfassung der Ergebnisse

Verfahren	Min. (Cross-Funktional)	Durchschnitt (Cross-Funktional)	Min	Durch-schnitt	Durchschnitt (Gesamt)
First Fit	**72,2 %**	**96,5 %**	72,2 %	94,7 %	95,6 %
Best Fit	70,5 %	95,5 %	69,2 %	95,4 %	95,5 %
Future Fit	70,5 %	95,5 %	**75,5 %**	**96,7 %**	**96,1 %**
SA	68,7 %	96,0%	40,0 %	90,7 %	93,4 %
GA	66,0 %	93,1 %	0,0 %	77,6 %	85,4 %
Hybrider SA	64,2 %	95,6 %	61,5 %	91,0 %	93,3 %
Hybrider GA	69,6 %	95,0 %	45,5 %	93,2 %	94,1 %

Die Auswertung zeigt, dass die Future Fit Heuristik als bester Kandidat bei diesen zufälligen Testinstanzen die besten Ergebnisse erzeugt. Bei diesem Verfahren werden für die Distribution der Product Owner Stories auf Entwicklerteams sowohl die verbleibenden Kapazitäten als auch die Aufwände der Product Owner Stories berücksichtigt, die von dem jeweiligen Team unter Berücksichtigung der Kompetenzen des Entwicklerteams umgesetzt werden können. Dieser Algorithmus zeichnet sich sowohl durch eine hohe Robustheit (minimale Ergebnisqualität), als auch durch eine hohe durchschnittliche Ergebnisqualität aus. An dieser Stelle zeigt sich, dass die naturanalogen Verfahren ebenfalls gute durchschnittliche Ergebnisse erzielen, jedoch die minimale Ergebnisqualität nicht garantiert werden kann. Durch die Verwendung von hybriden Algorithmen können diese Effekte zwar abgedämpft werden, allerdings wird die zugrunde liegende Heuristik im Mittel nicht übertroffen, sondern es wird lediglich eine Verbesserung der Ergebnisse in Bezug auf den Basisalgorithmus erzielt. Diese findet beim hybriden genetischen Algorithmus jedoch deutlich ausgeprägter statt als beim Simulated Annealing.

Zusammenfassend wird festgestellt, dass die hier untersuchten naturanalogen Verfahren gute Lösungskandidaten für das Sprintproblem generieren können, ohne die Spezifika des Problems berücksichtigen zu müssen. Dieses ist in der Tatsache begründet, dass diese Verfahren generisch sind und lediglich durch eine auf das Problem angepasste Ergebnisbewertung erweitert werden müssen. Somit ist eine tiefere Analyse des Problems nicht notwendig.

Die eigens für dieses Problem entwickelten Heuristiken liefern jedoch speziell für das Sprintproblem bzgl. der minimalen, wie auch der durchschnittlichen Ergebnisqualität bessere Ergebnisse. Außerdem benötigen die hier vorgestellten Verfahren weniger Rechenzeit als die hier ebenfalls untersuchten naturanalogen Verfahren.

Die erprobten hybriden Ansätze können zwar zu einer Verbesserung der Ergebnisse der reinen naturanalogen Verfahren führen, die Ergebnisse der initialen heuristischen Verfahren werden dabei jedoch nicht, wie vermutet, übertroffen.

Die in diesem Kapitel erzielten Ergebnisse beruhen auf zufälligen Instanzen, die entsprechend dem Vorgehen in Kapitel 8.3.1 erzeugt wurden. Aufgrund der Konstruktion dieser Testfälle kann nicht zwangsweise abgeleitet werden, wie sich die Ergebnisse in Bezug auf reale Projektinstanzen verhalten. Außerdem ist ebenfalls nicht geklärt, welche Ergebnisse reale Product Owner bzw. Projektverantwortliche erzielen.

Für die Erprobung im betrieblichen Kontext, beschrieben in Kapitel 1, wird somit die Future Fit Heuristik verwendet, um eine an die Arbeitsrealität angelehnte Probleminstanz zu lösen. Dieses Ergebnis wird mit dem manuell von Product Ownern mit mehrjähriger Berufserfahrung erzeugten Ergebnissen verglichen um festzustellen, ob durch dieses Verfahren ein tatsächliches Einsparpotenzial erzielt werden kann und der Product Owner dadurch tatsächlich in seiner Arbeit entlastet wird.

Als weitere Erprobung in einem realen Umfeld wird die Planung eines Projektleiters bzgl. eines realen Projektes mit den Ergebnissen des algorithmischen Vorgehens über mehrere Iterationen hinweg verglichen.

Die Forschungsfrage 3.1 bzgl. der Modellierung von Iterationen wird an dieser Stelle bereits beantwortet. Ein entsprechendes mathematisches Modell wurde in Kapitel 8.1 hergeleitet. Innerhalb des Modells ist es möglich Synergiepotenziale auf Basis von Product Owner Stories abzubilden. Darüberhinaus können mehrere Entwicklerteams in die Planung mit einbezogen werden. Es ist möglich die unterschiedlichen Charakteristiken je Team in die Modellierung einzubeziehen.

Kriterien zur Bestimmung der Validität einer Iterationsplanung werden ebenfalls angegeben. Innerhalb der in diesem Kapitel erprobten Optimierungsverfahren wird auch eine automatisierte Validitätskontrolle verwendet. Diese Kontrolle könnte ggf. auch bei einer manuellen Planung zur Unterstützung als nachfolgender Qualitätssicherungsschritt angewendet werden.

9 Erprobung im betrieblichen Kontext

Die in Kapitel 1 beschriebenen Untersuchungen haben gezeigt, dass die Iterationsplanung unter Berücksichtigung von Synergiepotenzialen zwischen Anforderungen und unter Berücksich-tigung mehrerer Rahmenbedingungen effizient durch naturanaloge wie auch heuristische Verfahren gelöst werden kann. Im Vergleich der unterschiedlichen untersuchten Verfahren stellte sich heraus, dass unter den gegebenen Bedingungen die Future Fit Heuristik mit die besten Ergebnisse unter Berücksichtigung der Verarbeitungsgeschwindigkeit und Ergebnis-qualität erzielt.

Aus diesen Untersuchungen kann jedoch weder die Funktionalität des Algorithmus auf realen Instanzen noch die Qualität des Algorithmus im Vergleich zum Menschen bewertet werden. Ziel dieses Kapitels ist es, anhand einer theoretischen Sprintplanung unter der Verwendung von Product Owner Stories zu belegen, dass die algorithmische Herangehensweise an dieses Problem sowohl in Bezug auf den Nutzwert als auch auf den Zeitaufwand zu besseren Lösungen führt, als dieses bei einer manuellen Planung der Fall wäre.

In einer zweiten Untersuchung werden die Ergebnisse des algorithmischen Vorgehens über mehrere Iterationen mit dem eines Projektverantwortlichen verglichen.

In diesem Kapitel wird zunächst der erste Versuchsaufbau genauer erläutert. Anschließend wird die Durchführung dieser Erprobung dokumentiert, bevor abschließend die Ergebnisse zusammengefasst und diskutiert werden.

Der Aufbau der zweiten Erprobung ist dabei analog zu dem beschriebenen Aufbau der ersten. Abschließend werden die Inhalte dieses Kapitels zusammengefasst.

9.1 Erprobung 1

In dieser ersten Erprobung werden die Ergebnisse mehrerer Product Owner mit dem Ergbnis des algorithmischen Vorgehens verglichen. Die Hypothesen für diese Untersuchung sind:

1. Durch das algorithmische Vorgehen ergibt sich eine deutliche Zeitersparnis bei der Bearbeitung einer zufälligen Instanz.
2. Der Algorithmus erzeugt bessere Ergebnisse als die menschlichen Probanden in Bezug auf eine zufällige Instanz.

Zur Überprüfung dieser Hypothesen wird eine quantitative Untersuchung durchgeführt.

9.1.1 Beschreibung des Versuchsaufbaus

Für die Erprobung wurden zunächst manuell 30 User Stories erzeugt und mit einem Nutzwert und teilweise auch mit Abgabefristen versehen. Aus diesen User Stories wurden 40 Product Owner Stories abgeleitet und mit entsprechenden Aufwänden versehen. Außerdem wurde eine Übersicht erstellt, die angibt, welches der beiden theoretischen Teams die entsprechenden Kompetenzen besitzt, um die jeweiligen Product Owner Stories umsetzen zu können. Bei dieser theoretischen Sprintplanung wird davon ausgegangen, dass es sich um nicht komplett cross-funktionale Entwicklerteam handelt. Team A besitzt in diesem theoretischen Szenario eine Kapazität von 30 Storypunkten und Team B eine von 25 Storypunkten.

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0_9

In der folgenden Abbildung 65 ist im oberen Teil eine User Story zu sehen, auf der kenntlich gemacht ist, welchen Nutzwert diese User Story besitzt, welche Product Owner Stories benötigt werden, um sie vollständig zu implementieren und ob die Abgabefrist für die aktuell zu betrachtende Sprintplanung relevant ist. Im unteren Teil der Abbildung ist dazu analog eine Product Owner Story dargestellt, auf der vermerkt ist, wie hoch der Aufwand für die Implementierung ist und welche User Stories mit dieser Product Owner Story assoziiert sind. Die Abgabefrist wird zwecks einer besseren Übersichtlichkeit ebenfalls mit angegeben.

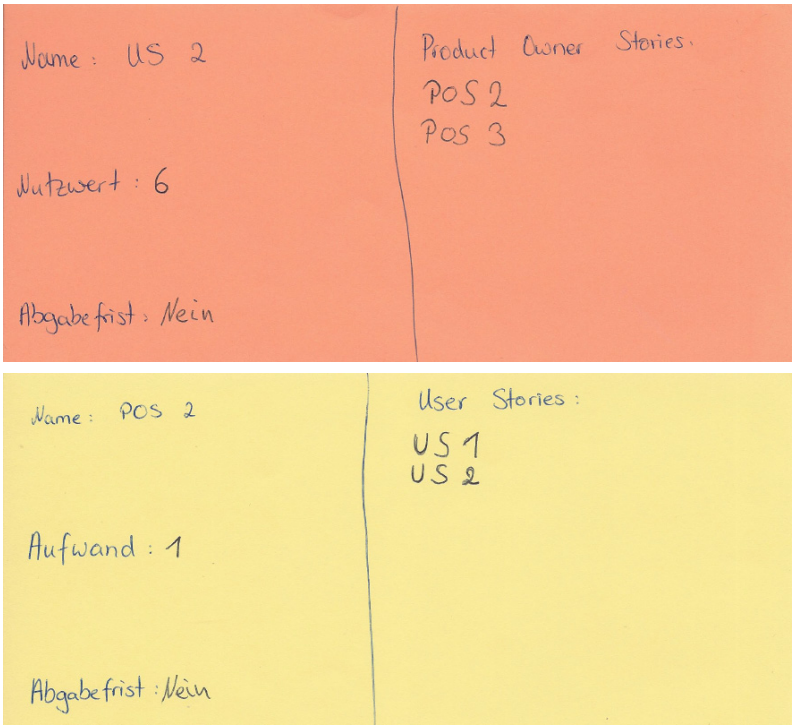


Abbildung 65: Darstellung einer User Story und einer Product Owner Story für die Erprobung (Quelle: eigene Darstellung)

Die Product Owner hatten die Aufgabe, einen validen Sprint unter Berücksichtigung der Rahmenparameter zu planen und dabei den Nutzwert soweit möglich zu maximieren. Zwecks einer besseren Verständlichkeit der Aufgabenstellung wurde an dieser Stelle darauf verzichtet Vorbedingungen mit in die Aufgabenstellung zu integrieren.

Um die Einarbeitungszeit für die Product Owner soweit wie möglich zu reduzieren und direkt eine gute Übersichtlichkeit zu gewährleisten, liegen die User Stories offen aus. Außerdem sind die User Stories sortiert, sodass die User Stories mit eine Abgabefrist ganz oben liegen. Die restlichen User Stories sind nach dem Verhältnis von Nutzen zu Aufwand sortiert.

Die Product Owner Stories liegen ebenfalls offen aus und sind nach ihrer ID fortlaufend sortiert. Außerdem ist die Übersichtsdarstellung, welches Team welche Product Owner Story bearbeiten kann, als Ausdruck an den Product Owner übergeben worden. Die Übersichtsdarstellung kann dem Anhang in Kapitel 12.3 entnommen werden.

Um allen Beteiligten möglichst identische Bedingungen zu verschaffen, wurde im Rahmen der Erprobung zunächst das Konzept von User Stories und Product Owner Stories vorgestellt. Anschließend wurde der Versuchsaufbau erklärt und ggf. entstandene Fragen beantwortet.

Anschließend wurde jedem Probanden Zeit zur Vorbereitung gegeben, in der jeder Proband die Möglichkeit hatte ggf. die Ordnung der User Stories und Product Owner Stories zu verändern, um eine für sich, soweit möglich, „gewohnte“ Ausgangssituation zu schaffen. Nachdem der jeweilige Proband diese Vorbereitungen abgeschlossen und seine Startbereitschaft signalisiert hatte, wurde eine Stoppuhr gestartet um die Zeit für die Bearbeitung messen zu können. Nach der Bearbeitung durch den Product Owner wurde die Stoppuhr angehalten. Anschließend prüften der Autor dieser Arbeit und ein freiwilliger Helfer unabhängig voneinander, ob es sich um einen im Sinne der Rahmenbedingungen validen Sprint handelt. Danach berechneten beide unabhängig voneinander den Nutzwert für den Sprint.

Im folgenden Kapitel werden die Ergebnisse und Beobachtungen dieser Untersuchung aufbereitet und dargestellt.[10]

9.1.2 Ergebnisse der Erprobung

In diesem Abschnitt werden die Beobachtungen und Messergebnisse der Probanden in zusammengefasster Form wiedergegeben, bevor sie in Kapitel 9.1.3 diskutiert und den Ergebnissen der automatisierten Sprinterstellung gegenüber gestellt werden.

Vor der eigentlichen Durchführung wurden das Setting und die Aufgabe von einem Probanden getestet. Da dieser Durchlauf ohne Probleme ablief, mussten keine Änderungen oder Anpassungen am Aufbau vorgenommen werden und die Ergebnisse und Beobachtungen werden ebenfalls mit in die Auswertung eingeschlossen.

Insgesamt haben 21 Personen teilgenommen. Die Teilnehmer wurden dabei in drei Gruppen unterteilt. Die erste Gruppe umfasst IT-Projektleiter mit mindestens 5 Jahren Berufserfahrung. Die zweite Gruppe umfasst alle IT-Projektleiter mit einer Berufserfahrung von weniger als 5 Jahren. Die dritte Gruppe bilden die an dieser Studie beteiligten Studenten.

Aus den Beobachtungen während der Durchführung war es möglich drei vorherrschende Strategien bei der Bearbeitung dieser Aufgabe zu identifizieren. Diese drei unterschiedlichen Strategien werden nun anhand von Beobachtungen jeweils eines Probanden, der diese Strategie wählte, dargestellt.

Bearbeitung von User Stories mit hohem Nutzwert:

Bei Proband A handelt es sich um einen IT-Projektleiter mit einer Berufserfahrung von mehr als fünf Jahren. Auffällig bei dieser Untersuchung ist, dass dieser Proband bei der Erledigung

[10] Diese Untersuchung fand nicht im direkten Arbeitskontext der Volkswagen AG statt. Alle Untersuchungen fanden im privaten und universitären Umfeld statt. Die Teilnahme war zu jederzeit freiwillig. Es bestand auch die Möglichkeit jederzeit abzubrechen und von der Erprobung zurückzutreten.

von vornherein ein sehr strukturiertes Vorgehen hatte. Zunächst wurden alle User Stories identifiziert, die aufgrund von Abgabefristen in diese Iteration eingeplant werden mussten. Anschließend wurden die benötigten Product Owner Stories aus der „Auslage“ entfernt und in einen separaten Stapel abgelegt.

Im nächsten Schritt entfernte der Proband alle User Stories mit einem Nutzwert größer sieben aus der Auslage und sortierte die entsprechenden Product Owner Stories dazu. Auch dieser Stapel wurde separat an die Seite gelegt.

Im Anschluss ordnete der Proband die Product Owner Stories gemäß den vorhandenen Kompetenzen der Teams zu. Er begann mit den Product Owner Stories, die zur Erfüllung der Anforderungen mit Abgabefrist nötig sind. Bei der Zuteilung der Product Owner Stories auf die Teams achtete er darauf, dass die Teams ungefähr gleichmäßig belastet wurden. Dieses Verfahren setzte der Proband anschließend auch mit den weiteren, zuvor zur Seite gelegten User Stories mit hohem Nutzerwert und den zugehörigen Product Owner Stories fort. Nach der Durchführung gab der Proband an, diese Strategie aus zwei Gründen gewählt zu haben:

1. User Stories mit einem hohen Nutzwert müssen mindestens für einen Kunden wichtig sein
2. Zitat: „Was man hat, das hat man.“

Als keine Anforderug aus dem zuvor identifizierten Umfang mehr vollständig eingeplant werden konnte, wählte der Proband weitere Product Owner Stories aus, die direkt zu einer User Story assoziiert sind und Aufwände aufweisen, die die Kapazität der Teams genau auslastet. Er begründete dieses Vorgehen damit, dass die Betrachtung von Blöcken mit mehreren Product Owner Stories zu zeitaufwendig sei und dass durch das gewählte Vorgehen vermutlich ein besseres Verhältnis zwischen Ergebnisqualität und Zeitaufwand erzielt werden könnte.

Der Proband benötigte für die Durchführung 31 Minuten. Das Ergebnis erfüllte alle Rahmenbedingungen und die Lösung des Probanden erzielte einen Nutzwert von 117.

Bearbeitung von Product Owner Stories mit wenig Aufwand:

Proband B identifizierte ebenfalls analog zu Proband A die Anforderungen, welche auf jeden Fall innerhalb des zu planenden Sprints umgesetzt werden müssen. Der Proband teilte die notwendigen Product Owner Stories auf die Teams auf.

Im nächsten Schritt identifizierte der Proband zunächst alle User Stories bzw. Product Owner Stories zwischen denen eine 1 : 1 Relation besteht. Anschließend wählte er die Product Owner Stories aus, die einen Aufwand von weniger als 5 Storypunkten aufweisen, und teilte diese auf die Teams auf. Nach der Durchführung gab der Proband an, dass er sich über das Verhältnis von Nutzen und Aufwand Gedanken gemacht hätte, jedoch keine Zeit in das Errechnen der Verhältnisse investieren wollte. Er folgerte an dieser Stelle, dass Anforderungen, die mit einem geringen Aufwand umgesetzt werden können, meist kein schlechtes Aufwand-Nutzen-Verhältnis haben können.

Anschließend betrachtete der Proband auch Bündel von Product Owner Stories, um eine Anforderung umzusetzen. Auch hier suchte der Proband Bündel mit einem möglichst geringen Aufwand. Insgesamt benötigte dieser Proband 25 Minuten. Er erzielte mit seiner Lösung einen Nutzwert von 97 Nutzwertpunkten.

Verwendung des Verhältnisses von Nutzwert und Aufwand:

Bei Proband C handelt es sich um einen IT Projektleiter mit einer Erfahrung von mehr als fünf Jahren. Dieser Proband identifizierte, wie Proband A, zunächst alle User Stories, die innerhalb

dieser Iteration abgeschlossen werden müssen. Anschließend wurden die Product Owner Stories Entwicklerteams zugeordnet. Bei der Product Owner Story, die sowohl Team A als auch Team B zugeordnet werden kann, wurde zunächst keine Entscheidung getroffen. Diese Story wurde auf einem separaten Stapel gelegt.

Anschließend erstellte der Proband eine Liste bzw. Tabelle mit folgendem Aufbau:

- User Story
- Nutzwert
- Summe der Aufwände aus POS
- Verhältnis von Nutzwert zu Aufwand

Nachdem der Proband diese Tabelle mit einem relativ hohen Aufwand erstellte, identifizierte er die User Stories mit einem vorteilhaften Verhältnis und teilte die dafür benötigten Product Owner Stories auf die Teams auf. Als die aus seiner Sicht „eindeutigen“ Anforderungen in die Iteration eingeplant waren, prüfte er die verbleibenden Optionen, die ein relativ ähnliches Nutzenwert-Aufwand-Verhältnis aufweisen, ob durch die vorherige Bearbeitung bereits teilweise benötigte Product Owner Stories zugeordnet sind, sodass es zu einer Verbesserung des Verhältnisses kommt.

Der Proband verfolgte diese Strategie bis zum Ende. Abschließend füllte er ein Team mit einer Product Owner Story auf, die keinen direkten Nutzwert erzeugte. Die Begründung hierfür war, dass zu diesem Zeitpunkt keine gewinnbringende Product Owner Story mehr für die Einplanung zur Verfügung stand. Diese Product Owner Story wurde nach Aussagen des Probanden mit hinzugefügt um ggf. schon für die nächste Iteration vorzuarbeiten.

Für die Bearbeitung, inklusive der Erstellung der Tabelle, benötigte der Proband 51 Minuten. Seine finale Lösung erreicht einen Wert von 104 Nutzwertpunkten und entsprach allen Rahmenbedingungen.

Nachfolgend werden in mehreren Diagrammen die Ergebnisse konsolidiert und aufbereitet.

Abbildung 66 zeigt, wie sich die 21 Teilnehmer der Studie auf die unterschiedlichen Gruppen verteilen. Bei der Auswahl der Probanden wurde darauf geachtet, dass jede Gruppe mindestens 5 Probanden umfasst, um die Anonymität der Befragten zu sichern.

In Abbildung 67 ist die Nutzungshäufigkeit der zuvor vorgestellten Strategien, unabhängig von der Probandengruppe, dargestellt.

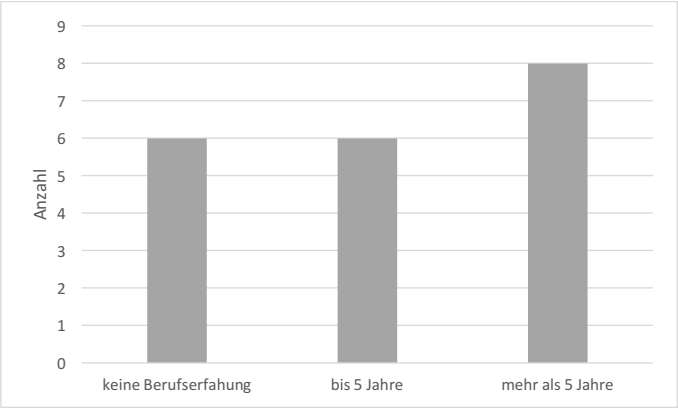


Abbildung 66: Verteilung der Teilnehmer auf die definierten Gruppen (N=21)

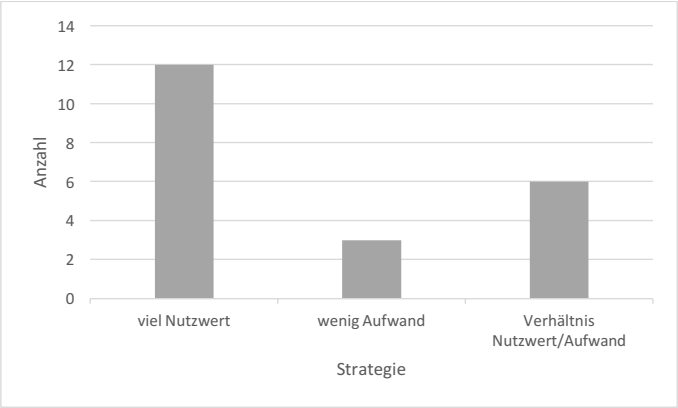


Abbildung 67: Nutzungshäufigkeit der Strategien (N=21)

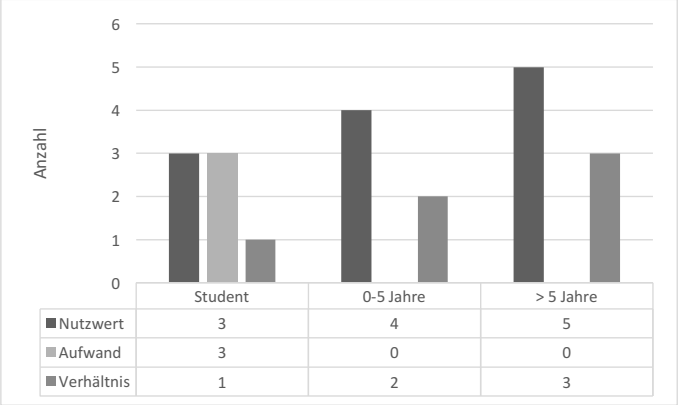
Es zeigte sich, dass mehr als die Hälfte aller Beteiligten eine Strategie verfolgte, die darauf abzielte zunächst die User Stories zu bearbeiten, die einen möglichst hohen Nutzwert liefern. Für die Anwendung dieser Methode wurden unterschiedliche Begründungen geliefert. Einige Probanden gaben an, dass User Stories mit einem hohen Nutzwert für mindestens einen Anforderer von hoher Bedeutung sein müssen, sodass diese bearbeitet werden sollten. Andere gaben an, dass eine Näherung an das „Optimum“ durch die Anwendung dieser Strategie vermutlich am einfachsten zu erreichen sei.

Probanden, die Product Owner Stories bzw. User Stories mit geringen Aufwänden bearbeiteten, gaben als Begründung für die Auswahl dieser Strategie an, dass durch diese Strategie vermutlich die meisten „easy wins“ gemacht werden können. Außerdem wurde erwähnt, dass durch diese Strategie viele (kleinere) Anforderungen umgesetzt werden könnten und nicht nur wenige

aufwendige. Eine weitere Überlegung, die bei den Probanden mit anklang, war, dass Anforderungen mit einem geringen Aufwand per se kein „schlechtes“ Verhältnis von Aufwand und Nutzen haben können.

Teilnehmer, welche das tatsächliche Verhältnis für einzelne User Stories oder expandierte Bündel von Anforderungen berechneten und die Priorisierung der Zuteilung über das Verhältnis regulierten, gaben an, dass dieses zwar ein aufwendiges Verfahren sei, jedoch durch dieses Vorgehen eine gewisse Transparenz innerhalb der Strategie erzielt werden könnte. Aus Sicht der meisten Befragten können durch dieses Vorgehen gute bis sehr gute Ergebnis erzielt und reproduzierbare Ergebnisse erzeugt werden.

In Abbildung 68 ist die Nutzung je Strategie aufgeschlüsselt nach den unterschiedlichen Probandengruppen dargestellt. Bei dieser Auswertung ist auffällig, dass die Strategie, sich über Anforderungen mit einem geringen Aufwand dem Optimum zu nähern, ausschließlich von Studenten angewendet wurde.



	Student	0-5 Jahre	> 5 Jahre
Nutzwert	3	4	5
Aufwand	3	0	0
Verhältnis	1	2	3

Abbildung 68: Nutzung der Strategien je Probandengruppe (N=21)

Den Abbildungen 69 und 70 sind die durchschnittlichen Bearbeitungszeiten in Minuten je Strategie bzw. je Probandengruppe zu entnehmen. Es zeigt sich, dass die Strategien, die zunächst die Anforderungen mit einem hohen Nutzwert bzw. mit einem geringen Aufwand bearbeiten, im Duchschnitt in ca. 27 Minuten durchgeführt werden. Der Lösungsweg über das Verhältnis von Aufwand und Nutzen ist, wie bereits beschrieben, deutlich aufwendiger. Dieses zeigt sich in der durchschnittlichen Bearbeitungszeit. Sie beträgt für diese Strategie im Mittel 46,5 Minuten.

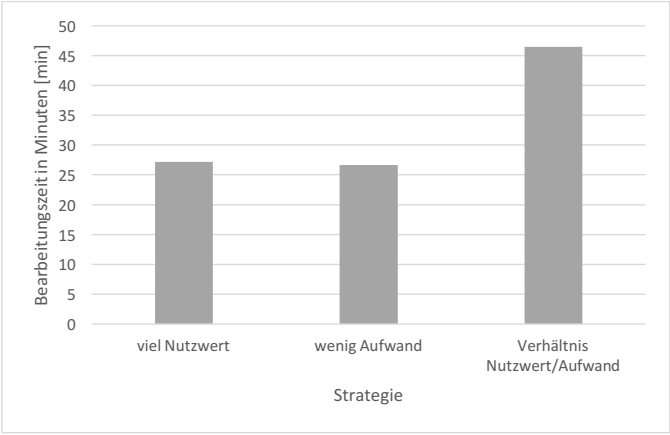


Abbildung 69: Durchschnittliche Bearbeitungszeit je Strategie in Minuten (N=21)

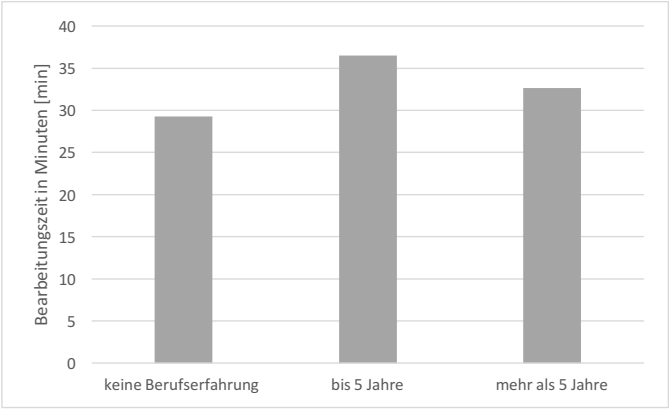


Abbildung 70: Durchschnittliche Bearbeitungszeit je Probandengruppe (N=21)

Die Auswertung zeigt auch, dass Studenten ohne Berufserfahrung für die Bearbeitung der Aufgabe im Schnitt am wenigsten Zeit aufwenden. Für diese Beobachtung können zwei Begründungen aus der Untersuchung abgeleitet werden. Zum einen hat kein Proband aus dieser Gruppe die zeitaufwendige Strategie über das Verhältnis von Nutzen und Aufwand gewählt (vgl. Abbildung 68 und Abbildung 69). Zum anderen zeigt sich, wie der Abbildung 71 zu entnehmen ist, dass lediglich ein Drittel der Studenten ohne Berufserfahrung eine „fehlerfreie“ Lösung konstruierten. Hierbei wurde jedoch nur eine Art von Fehler beobachtet. Die aufgetretenen Fehler beschränkten sich auf Product Owner Stories, die einem Team zugeordnet wurden, welches diese Story bzw. Stories nicht bearbeiten konnte.

Für die weitere Auswertung wurden diese Product Owner Stories aus der jeweiligen Lösung entfernt und nicht in die Berechnung des Nutzwertes mit einbezogen.

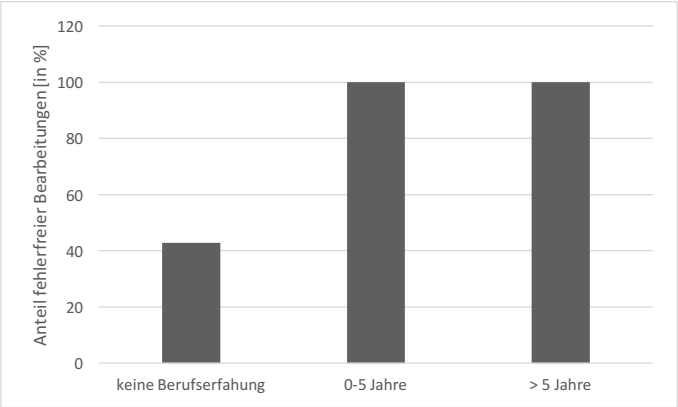


Abbildung 71: Anteil fehlerfreier Bearbeitungen je Probandengruppe(N=21)

In den nachfolgenden Abbildungen 72 und 69 sind jeweils die durchschnittlich erzeugten Nutzenpunkte je Proband und Probandengruppe, bzw. je Proband und Lösungsstrategie angegeben. Die Maxima und Minima sind jeweils durch Striche gekennzeichnet.

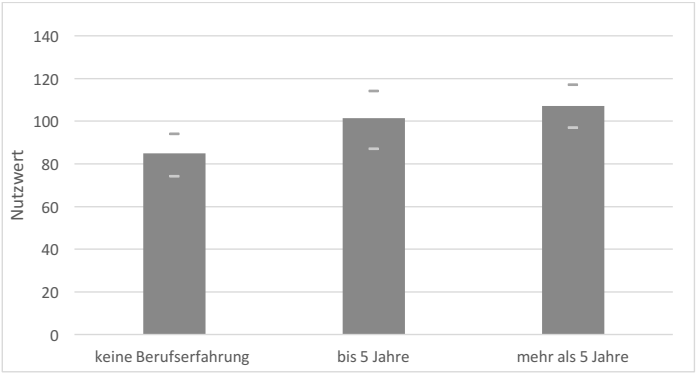


Abbildung 72: Durchschnittlicher, maximaler & minimaler Nutzwert je Probandengruppe (N=21)

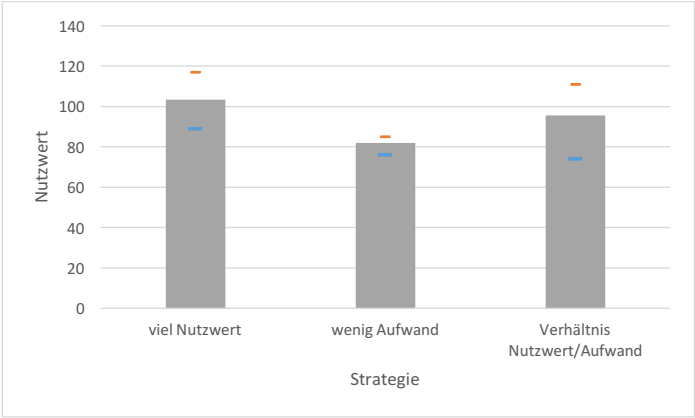


Abbildung 73: Durchschnittlicher, maximaler & minimaler Nutzwert je Strategie (N=21)

Aus der Auswertung der Abbildung 72 wird deutlich, dass es scheinbar einen Zusammenhang zwischen Lösungsqualität und Berufserfahrung gibt. Die Auswertung legt nahe, dass Personen die über ein größeres Erfahrungswissen verfügen, sich schneller auf diese neue Problemstellung einstellen und ihre bisherigen Erfahrungen einbringen können.

Die Auswertung der unterschiedlichen Strategien in Abbildung 73 zeigt, dass für einen Menschen die Strategie, zunächst die Anforderungen zu bearbeiten, die einen möglichst hohen Nutzwert besitzen, tendenziell die besten Ergebnisse liefert. Dieses gilt für den Durchschnitt, als auch für den best case sowie für den worst case. Probanden, die sich über das Verhältnis von Aufwand und Nutzwert der Lösung nähern, zeigen, dass die Ergebnise im besten Fall auch die Ergebnisse der Nutzwertstrategie erreichen können. Die Streuung der Ergebnisse ist jedoch deutlich größer und auch die durchschnittliche Ergebnisqualität liegt unter der der Strategie Anforderungen mit einem hohen Nutzwert zu bearbeiten. Die „Probleme“, die bei dieser Strategie zu erkennen waren, sind zum einen, dass die Verhältnisse teilweise nur überschlägig gerechnet wurden und zum anderen, dass die einmal berechneten Verhältnisse nicht mehr aktualisiert wurden. Dieses ist jedoch notwendig, da durch die Allokation von Product Owner Stories, die zu einer User Story gehören, auch Aufwände anderer User Stories mit bearbeitet werden können. Dementsprechend müsste das Verhältnis auch nachträglich nach oben korrigiert werden.

Die Strategie, mit möglichst wenig Aufwand zu einem guten Ergebnis zu kommen, scheint eine der stabilsten Methoden zu sein, da die Abweichungen zwischen dem besten Ergebnis und dem Ergebnis mit dem geringsten Nutzwert am dichtesten beieinander liegen. Jedoch ist die Lösungsqualität dieser Strategie im Vergleich zu den anderen Strategien deutlich geringer. Außerdem ist zu beachten, dass es für dieses Vorgehen die wenigsten Beobachtungen gibt, sodass die hier vorgestellten Ergebnisse bei einer größeren Anzahl an Beobachtungen sich noch stärker ändern könnten (vgl. Abbildung 67).

Im nachfolgenden Abschnitt 9.1.3 werden die hier beschriebenen Ergebnisse mit den Ergebnissen des algorithmischen Vorgehens verglichen und etwaige Empfehlungen abgeleitet.

9.1.3 Diskussion der Ergebnisse

Die Auswertung der Erprobung zeigt, dass die Berufserfahrung und das damit verbundene Erfahrungswissen eine Auswirkung auf die Qualität der Ergebnisse hat. Die Auswertung zeigt auch, dass die Erfahrung wenig Auswirkung auf die Bearbeitungszeit hat. Die gewählte Strategie und die Sorgfalt bei der Bearbeitung sind hierfür die ausschlaggebenden Kriterien.

Um einen Vergleich zu dem in Kapitel 7 beschriebenen Algorithmus erzielen zu können, wurde die von allen Teilnehmern bearbeitete Probleminstanz in eine für den Algorithmus bearbeitbare Form überführt und anschließend mittels der „Futur Fit Heuristik“ gelöst. Der Algorithmus erzeugt für das betrachtete Problem eine Iteration, die insgesamt 120 Nutzenpunkte erreicht. Die Laufzeit inklusive des Einlesens des Problems beträgt ca. 4 Sekunden.

Dies bedeutet, dass die algorithmische Herangehensweise an dieses Problem zum einen in der Lage ist, wesentlich schneller eine mögliche Iteration zu planen, als Menschen dazu in der Lage ist. Zum anderen sind die Ergebnisse, die vom Algorithmus erzeugt werden, besser als die, die von einem Probanden erzeugt werden. Im Vergleich dazu liegt die beste menschliche Lösung bei 117 Nutzenpunkten und einer Bearbeitungszeit von ca. 30 Minuten. Bei einer sich ergebenden durchschnittlichen Bearbeitungszeit von ca. 33 Minuten und einem durchschnittlichen Nutzwert von 98 Nutzenpunkten ergibt sich eine Ergebnisverbesserung von ca 20 % bei einer Zeitersparnis von 99,8 %. Darüber hinaus wird der vom Algorithmus erzeugte Lösungsvorschlag so generiert, dass er immer den gegebenen Rahmenbedingungen entspricht, sofern diese erfüllbar sind.

Bei dieser Auswertung ist zu beachten, dass bewusst auf die Modellierung von Abhängigkeiten im Sinne einer Vorgänger-Nachfolger-Beziehung verzichtet wurde, damit die Aufgabe mit einem für die Probanden begrenzten Zeitaufwand zu erledigen war. Es ist davon auszugehen, dass durch eine Hinzunahme dieser Rahmenbedingung die Bearbeitungszeit weiter ansteigen würde und auch die Fehlerquote.

Mit dieser Untersuchung sollte gezeigt werden, dass ein algorithmisches Vorgehen bei der Iterationsplanung Vorteile bzgl. der Verarbeitungszeit liefert und dabei mindestens die Ergebnisqualität eines Menschen erreicht wird. Die Studie hat gezeigt, dass das algorithmische Vorgehen zur Iterationsplanung für erfahrene Product Owner sowohl schnellere, als auch qualitativ vergleichbare Ergebnisse liefert. Für Probanden mit einem geringeren Erfahrungswissen ergibt sich nicht nur ein Potenzial hinsichtlich der Bearbeitungszeit, sondern auch bzgl. der Ergebnisqualität. Somit kann zusammenfassend festgestellt werden, dass das in Kapitel 7 erarbeitete algorithmische Verfahren die an ihn gestellten Ansprüche im Hinblick auf Verarbeitungsgeschwindigkeit und Ergebnisqualität erfüllt.

Abschließend lässt sich feststellen, dass die beiden zu Beginn des Kapitels formulierten Hypothesen als bestätigt angesehen werden können.

9.2 Erprobung 2

In der zweite Erprobung wird das algorithmische Vorgehen mit dem eines Projektleiters anhand eines realen Projektes über mehrere Iterationen hinweg verglichen. Diese Erprobung ist notwendig, da noch nicht geklärt ist, ob die Ergebnisse der zufällig erzeugten Instanzen übertragbar sind.

Hierfür wurde ein Projekt im Bereich des Wissensmanagements ausgewählt. Das bestehende System soll durch ein neues System ersetzt werden. Hierfür wurden initial 93 User Stories im Backlog erzeugt. Anhand dieses Projektes wird überprüft, ob die Ergebnisse in realen Projektsituationen mit denen aus den zufälligen Instanzen näherungsweise übereinstimmen. Somit ergibt sich für diese Untersuchung folgende Hypothese:

1. Der Algorithmus erzeugt intial auf realen Instanzen einen höheren Nutzwert als der Mensch.

Es wird bei dieser Hypothese von einem initialen Product Backlog ohne Veränderung über die Zeit ausgegangen. Planen sowohl der Algorithmus als auch der Projektverantwortliche genügend Iterationen, werden alle Product Owner Stories abgearbeitet. Spätestens zu diesem Zeitpunkt haben sowohl der Algorithmus, als auch der Projektverantwortliche den identischen Nutzwert erzielt.

Es ist zu beachten, dass ein größerer Anteil an Softwareprojekten aus unterscheidlichen Gründen vorzeitig abgebrochen wird (Dr. Wilma Heim 1999; Standish Group International 2014). Aufgrund dieser Tatsache ist vor allem der initiale Nutzwertzuwachs von großer Bedeutung, da er die Investionen des Kundens auch bei einem Abbruch des Projektes bestmöglich schützt.

In den folgenden Unterkapiteln wird zunächst der Versuchsaufbau genauer beschrieben. Anschließend werden die Ergebnisse der Erprobung zusammengefasst und diskutiert.

9.2.1 Versuchsaufbau

Zu Beginn des Projektes, welches im Bereich des konzernweiten Wissensmanagements angesiedelt ist, wurde durch die Anforderer ein initiales Product Backlog erstellt. Dieses Backlog umfasst 93 User Stories.

Diese User Stories wurden durch die Anforderer bzgl. ihres Nutzwertes bewertet. Eine Bewertung des Aufwandes erfolgte initial durch das Entwicklerteam. Es identifizierte ebenfalls Abhängigkeiten und mögliche Synergien. Aus diesen Synergien wurden anschließend Product Owner Stories abgeleitet, welche erneut im Aufwand bewertet wurden. Das Vorgehen entspricht dabei dem Vorgehen, welches in Kapitel 7 beschrieben wurde.

Dieses so erstellte Backlog bildet die Grundlage für die Planung des Projektverantwortlichen wie auch die für das algorithmische Vorgehen. Der verantwortliche Projektleiter weist eine Erfahrung von mehr als 8 Jahren auf.

Die Rahmenbedingungen bei der Planung sind:

- 3 Iterationen werden geplant
- 1 cross-funktionales Team[11]
    - Kapazität: 160 Storypunkte
- Anforderungen haben keine Abgabefrist
- 93 User Stories
- 95 Product Owner Stories
- 25 modellierte Abhängigkeiten

[11] Auch im Fall von nur einem Team ist das Sprintproblem NP-schwer (s. Kapitel 8.2)

Darüberhinaus wurde die planende Person nicht im Vorfeld darüber informiert, dass die zu erstellende Planung als Referenzplanung verwendet wird[12]. Dieses Vorgehen ist insofern gerechtfertigt, da eine reale Projektsituation gelten muss. Es soll verhindert werden, dass die planende Person eine von der „Norm“ abweichende Planung durchführt.

9.2.2 Ergebnisse der Erprobung

Nachdem die Planung von drei Iteration sowohl durch den Projektleiter als auch durch den Algorithmus abgeschlossen war, wurden die Nutzwerte je Iteration bestimmt. Abbildung 74 stellt die Ergebnisse dar.

Der Vergleich des gesamten Nutzwertes, der über die Laufzeit von 3 Iterationen erzielt wird, zeigt, dass das algorithmische Vorgehen ca. 21 % mehr Nutzwert generiert. In der ersten Iteartion wird durch die algorithmische Planung ca. 79 % mehr Nutzwert erzielt. Bei der Betrachtung der einzelnen Iterationen zeigt sich, dass sich diese Differenz durch die Planung der ersten Itartion ergibt. In den folgenden Iterationen erzeugt der Algorithmus weniger Nutzwert als die Planung des realen Projektleiters. Dies bedeutet jedoch nicht, dass der Algorithmus „schlechtere“ Planungen erstellt. Die unterschiedlichen Ergebnisse sind durch die unterschiedliche Planungsbasis begründet. Aufgrund der Planung der ersten Iteration standen dem Algorithmus wie auch dem real planenden Menschen unterschiedliche Startbedingungen zur Verfügung. Außerdem ist Abbildung 74 zu entnehmen, dass die reale Planung einen kontinuierlicheren Nutzwertzuwachs aufweist.

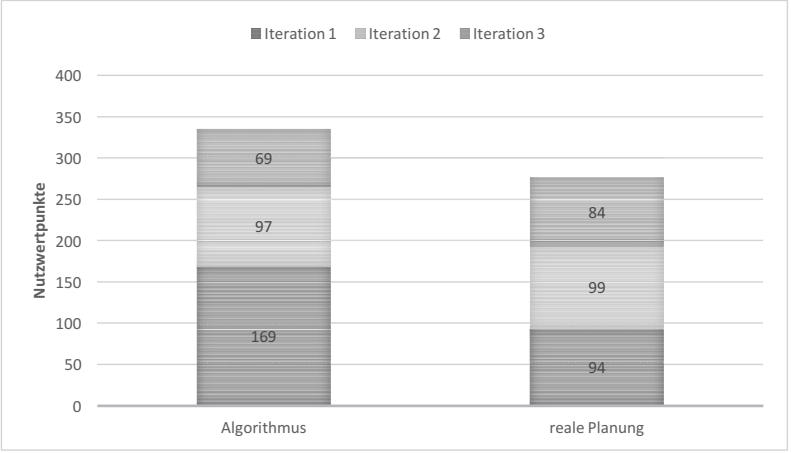


Abbildung 74: Vergleich des Nutzwertzuwachses

In einem weiteren Schritt wurden die „Ähnlichkeiten“ bzw. die Unterschiede bei der Planung analaysiert.

[12] Nach Abschluss der Planung wurde die Erlaubnis nachträglich eingeholt.

Zunächst wurde die Ähnlichkeit der beiden Mengen bestimmt. Ein Maß für diese Ähnlichkeit ist z.B. der Jaccard Index. Dieser Index wird mit der folgenden Formel berechnet (Levandowsky und Winter 1971):

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Für die hier betrachteten Mengen von User Stories ergibt sich ein Jaccard-Koeffizient von ca. 0,55. Dieses ist ein Indiz dafür, dass die beiden Mengen einen gemeinsamen Kern aufweisen.

Eine detaillierte Betrachtung zeigt, dass beide Planungen die essentiellen Anforderungen an das zu erstellende Wissensmanagementsystem erfüllen. Beide Planungen beinhalten die „notwendigen" Anforderungen, die es einem späteren Nutzer erlauben, neues Wissen ins System einzugeben, dieses zu suchen und übersichtlich angezeigt zu bekommen.

Aus dem Vergleich der beiden Planungen ergibt sich, dass die reale, menschliche Planung dazu tendiert Iterationen mit Anforderungen aus wenigen „Themenblöcken" zu bilden. Dies bedeutet, dass bei dieser Planung teilweise auf Nutzwert verzichtet wurde, um thematisch zusammenhängende Anforderungen zu bearbeiten.

Aufgrund dieser Beobachtung wurde das entsprechende Entwicklerteam, bestehend aus vier Personen, gefragt, ob es aus ihrer Sicht einen Vorteil bei der Bearbeitungsgeschwindigkeit durch die Bildung dieser Blöcke gibt. Alle vier Entwickler gaben an, dass der Wechsel zwischen unterschiedlichen Themenbereichen innerhalb einer Applikationen zu keinen Vor- oder Nachteilen führt. Sie ergänzten jedoch, dass es sehr wohl bei Wechseln in andere Applikationen, die unterschiedliche Technologien oder Programmiersprachen einsetzen, zu Verzögerungen aufgrund einer erhöhten Einarbeitungszeit kommt.

9.2.3 Diskussion der Ergebnisse

Die Auswertung der Ergebnisse hat gezeigt, dass mit dem in dieser Arbeit abgeleiteten Modell reale Iterationenen geplant werden können. Die Foschungshypothese für diese Erprobung hat sich ebenfalls bestätigt. Innerhalb der ersten Iteration wird durch den Algorithmus ca. 79 % mehr Nutzwert generiert als bei der realen Planung. Auch bei einer Planung über drei Iterationen zeigt sich, dass das algorithmische Vorgehen noch ca. 21 % mehr Nutzwert generiert. Hierbei ist anzumerken, dass Veränderungen im Backlog über die Zeit an dieser Stelle nicht mit berücksichtigt wurden.

Aus den Ergebnissen kann abgeleitet werden, dass durch die algorithmische Planung ein besserer Schutz der Investitionen des Anforderers entsteht. Sollte es zu einem frühzeitigen Abbruch des Softwareentwicklungsvorhabens kommen, werden im Fall des algorithmischen Vorgehens Softwareartefakte übergeben, die für den Kunden einen höheren Wert haben als bei der traditionellen Planung.

Die Erprobung innerhalb eines realen Softwareentwicklungsprojekts hat gezeigt, dass ggf. eine thematische Bündelung von Anforderungen innerhalb einer Iteration mit berücksichtigt werden sollte. Auch wenn die Entwickler selbst keine größeren Einbußen durch Themenwechsel vermuten, wäre dieses ein Punkt für weitere Forschungen. Sollte sich bewahrheiten, dass dieses für andere Projekete ein kritischer Erfolgsfaktor ist, so müsste dass Modell zur Iterationsplanung entsprechend erweitert werden.

Aufgrund der geringen Anzahl an realen Projekten, die im Zusammenhang mit dieser Erprobung untersucht werden konnten, ergeben sich Einschränkungen bzgl. der Aussagekraft und Übertragbarkeit der Ergebnisse. Um die Ergebnisse, die hier im Rahmen eines Projektes entstanden sind, weiterführend zu validieren, ist es notwendig vergleichbare Erhebungen über einen längeren Zeitraum in unterschiedlichen Projekten durchzuführen. Dadurch kann zum einen die Anwendbarkeit im industriellen Kontext weiterführend überprüft werden und zum anderen können weitere Erfahrungen und Ergebnisse über die Qualität der Algorithmen gesammelt werden.

9.3 Zusammenfassung

Ausgehend von den Ergebnissen aus Kapitel 8 wurden weitere Erprobungen durchgeführt. Aufbauend auf diesen algorithmischen Ergebnissen ist in den weiteren Erprobungen der Mensch als planende Instanz mit in die Betrachtungen eingeschlossen worden. In diesem Zusammenhang ergaben sich zwei Untersuchungen.

In der ersten Erprobung (s. Kapitel 9.1) erfolgte ein Vergleich zwischen den Ergebnissen des algorithmischen Vorgehens zu dem von Menschen. An dieser Studie beteiligten sich insgesamt 21 Personen. Bei einer zufälligen Testinstanz erreichten die Ergebnisse der menschlichen Vorgehensweise nicht das Ergebnis des Algorithmus. Außerdem zeigte die Studie, dass mit zunehmender Erfahrung der Projektverantwortlichen die Qualität der Ergebnisse tendenziell zunimmt. Außerdem konnten drei „Muster“ beim menschlichen Vorgehen identifiziert werden. Diese Muster können ggf. für eine weitere Verbesserung der algorithmischen Ergebnisse verwendet werden.

Im Rahmen einer zweiten Untersuchung (s. Kapitel 9.2) wurde das initiale Product Backlog eines realen Projektes verwendet. Anhand dieser Eingabe konnte zum einen abgeleitet werden, dass reale Projektinstanzen mit dem erarbeiteten Modell abgebildet werden können. Zum anderen konnte auch gezeigt werden, dass sowohl initial als auch über einen Zeitraum von mehreren Sprints hinweg mithilfe des Algorithmus Iterationen geplant werden können. Über eine Zeitspanne von drei Iterationen ergibt sich ein Nutzwertzuwachs von 21%. Wird lediglich die initiale Iteration betrachtet, ergibt sich ein Plus von etwa 80 % in Bezug auf die menschliche Planung.

Allgemein kann festgestellt werden, dass in Bezug auf die hier vorliegenden Testbedingungen das algorithmische Vorgehen vergleichbare oder bessere Ergebnisse vorschlagen kann als der Projektverantwortliche. Für eine Verallgemeinerung der Ergebnisse, vor allem in Bezug auf reale Projekte, ist es notwendig weitere Daten aus unterschiedlichen Projekten und Projektsituationen zu sammeln und zu analysieren.

10 Ausblick und Fazit

Diese Arbeit wurde motiviert durch die steigende Bedeutung von agilen Softwarenentwicklungsmethoden im industriellen Kontext. Hierbei wurden unterschiedliche Herausforderungen bei der Anwendung und Verwendung dieser Methoden identifiziert. Für einige dieser Herausforderungen gibt es jedoch bereits schon unterschiedliche Lösungskonzepte innerhalb der Literatur, sodass das Ziel dieser Arbeit in der systematischen Nutzung von sich ergebenden Synergiepotenzialen ihren Schwerpunkt gefunden hat.

Zunächst wurde die fehlende Möglichkeit, Synergiepotenziale zwischen IT-Anforderungen unterschiedlicher Anforderer zu dokumentiern, als ein Verbesserungspotenzial identifiziert. Um einem Product Owner eine systematische Abbildung dieser Potenziale innerhalb agiler Softwareentwicklungs- bzw. Projektmanagementmethoden, wie Scrum, zu ermöglichen, wurde ein neues Produktrückstandselement eingeführt. Dieses neue Produktrückstandselement fügt sich dabei zwischen die „klassischen" User Stories und Aufgaben (Tasks) ein. Somit ist es für den Kunden weiterhin möglich „seine" Anforderung zu verfolgen und für die IT besteht die Möglichkeit, diese Anforderung mit weiteren Anforderungen zu kombinieren, um Synergieeffekte nutzen zu können, die zu Einsparungen führen können. Ein weiterer Vorteil dieses Ansatzes ist, dass dieser dynamisch innerhalb eines Projektes, je nach Projektsituation, mit aufgenommen oder auch wieder herausgenommen werden kann. Die dafür notwendigen Anpassungen am Scrum-Rahmenwerk wurden ebenfalls beschrieben. Ein Ergebnis dieser Beschreibung ist, dass es je nach vorliegender Projektsituation notwendig sein kann ein weiteres Meeting mit in den regulären Sprintablauf zu integrieren.

Desweiteren wurde die Iterationsplanung für große IT-Projekte mit mehreren Entwicklerteams als weiteres Potenzial identifiziert. Um Product Ownern eine softwareseitige Unterstützung bei dieser Thematik bieten zu können, musste zunächst die Frage nach der Nutzenbewertung von Anforderungen geklärt werden. Hierbei ist ein Vorgehen unter der Verwendung des Analytischen Hierarchieprozesses (AHP) entstanden, welches es ermöglicht, den Nutzwert einzelner Anforderungen durch die Nutzenbewertung unterschiedlicher Anforderer zu ermitteln und entsprechend zu aggregieren.

Aufbauend auf dieser Nutzenbewertung von Anforderungen und den sich durch die systematische Dokumentation von Synergiepotenzialen geänderten Rahmenbedingungen für die Iterationsplanung wurde ein Modell zur Beschreibung valider Iterationsplanungen abgeleitet. Im nächsten Schritt fand dieses Modell Verwendung, um daraus ein Optimierungsproblem abzuleiten. Zur Lösung dieses NP-schweren Problems wurden unterschiedliche Lösungsverfahren auf Basis von Heuristiken und naturanalogen Verfahren entwickelt. Anschließend foglte ein Vergleich der Ergebnisse der unterschiedlichen Verfahren anhand von 100 zufälligen Testinstanzen.

Es zeigte sich, dass sowohl die heuristischen, als auch die naturanalogen Verfahren in der Lage sind, das Problem effizient zu lösen. Der Vergleich der Verfahren ergab, dass eine speziell auf das Problem angepasste Heuristik die besten Ergebnisse bzgl. der 100 zufälligen Testinstanzen liefert. Bei dieser Evaluation hat sich die sogenannte „Future Fit Heuristik" als beste Methode zur Lösung des definierten Problems herausgestellt. Die Zielerreichung lag im Mittel über alle 100 Testinstanzen bei über 96 %.

Anschießend wurde mittels einer Studie, an der 21 Probanden mit unterschiedlichen Vorerfahrungen teilgenommen haben, gezeigt, dass die automatische Vorschlagsgenerierung für Iterationen zwei Vorteile gegenüber der klassischen vollständig durch Menschen erstellten Iteartionsplanung besitzt. Der erste Vorteil ist, dass die Zeit, die für die Iterationsplanung benötigt wird,

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0

deutlich gesenkt werden kann. Der in großen Projekten stark ausgelastete Product Owner kann somit entlastet werden und erhält die Möglichkeit die frei werdende Kapazität durch andere Tätigkeiten gewinnbringend zu nutzen.

Der zweite Vorteil, der mit der entwickelten automatischen Iterationsplanung einhergeht ist, dass die Qualität der Iterationen bzgl. des Nutzwertes, welcher innerhalb einer Iteration erzeugt wird, steigt. Dieser Effekt lässt sich vor allem bei Personen mit einer geringen Berufserfahrung feststellen. Bei erfahrenen IT-Projektleiter bzw. Product Ownern lassen sich auch Verbesserungen feststellen, jedoch fallen diese durchschnittlich nicht so groß aus, wie bei Personen mit geringerer Erfahrung.

In einer zweiten Erprobung wurde das initiale Product Backlog eines realen Projektes verwendet. Anhand eines realen Projektes konnte festgestellt werden, dass mit dem abgeleiteten Modell reale Iterationen geplant werden können. Bei der Planung zeigt sich, dass vor allem initial deutlich mehr Nutzwert erzeugt werden kann und so die Investitionen eines Kunden bzw. Anforderers bei einem Projektabbruch besser geschützt werden. Über einen Planungszeitraum von drei Iterationen konnte durch die automatisierte Planung ein Nutzwert-Plus von 21 % erzielt werden.

Zusammenfassend sind im Rahmen dieser Arbeit drei Hauptergebnisse entstanden. Zunächst wurde ein Vorgehen entwickelt, dass es ermöglicht den Nutzwert für Anforderungen in Zusammenarbeit mit unterschiedlichen Anforderern zu bestimmen. Als weiteres Ergebnis ist ein neues Produktrückstandselement definiert worden, durch welches Synergiepotenziale systematisch innerhalb der agilen Softwareentwicklung abgebildet werden können. Als drittes Ergebnis ist ein angepasstes Modell einer validen Iteration entstanden und daraus ein entsprechend Optimierungsproblem abgeleitet worden, welches sich mit heuristischen und naturanalogen Verfahren lösen lässt. Die aus einem Vergleich hervorgegangene beste algorithmische Lösungsstrategie wurde anschließend verwendet, um zu zeigen, dass die Ergebnisse qualitativ die eines Menschen sogar noch übertreffen. Außerdem ergeben sich signifikante Zeiteinsparungen durch ein algorithmisches Vorgehen.

Um dieses Verfahren jedoch innerhalb der IT von Volkswagen oder aber auch in anderen IT-Bereichen mit einem möglichst geringen Aufwand einführen zu können, ist es im Weiteren noch notwendig, eine entsprechende Toolunterstützung bereitzustellen. Eine Integration dieses Modells zur Abbildung von Synergien und die automatische Generierung eines optimierten Iterationsvorschlags müssten bspw. in Microsoft Team Foundation Server, Atlassian Jira oder ähnlichen Applikationen mit integriert werden.

Diese Arbeit bildet die Grundlage für mehrere weiterführende Forschungen. Eine mögliche Forschungsrichtung, die sich aus dieser Arbeit ergibt, ist die Anwendung und Übertragung des Vorgehensmodells zur Quantifizierung des Nutzwertes von IT-Anforderungen. Hierbei wäre zu prüfen, ob das vorgeschlagene und im Kontext von Managementinformationssystemen vielversprechend erprobte Verfahren direkt auf andere Domänen übertragbar ist. Aus weiteren Untersuchungen könnten sich Erweiterungen und Verbesserungen zu dem beschriebenen Verfahren ableiten lassen.

Weitere Forschungsmöglichkeiten ergeben sich im Bereich der Abbildung und Betrachtung von Synergien zwischen IT-Anforderungen. In diesem Themenfeld könnte weiterführend, zu den in dieser Arbeit geleisteten Grundlagen, untersucht werden, ob der Einsatzes von Product Owner Stories und die Möglichkeit, diese auch wieder aus einem Projekt auszuphasen, reibungslos funktioniert. Hierbei wäre in unterschiedlichen Projekten zu evaluieren, welche Maßnahmen

tatsächlich nötig sind. Außerdem ergibt sich ein weiterer Untersuchungsansatz in der Überprüfung des hier vorgestellten Ansatzes mit einem zusätzlichen Treffen zur Identifikation von Synergien zwischen Anforderungen. In anderen Unternehmenskulturen ist dieses ggf. nicht notwendig, bzw. müssen Anpassungen vollzogen werden, um die Abbildung und Nutzung von Synergiepotenzialen innerhalb der IT zu gewährleisten.

In diesem Zusammenhang kann analysiert werden, ob das vorgestellte Modell für valide und optimale Iterationen in anderen Projektumfeldern hinreichend ist oder ob andere Kriterien in abweichenden Projektumfeldern betrachtet werden müssen, welche eine Anpassung des Modells notwendig machen. Sind Anpassungen an dem Modell notwendig, kann daraufhin geprüft werden, ob die hier vorgestellten algorithmischen Lösungsansätze weiterhin tragfähig sind oder ob andere Verfahren bei einem angepassten Modell besser geeignet sind als die in dieser Arbeit vorgestellten Verfahren zur Vorschlagsgenerierung.

Eine letzte weitere Forschungsrichtung kann in der Integration des hier vorgestellten Vorgehens und dem Ansatz von Software-Produktlinien gefunden werden. In diesem Bereich könnte untersucht werden, inwieweit die Definition von Features und Kernprodukten im Einklang mit Synergien auf Anforderungsebene stehen und ob sich dieses hier beschriebene Vorgehen für den Einsatz bei der Entwicklung von Software-Produktlinien eignet, um weitere Synergiepotenziale zu finden.

11 Literaturverzeichnis

Azar, Jim; Smith, Randy K.; Cordes, David (2007): Value-oriented requirements prioritization in a small development organization. In: *Software, IEEE* 24 (1), S. 32–37.

Bajaj, Punam; Arora, Vineet (2013): A Novel Approach to Select an Appropriate Requirements Prioritization Technique. In: *International Journal of Computer Applications* 77 (9), S. 25–30.

Baker, Brenda S. (1985): A new proof for the first-fit decreasing bin-packing algorithm. In: *Journal of Algorithms* 6 (1), S. 49–70. DOI: 10.1016/0196-6774(85)90018-5.

Balas, Egon; Zemel, Eitan (1980): An Algorithm for Large Zero-One Knapsack Problems. In: *Operations Research* 28 (5), S. 1130–1154. DOI: 10.1287/opre.28.5.1130.

Balzert, Helmut (1998): Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Heidelberg [u.a.]: Spektrum Akad. Verl (Lehrbücher der Informatik, / Helmut Balzert ; 2).

Barkow, Reinhard (2010): Enterprise-architecture-Management in der Praxis. Wandel, Komplexität und IT-Kosten im Unternehmen beherrschen. 1. Aufl. Hg. v. Jan H. Keuntje. Düsseldorf: Symposion.

Beier, Rene; Vöcking, Berthold (2003): Random knapsack in expected polynomial time. In: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing. ACM, S. 232–241.

Bente, Stefan; Bombosch, Uwe; Langade, Shailendra (2012): Collaborative enterprise architecture. Enriching EA with lean, agile, and enterprise 2.0 practices. [S.l.]: Morgan Kaufmann.

Berander, Patrik; Jönsson, Per (2006): Hierarchical cumulative voting (hcv)—prioritization of requirements in hierarchies. In: *International Journal of Software Engineering and Knowledge Engineering* 16 (06), S. 819–849.

Bitzer Philipp; Heidecke Hans-Christian; Leimeister Jan Marco (2014): SINIT@VW – Ein Ansatz zur systematischen Identifikation des Nutzens von IT-Anwendungen. Berlin, Boston: De Gruyter (Betriebliche Informationssysteme in der Automobilproduktion Soziotechnisches System - Nutzerpersönlichkeit - Nutzungserleben - Rollout und Betrieb - Fabriksteuerung - Informationen auf Shopfloor - IT-Nutzen), 03.11.2014.

Bleek, Wolf-Gideon; Wolf, Henning (2008): Agile Softwareentwicklung. Werte, Konzepte und Methoden. 1. Aufl. Heidelberg: Dpunkt-Verl. (it-agile).

Böckle, Günter; Pohl, Klaus; van der@Linden, Frank (2005): Software product line engineering. Foundations, principles, and techniques ; with 10 tables. Berlin [u.a.]: Springer.

Bolte, A.; Martin, H. (1992): Prozessbeherrschung durch Erfahrungswissen und deren technische Unterstützung. In: *Institut für Arbeitswissenschaft GHK (Hrsg.): Erfahrungsgeleitete Arbeit mit Werkzeugmaschinen. Kassel: Institut für Arbeitswissenschaft.*

Bosch, Jan (2002): Maturity and evolution in software product lines: Approaches, artefacts and organization. In: Software Product Lines: Springer, S. 257–271.

Bradley, Randy V.; Pratt, Renee M. E.; Byrd, Terry Anthony; Simmons, Lakisha (2011): The role of enterprise architecture in the quest for it value. In: *MIS Quarterly Executive* 10 (2), S. 19–27.

Brennan, Kevin (2009): A guide to the Business analysis body of knowledge (BABOK guide). Version 2.0. Toronto: International Institute of Business Analysis.

Bühne, S. (2002): Darstellung der Variabilität von Software-Produktfamilien durch Use Cases: Diplom.de. Online verfügbar unter https://books.google.de/books?id=DQxkAQAAQBAJ.

Cardinal, Mario (2013): Executable specifications with Scrum. A practical guide to Agile requirements discovery. Upper Saddle River, NJ: Addison-Wesley.

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0

Chu, P. C.; Beasley, J. E. (1998): A Genetic Algorithm for the Multidimensional Knapsack Problem. In: *Journal of Heuristics* 4 (1), S. 63–86. DOI: 10.1023/A:1009642405419.

Cleland-Huang, Jane; Denne, Mark (2005): Financially informed requirements prioritization. In: Gruia-Catalin Roman, William Griswold und Bashar Nuseibeh (Hg.): Proceedings of the 27th international conference on Software engineering. St. Louis, MO, USA, S. 710–711.

Cohn, Mike (2004): User stories applied. For agile software development. Boston: Addison-Wesley (Addison-Wesley signature series).

Cohn, Mike (2006): Agile estimating and planning. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference (Robert C. Martin series).

Coram, Michael; Bohner, Shawn (2005): The impact of agile methods on software project management. In: Engineering of Computer-Based Systems, 2005. ECBS'05. 12th IEEE International Conference and Workshops on the. IEEE, S. 363–370.

Cotta, C.; Troya, J. M. (1998): A Hybrid Genetic Algorithm for the 0–1 Multiple Knapsack Problem. In: George D. Smith, Nigel C. Steele und Rudolf F. Albrecht (Hg.): Artificial Neural Nets and Genetic Algorithms. Vienna: Springer Vienna, S. 250–254.

Daut, Patrick (2013): Alternative zur Rolle des Product Owner oder: Wie bleiben wir agil? Die Skalierung von Agilität in großen Organisationen. In: Business Technology, Bd. 4.2013, S. 6–12.

Denne, Mark; Huang, Jane (2004): Software by numbers. Low risk, high return development. Upper Saddle River, N.J.: Prentice Hall.

Derbier, Géry (2003): Agile Development in the old economy. In: Agile Development Conference, 2003. ADC 2003. Proceedings of the. IEEE, S. 125–131.

Djannaty, Farhad; Doostdar, Saber (2008): A hybrid genetic algorithm for the multidimensional knapsack problem. In: *International Journal of Contemporary Mathematical Sciences* 3 (9), S. 443–456.

Donegan, H. A.; Dodd, F. J.; McMaster, T. B. M. (1992): A New Approach to AHP Decision-Making. In: *The Statistician* 41 (3), S. 295. DOI: 10.2307/2348551.

Downey, Scott; Sutherland, Jeff (2013): Scrum Metrics for Hyperproductive Teams: How They Fly like Fighter Aircraft. In: 2013 46th Hawaii International Conference on System Sciences (HICSS). Wailea, HI, USA, S. 4870–4878.

Dr. Wilma Heim (1999): Die Einführung neuer Softwaresysteme. Erfolgsfaktoren und Hemmnisse. Wiesbaden: Deutscher Universitätsverlag (Entscheidungs- und Organisationstheorie).

Drexl, A. (1988): A simulated annealing approach to the multiconstraint zero-one knapsack problem. In: *Computing* 40 (1), S. 1–8. DOI: 10.1007/BF02242185.

Dudziński, Krzysztof; Walukiewicz, Stanisław (1987): Exact methods for the knapsack problem and its generalizations. In: *European journal of operational research* 28 (1), S. 3–21. DOI: 10.1016/0377-2217(87)90165-2.

Dyckhoff, Harald (1990): A typology of cutting and packing problems. In: *European journal of operational research* 44 (2), S. 145–159. DOI: 10.1016/0377-2217(90)90350-K.

Eilermann, Beate (2013): Nutzungserleben betrieblicher Informationssysteme in der Automobilindustrie. Berlin: Logos Berlin (AutoUni - Schriftenreihe, 42).

Falkenauer, E.; Delchambre, A.: A genetic algorithm for bin packing and line balancing. In: IEEE International Conference 12-14 May 1992, S. 1186–1192.

Garey, Michael R.; Johnson, David S. (1979): Computers and intractability. A guide to the theory of NP-completeness. San Francisco: W.H. Freeman (Series of books in the mathematical sciences).

Gerdes, Ingrid; Klawonn, Frank; Kruse, Rudolf (2004): Evolutionäre Algorithmen. Genetische Algorithmen - Strategien und Optimierungsverfahren - Beispielanwendungen ; [mit Online-Service zum Buch]. 1. Aufl. Wiesbaden: Vieweg (Computational intelligence).

Gigerenzer, Gerd; Todd, Peter M. (1999): Simple heuristics that make us smart. New York: Oxford University Press (Evolution and cognition).

Gloger, Boris (2009): Scrum. Produkte zuverlässig und schnell entwickeln. 2. Aufl. München: Hanser.

Gumm, Heinz-Peter; Sommer, Manfred (2006): Einführung in die Informatik. 7., vollst. überarb. Aufl. München [u.a.]: Oldenbourg.

Hamilton, Patrick (2008): Wege aus der Softwarekrise. Verbesserungen bei der Softwareentwicklung. Berlin: Springer.

Hanser, Eckhart (2010): Agile Prozesse: Von XP über Scrum bis MAP. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg (EXamen.press, 0).

Holland, John H. (1992a): Adaptation in natural and artificial systems. An introductory analysis with applications to biology, control, and artificial intelligence. 1st MIT Press ed. Cambridge, Mass.: MIT Press (Complex adaptive systems).

Holland, John H. (1992b): Genetic algorithms. In: *Scientific american* 267 (1), S. 66–72.

Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. (2002): Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie. 2., überarb. Aufl. München [u.a.]: Pearson Studium (Informatik).

Hossain, Emam; Babar, Muhammad Ali; Paik, Hye-young (2009): Using scrum in global software development: a systematic literature review. In: Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on. IEEE, S. 175–184.

Irani, Zahir (2002): Information systems evaluation: navigating through the problem domain. In: *Information & Management* 40 (1), S. 11–24.

ISO/IEC (2001): ISO/IEC 9126. Software engineering – Product quality: ISO/IEC.

Johnson, D. S. (1973): Near-optimal Bin Packing Algorithms: Massachusetts Institute of Technology (Massachusetts Institute of Technology, project MAC). Online verfügbar unter http://books.google.de/books?id=8pGNGAAACAAJ.

Johnson, D. S.; Demers, A.; Ullman, J. D.; Garey, M. R.; Graham, R. L. (1974): Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms. In: *SIAM J. Comput.* 3 (4), S. 299–325. DOI: 10.1137/0203025.

Johnson, David S. (1974): Fast algorithms for bin packing. In: *Journal of Computer and System Sciences* 8 (3), S. 272–314. DOI: 10.1016/S0022-0000(74)80026-7.

Karlsson, Joachim; Wohlin, Claes; Regnell, Björn (1998): An evaluation of methods for prioritizing software requirements. In: *Information and Software Technology* 39 (14), S. 939–947.

Karp, Richard M. (1972): Reducibility among Combinatorial Problems. In: Raymond E. Miller, James W. Thatcher und Jean D. Bohlinger (Hg.): Complexity of Computer Computations. Boston, MA: Springer US, S. 85–103.

Kastens, Uwe; Kleine Büning, Hans (2005): Modellierung. Grundlagen und formale Methoden. München [u.a.]: Hanser.

Kemper, H.G., Baars, H. & Mehanna W. (2010): Business Intelligence - Grundlagen und praktische Anwendung. Wiesbaden: Vieweg.

Kent Beck et al. (2014): Manifesto for Agile Software Development. Online verfügbar unter http://agilemanifesto.org/, zuletzt aktualisiert am 21.11.2014, zuletzt geprüft am 15.06.2016.

Kettunen, Petri (2013): Bringing Total Quality in to Software Teams: A Frame for Higher Performance. In: Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, Clemens Szyperski, Brian Fitzgerald et al. (Hg.): Lean Enterprise Software and Systems, Bd. 167. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Business Information Processing), S. 48–64.

Khuri, Sami; Bäck, Thomas; Heitkötter, Jörg: The zero/one multiple knapsack problem and genetic algorithms. In: Hal Berghel, Terry Hlengl und Joseph Urban (Hg.): the 1994 ACM symposium. Phoenix, Arizona, United States, S. 188–193.

Knuth, Donald E. (1998): The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc.

Kohli, Rajiv; Devaraj, Sarv (2003): Measuring information technology payoff: A meta-analysis of structural variables in firm-level empirical research. In: *Information systems research* 14 (2), S. 127–145.

Komus, Ayelt (2014): Internationale Studie: Status Quo Agile 2015. Zweite Studie des BPM-Labors der Hochschule Koblenz, Prof. Dr. Ayelt Komus, über die Verwendung agiler Methoden. Hochschule Koblenz. Online verfügbar unter https://www.hs-koblenz.de/fileadmin/media/fb_wirtschaftswissenschaften/Forschung_Projekte/Forschungsprojekte/Status_Quo_Agile/Studie_2014/2014.07.23_Bericht_Highlights_final.v.1.01.pdf, zuletzt aktualisiert am 08.01.2015, zuletzt geprüft am 01.05.2015.

Kunde, Manfred; Steppat, Horst (1985): First fit decreasing scheduling on uniform multiprocessors. In: *Discrete Applied Mathematics* 10 (2), S. 165–177. DOI: 10.1016/0166-218X(85)90010-1.

Kwong, C. K.; Bai, H. (2002): A fuzzy AHP approach to the determination of importance weights of customer requirements in quality function deployment. In: *Journal of intelligent manufacturing* 13 (5), S. 367–377.

Lacey, Mitch (2012): The Scrum field guide. Practical advice for your first year. 1st ed. [S.l.]: Addison-Wesley Professional.

Leung, Stephen C.H.; Zhang, Defu; Zhou, Changle; Wu, Tao (2012): A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem. In: *Computers & Operations Research* 39 (1), S. 64–73. DOI: 10.1016/j.cor.2010.10.022.

Levandowsky, Michael; Winter, David (1971): Distance between sets. In: *Nature* 234 (5323), S. 34–35.

Martello, S.; Toth, P. (1981): Heuristic algorithms for the multiple knapsack problem. In: *Computing* 27 (2), S. 93–112. DOI: 10.1007/BF02243544.

Martello, Silvano; Toth, Paolo (1990): Lower bounds and reduction procedures for the bin packing problem. In: *Discrete Applied Mathematics* 28 (1), S. 59–70. DOI: 10.1016/0166-218X(90)90094-S.

Martin, H. (1994): Grundlagen der menschengerechten Arbeitsgestaltung - Handbuch für die betriebliche Praxis. Köln: Bund-Verlag.

Meinel, Christoph; Mundhenk, Martin (2006): Mathematische Grundlagen der Informatik. Mathematisches Denken und Beweisen ; eine Einführung. 3., überarb. und erw. Aufl. Wiesbaden: Teubner (Lehrbuch Informatik).

Misra, Subhas Chandra; Kumar, Vinod; Kumar, Uma (2009): Identifying some important success factors in adopting agile software development practices. In: *Journal of Systems and Software* 82 (11), S. 1869–1890. DOI: 10.1016/j.jss.2009.05.052.

Mühleck, K. H. & Heidecke, H.-C. (2012): Trends in der Entwicklung von IT-Systemen in der Automobilindustrie. In: Werner Neubauer und Bernd Rudow (Hg.): Trends in der Automobilindustrie. Entwicklungstendenzen, Betriebsratsarbeit, Steuer- und Fördertechnik, Gießereitechnik, Informationstechnologie, Informations- und Assistenzsysteme. München: Oldenbourg, S. 85–104.

Nerur, Sridhar; Mahapatra, RadhaKanta; Mangalaraj, George (2005): Challenges of migrating to agile methodologies. In: *Commun. ACM* 48 (5), S. 72–78. DOI: 10.1145/1060710.1060712.

Neubauer, Werner; Rudow, Bernd (Hg.) (2012): Trends in der Automobilindustrie. Entwicklungstendenzen, Betriebsratsarbeit, Steuer- und Fördertechnik, Gießereitechnik, Informationstechnologie, Informations- und Assistenzsysteme. München: Oldenbourg.

Nonaka, Ikujiro; Takeuchi, Hirotaka (2004): Hitotsubashi on knowledge management: Wiley.

Olsen, A. L. (1994): Penalty functions and the knapsack problem.

Paasivaara, Maria; Durasiewicz, Sandra; Lassenius, Casper (2009): Using Scrum in Distributed Agile Development: A Multiple Case Study: IEEE.

Paasivaara, Maria; Lassenius, Casper; Heikkilä, Ville T.: Inter-team coordination in large-scale globally distributed scrum. In: Per Runeson, Martin Höst, Emilia Mendes, Anneliese Andrews und Rachel Harrison (Hg.): the ACM-IEEE international symposium. Lund, Sweden, S. 235–238.

Parnas, David Lorge (1976): On the design and development of program families. In: *Software Engineering, IEEE Transactions on* (1), S. 1–9.

Paul Clements; Linda Northrop (2001): Software Product Lines: Practices and Patterns: Addison-Wesley Professional.

Petersen, Kai (2010): Implementing Lean and Agile software development in industry.

Pettit, R. (2006): AgileConnection | The Agile Experience: The Business Value of Agility. Hg. v. Agile Journal. Online verfügbar unter http://www.agileconnection.com/article/agile-experience-business-value-agility, zuletzt geprüft am 01.05.2015.

Pichler, Roman (2010): Agile product management with scrum: Creating Products that costumers love: Addison-Wesley Professional.

Pirkul, Hasan (1987): A heuristic solution procedure for the multiconstraint zero-one knapsack problem. In: *Naval Research Logistics* 34 (2), S. 161–172.

Pisinger, David (1995): Algorithms for knapsack problems.

Plath, Hans-Eberhard (2002): Erfahrungswissen und Handlungskompetenz ‑ Konsequenzen für die berufliche Weiterbildung. In: *IAB-Kompendium Arbeitsmarkt-und Berufsforschung. Beiträge zur Arbeitsmarkt-und Berufsforschung, BeitrAB*, S. 517–529.

Pohl, Klaus; Metzger, Andreas (2008): Variabilitätsmanagement in Software-Produktlinien. In: Software Engineering, S. 28–41.

Polanyi, Michael (1985): Implizites Wissen. 1. Aufl. Frankfurt am Main: Suhrkamp (Suhrkamp Taschenbuch Wissenschaft, 543).

Racheva, Zornitza; Daneva, Maya; Buglione, Luigi (2008): Supporting the dynamic reprioritization of requirements in agile development of software products. In: Software Product Management, 2008. IWSPM'08. Second International Workshop on. IEEE, S. 49–58.

Racheva, Zornitza; Daneva, Maya; Sikkel, Klaas (2009): Value Creation by Agile Projects: Methodology or Mystery? In: Will Aalst, John Mylopoulos, Norman M. Sadeh, Michael J. Shaw, Clemens Szyperski, Frank Bomarius et al. (Hg.): Product-Focused Software Process Improvement, Bd. 32. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Business Information Processing), S. 141–155.

Ramanathan, R.; Ganesh, L. S. (1994): Group preference aggregation methods employed in AHP: An evaluation and an intrinsic process for deriving members' weightages. In: *European journal of operational research* 79 (2), S. 249–265.

Rao, R. L.; Iyengar, S. S. (1994): Bin-packing by simulated annealing. In: *Computers & Mathematics with Applications* 27 (5), S. 71–82. DOI: 10.1016/0898-1221(94)90077-9.

Reeves, Colin (1996): Hybrid genetic algorithms for bin-packing and related problems. In: *Ann Oper Res* 63 (3), S. 371–396. DOI: 10.1007/BF02125404.

Rein, Alexander-Derek; Münch, Jürgen (2013): Feature Prioritization Based on Mock-Purchase: A Mobile Case Study. In: Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, Clemens Szyperski, Brian Fitzgerald et al. (Hg.): Lean Enterprise Software and Systems, Bd. 167. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Business Information Processing), S. 165–179.

Rising, Linda; Janoff, Norman S. (2000): The Scrum software development process for small teams. In: *IEEE software* (4), S. 26–32.

Rubin, Kenneth S. (2012): Essential Scrum. A practical guide to the most popular agile process. Upper Saddle River, NJ: Addison-Wesley (The Addison-Wesley signature series).

Rudow, Bernd; Heidecke, Hans-Christian (2014): Betriebliche Informationssysteme in der Automobilproduktion: De Gruyter, Oldenbourg.

Ruehl, Stefan T.; Andelfinger, Urs; Rausch, Andreas; Verclas, Stephan A. W. (2012): Toward realization of deployment variability for software-as-a-service applications. In: Cloud computing (cloud), 2012 ieee 5th international conference on. IEEE, S. 622–629.

Ruigies, A. (1998): Simulated Annealing und verwandte Verfahren für das Traveling Salesman Problem: Zur Studie gehört Software, die nur in digitaler Form (CD oder Download) erhältlich ist: Diplom.de.

Rupp, Chris; SOPHISTen, die (2014): Requirements-Engineering und -Management. Aus der Praxis von klassisch bis agil. 6., aktualisierte und erweiterte Auflage. München: Hanser, Carl.

Saaty, R. W. (1987): The analytic hierarchy process—what it is and how it is used. In: *Mathematical Modelling* 9 (3-5), S. 161–176. DOI: 10.1016/0270-0255(87)90473-8.

Saaty, Thomas L. (1990): The analytic hierarchy process. Planning, priority setting, resource allocation. 2nd ed. Pittsburgh, PA: RWS Publications.

Saaty, Thomas L. (1994): Fundamentals of decision theory. Pittsburgh, Pa: RWS Publications.

Saaty, Thomas L. (1999): Decision making for leaders: the analytic hierarchy process for decisions in a complex world: RWS Publications (2).

Saaty, Thomas L. (2003): Decision-making with the AHP: Why is the principal eigenvector necessary. In: *European journal of operational research* 145 (1), S. 85–91.

Saaty, Thomas L. (2008): Decision making with the analytic hierarchy process. In: *International journal of services sciences* 1 (1), S. 83–98.

Schwaber, K.; Sutherland, J. (2013a): The Scrum Guide. The Definitive Guide to Scrum: the Rules of the game. Hg. v. www.scrum.org. Online verfügbar unter https://www.scrum.org/scrum-guide, zuletzt geprüft am 18.08.2014.

Schwaber, Ken (2004): Agile project management with Scrum. Redmond, Wash.: Microsoft Press.

Schwaber, Ken; Beedle, Mike (2002): Agile software development with Scrum. Pearson international ed. Upper Saddle River, NJ: Pearson Education International (Series in agile software development).

Schwaber, Ken; Sutherland, Jeff (2013b): The Scrum Guide. The Definitve Guide to Scrum: The Rules of the Game. Online verfügbar unter http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf, zuletzt aktualisiert am July 2013, zuletzt geprüft am 27.11.2014.

Schwindt, Christoph (2005): Resource allocation in project management. Berlin, New York, NY: Springer (GOR-publications).

Sobiech, Fabian; Eilermann, Beate; Rausch, Andreas (2014): On iteration optimization for non-cross-functional teams in Scrum, zuletzt geprüft am 2014.

Sobiech, Fabian; Eilermann, Beate; Rausch, Andreas (2015): A heuristic approach to solve the elementary sprint optimization problem for non-cross-functional teams in Scrum. In: *ACM SIGAPP Applied Computing Review* 14 (4), S. 19–26.

Sobiech, Fabian; Eilermann, Beate; Rausch, Andreas (2016): Using Synergies between User Stories in Scrum: IACSIT PRESS (Lecture Notes on Software Engineering, Vol 4., No. 2).

Standish Group International (2014): Chaos Report. Online verfügbar unter http://www.standishgroup.com/, zuletzt geprüft am 11.10.2015.

Stawowy, Adam (2008): Evolutionary based heuristic for bin packing problem. In: *Computers & Industrial Engineering* 55 (2), S. 465–474. DOI: 10.1016/j.cie.2008.01.007.

Stober, Thomas; Hansmann, Uwe (2010): Overview of Agile Software Development. In: Thomas Stober und Uwe Hansmann (Hg.): Agile Software Development. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 35–59.

Sureshchandra, Kalpana; Shrinivasavadhani, Jagadish: Adopting Agile in Distributed Development. In: 2008 IEEE International Conference on Global Software Engineering (ICGSE). Bangalore, S. 217–221.

Sutherland, Jeff (Hg.) (1997): Business object design and implementation. Workshop proceedings. London [u.a.]: Springer.

Sutherland, Jeff; Viktorov, Anton; Blount, Jack; Puntikov, Nikolai (2007): Distributed Scrum: Agile Project Management with Outsourced Development Teams. In: 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07). Waikoloa, HI, USA, S. 274a.

Szőke, Ákos (2009): Decision Support for Iteration Scheduling in Agile Environments. In: Will Aalst, John Mylopoulos, Norman M. Sadeh, Michael J. Shaw, Clemens Szyperski, Frank Bomarius et al. (Hg.): Product-Focused Software Process Improvement, Bd. 32. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Business Information Processing), S. 156–170.

Ting, Chuan-Kang (2005): On the Mean Convergence Time of Multi-parent Genetic Algorithms Without Selection. In: Mathieu S. Capcarrere (Hg.): Advances in artificial life. 8th European conference ; proceedings, Bd. 3630. Berlin, Heidelberg, New York: Springer (Lecture notes in computer science, Vol. 3630 : Lecture notes in artificial intelligence), S. 403–412.

van Gurp, Jilles; Bosch, Jan; Svahnberg, Mikael (2001): On the notion of variability in software product lines. In: Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on. IEEE, S. 45–54.

Vlaanderen, Kevin; Jansen, Slinger; Brinkkemper, Sjaak; Jaspers, Erik (2011): The agile requirements refinery: Applying SCRUM principles to software product management. In: *Information and Software Technology* 53 (1), S. 58–70. DOI: 10.1016/j.infsof.2010.08.004.

von der Maßen, Thomas; Lichter, Horst (2003): Modellierung von variabilität mit uml use cases. In: *Softwaretechnik-Trends* 23 (1).

Wegener, Ingo (2003): Komplexitätstheorie. Grenzen der Effizienz von Algorithmen. Berlin [u.a.]: Springer (Springer-Lehrbuch).

Wiegers, Karl (1999): First things first: prioritizing requirements. In: *Software Development* 7 (9), S. 48–53.

Wirdemann, Ralf (2011): Scrum mit User Stories. 2., erweiterte Auflage. München: Hanser, Carl.

Wnuk, Krzysztof; Regnell, Björn; Berenbach, Brian (2011): Scaling Up Requirements Engineering - Exploring the Challenges of Increasing Size and Complexity in Market-Driven Software Development. In: Requirements Engineering: Foundation for Software Quality: Springer, S. 54–59.

Zanakis, Stelios H. (1977): Heuristic 0-1 Linear Programming: An Experimental Comparison of Three Methods. In: *Management Science* 24 (1), S. p 91-104. Online verfügbar unter http://www.jstor.org/stable/2630731.

12 Anhang

12.1 Auflistung aller abgefragten Kombinationen innerhalb von AHP

Tabelle 14: Auflistung aller abgefragten Vergleiche für die Bestimmung des Nutzwertvektors

Nutzwertdimension	Nutzwertdimension
financial value	work organizational value
financial value	negative value
financial value	software qualitative value
financial value	tertiary value
financial value	strategic value (customer)
financial value	strategic value (IT)
work organizational value	negative value
work organizational value	software qualitative value
work organizational value	tertiary value
work organizational value	strategic value (customer)
work organizational value	strategic value (IT)
negative value	software qualitative value
negative value	tertiary value
negative value	strategic value (customer)
negative value	strategic value (IT)
software qualitative value	tertiary value
software qualitative value	strategic value (customer)
software qualitative value	strategic value (IT)
tertiary value	strategic value (customer)
tertiary value	strategic value (IT)
strategic value (customer)	strategic value (IT)

F. Sobiech, *Abbildung von Synergiepotenzialen zwischen IT-Anforderungen in Scrum*, AutoUni – Schriftenreihe 95,
DOI 10.1007/978-3-658-16328-0

12.2 Rucksackproblem

In allen Varianten des Rucksackproblems geht es darum, eine Menge an Gegenständen, von denen jeder einen Nutzwert p_j und ein Gewicht w_j besitzt, in einen oder mehrere Rucksäcke zu packen. Dabei besitzt jeder Rucksack eine bestimmte Gewichtsgrenze c, um auszudrücken, dass dieser Rucksack nicht überladen werden kann. Es ist im Regelfall davon auszugehen, dass alle Koeffizienten p_j, w_j und c positiv und ganzzahlig sind.

Beim 0-1 Rucksackproblem besteht das Problem darin, aus einer gegebenen Menge an Gegenständen eine Teilmenge dieser Gegenstände auszuwählen, sodass die Summe der korrespondierenden Nutzwerte maximiert wird, ohne dass dabei die Summe der Gewichte die Kapazitätsgrenze c des Rucksacks übersteigt. Dieses kann wie folgt als Maximierungsproblem formuliert werden:

$$maximiere \sum_{j=1}^{n} p_j * x_j$$

$$mit \sum_{j=1}^{n} w_j * x_j \leq c,$$

$$x_j \in \{0,1\}, \qquad j = 1, \dots, n$$

Hierbei wird x_j als binäre Variable aufgefasst, die 1 liefert, genau dann, wenn der Gegenstand j Teil der Teilmenge von Gegenständen ist, die sich innerhalb des Rucksacks befinden sollen und sonst 0 zurück liefert. (vgl. Pisinger 1995)

Die Entscheidungsvariante dieses Problems kann in Anlehnung an die obere Darstellung so formuliert werden, dass nicht die Nutzwertmaximierung von Bedeutung ist, sondern eine Befüllung des Rucksacks gesucht wird, die mindestens über einen bestimmten Nutzwert a verfügt. Somit würde sich folgende alternative Formulierung ergeben:

$$a \geq \sum_{j=1}^{n} p_j * x_j$$

$$mit \sum_{j=1}^{n} w_j * x_j \leq c,$$

$$x_j \in \{0,1\}, \qquad j = 1, \dots, n$$

Die Bedingung bzgl. der Gewichtsschranke bleibt hierbei unberührt.

Gibt es eine bestimmte Anzahl jedes Gegenstandes, so wird auch vom „Bounded Knapsack Problem“ gesprochen. Es gibt also eine bestimmte Anzahl m_j von Gegenstand j. Das Problem lässt sich dann wie folgt formulieren:

$$maximiere \sum_{j=1}^{n} p_j * x_j$$

$$mit \sum_{j=1}^{n} w_j * x_j \leq c,$$

$$x_j \in \{0,1, \dots, m_j\}, \qquad j = 1, \dots, n$$

Bei dieser Formulierung ist x_j keine binäre Variable mehr, die lediglich die Zugehörigkeit zum Rucksack angibt. Sie gibt jetzt auch die Häufigkeit an, mit der dieser Gegenstand eingepackt wird, um den Nutzwert des gesamten Rucksacks zu maximieren.

12.3 Handout zur Erprobung

Product Owner Story	Team A	Team B
POS 1	x	
POS 2		x
POS 3		x
POS 4	x	
POS 5	x	x
POS 6		x
POS 7		x
POS 8	x	
POS 9	x	
POS 10	x	x
POS 11		x
POS 12	x	
POS 13	x	
POS 14	x	
POS 15		x
POS 16	x	
POS 17		x
POS 18	x	
POS 19	x	x
POS 20	x	
POS 21		x
POS 22	x	
POS 23	x	x
POS 24		x

POS 25	x	x
POS 26		x
POS 27	x	
POS 28	x	
POS 29	x	
POS 30		x
POS 31		x
POS 32	x	
POS 33		x
POS 34		x
POS 35		x
POS 36	x	
POS 37		x
POS 38	x	
POS 39		x
POS 40	x	