Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Analyse von Konflikten beim Requirements Engineering in hybriden Entwicklungsansätzen

Analysis of conflicts in requirements engineering in hybrid development approaches

Masterarbeit
im Studiengang Informatik

von

Niklas Blume

Prüfer: Prof. Dr. rer. nat. Kurt Schneider
Zweitprüfer: Dr. Jil Klünder
Betreuer: Nils Prenner

Hannover, 03.11.2021

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 03.11.2021

Niklas Blume

Zusammenfassung

Das Requirements Engineering, also das Ermitteln, Dokumentieren, Kommunizieren und Verwalten von Anforderungen, bildet das Grundgerüst eines jeden Projekts. Schon bevor eine eigentliche Software-Anwendung in die Entwicklung geht, muss ausführlich bestimmt werden, für was und wie diese Software am Ende des Projekts nutzen soll. Grundlegend gibt es zwei verschiedene Wege, wie man Requirements Engineering durchführen kann - traditionell und agil. Zwischen diesen beiden Varianten existiert noch das hybride Requirements Engineering, in der traditionelle und agile Methoden miteinander verbunden werden, womit versucht wird, die Vorteile beider Methoden zu nutzen. Da sowohl das traditionelle als auch das agile RE sehr verschieden voneinander sind, entstehen Konflikte, wenn man diese beiden miteinander kombiniert. Diese Konflikte und das Lösen dieser ist Bestandteil der vorliegenden Masterarbeit. Als Grundlage für die Konflikte wurde eine systematische Literaturrecherche durchgeführt, um die prägnantesten Konflikte näher zu analysieren und mit einer Lösung zu verbinden.

Abstract

Analysis of conflicts in requirements engineering in hybrid development approaches

Requirements engineering, i.e. determining, documenting, communicating and managing requirements, forms the basic framework of every project. Even before an actual software application goes into development, it must be planned in detail what and how this software is to be used at the end of the project. There are basically two different ways in which requirements engineering can be carried out - traditional and agile. Between these two variants there is also hybrid requirements engineering, in which traditional and agile methods are combined with one another, which attempts to use the advantages of both methods. Since both traditional and agile RE are very different from each other, conflicts arise when you combine them. These conflicts and the resolution of them is part of this master's thesis. As a basis for the conflicts, a systematic literature research was carried out in order to analyze the most concise conflicts and combine them with a solution.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Das Bestimmen von Anforderungen ist einer der wichtigsten Bereiche im Software Engineering, um die Kommunikation zwischen Kunden und Entwicklern zu optimieren und trägt dazu bei, die Entwicklung des eigentlichen Projekts voranzutreiben. Das richtige Bestimmen von Anforderungen ist die Aufgabe des Requirements Engineering (kurz: RE). Das Requirements Engineering ist sehr wichtig für die richtige Umsetzung eines Softwareprojekts, da es die Anforderungen des Kunden sammelt und diese gut dokumentiert an die Entwickler gibt. Werden Anforderungen eines Projekts nicht korrekt an die Entwickler mitgeteilt, hat dies gravierende Folgen für den weiteren Projektablauf, wie zum Beispiel fehlende Stabilität des Projekts oder das Übersehen von notwendigen Anforderungen [6], wodurch es zur Verzögerung oder Abbruch des kompletten Projektes kommen kann.

Beim Requirements Engineering wird zwischen zwei verschiedenen Methoden unterschieden: dem traditionellen RE und dem agilen RE. Beide Methoden agieren auf sehr unterschiedliche Art und Weise und haben ihre Vor- und Nachteile.

Das traditionelle Requirements Engineering auf der einen Seite überzeugt vor allem mit einer ausführlichen Dokumentation, da die Kundenwünsche im Vorfeld des eigentlichen Projekts gründlich ermittelt werden. Dadurch können die Anforderungen umfangreich beschrieben an die Entwickler weitergeleitet werden. Das Problem dabei ist jedoch, dass dies auf der einen Seite sehr viel Zeit in Anspruch nimmt, und auf der anderen Seite können die Anforderung während des Projektablaufs wenig oder gar nicht mehr angepasst werden, da diese nur im Vorfeld erhoben werden.

Nach Ramesh et al. [40] muss sich das traditionelle RE weiterentwickelt, da sich das Requirements Engineering zum einen an veränderbare Anforderungen anpassen muss und zum anderen muss das RE auf die schnell verändernde Technologie reagieren. Durch dieses Bedüfnis haben sich agile

Methoden entwickelt, wie zum Beispiel das iterative RE, Prototyping und das konstante Planen von Anforderungen [40] , auch nachdem die Entwicklung des Projekts begonnen hat, um Anpassung im Nachhinein noch umsetzen zu können. Dadurch entstehen jedoch Nachteile, wie eine mangelhafte Dokumentation oder eine lückenhafte Kosten- und Zeitplanung.

Da besonders große Firmen nicht direkt ihre Methode des Requirements Engineering ohne Probleme ändern können, da dies über eine lange Zeit in dem jeweiligen Betrieb so betrieben wurde, muss ein Mittelweg gefunden werden, um vom traditionellen RE in die Richtung des agilen RE zu gehen. Dies ist das hybride Requirements Engineering (kurz: HRE). Das HRE beschreibt eine Mischung aus agilem und traditionellen RE und versucht, die Einschränkungen des traditionellen RE mit den Vorteilen des agilen RE zu verbinden [27].

1.1 Problemstellung

Das hybride Requirements Engineering ist eine Kombination aus traditionellem RE und agilen RE. Mit dem HRE wird Firmen ermöglicht, die Vorteile der beiden RE Methoden zu kombinieren und je nach Bedürfnis eine Kombination aus agilen und traditionellen Methoden zu nutzen. Jedoch unterscheiden sich beide Methoden sehr stark voneinander, wodurch eine Kombination der RE Varianten nicht ohne Probleme funktioniert. Beispielsweise wird beim traditionellen RE sehr viel Fokus auf eine ausführliche Dokumentation gelegt, beim agilen Requirements Engineering liegt die Priorität nicht auf der Dokumentation, da durch stetige Änderungen diese auch stetig angepasst werden müsste, wodurch eine lückenhafte Dokumentation entsteht. Durch Unterschiede wie dieser und weiteren entstehen Konflikte bei der Anwendung des HRE. Daher müssen Lösungsansätze gefunden werden, um Konflikte ausschließen zu können bzw. um den Umgang mit ihnen zu erleichtern.

1.2 Zielsetzung

Diese Masterarbeit hat das Ziel, Konflikte im Zusammenhang mit dem hybriden Requirements Engineering mithilfe einer systematischen Literaturrecherche (SLR) zu ermitteln. Dieses Ziel wird in zwei kleinere Themen unterteilt: Zum einen geht es darum, Konflikte zu ermitteln, in die ein Unternehmen kommen kann, wenn es hybride bzw. agile Methoden adaptieren will, oder welche Konflikte es gibt, wenn es bereits einige dieser Methoden adaptiert hat. In diesem Zusammenhang werden auch die

Ursachen der Konflikte analysiert, um gegebenenfalls Gemeinsamkeiten und Wechselwirkungen zwischen Konflikten zu ermitteln.

Zum anderen geht es um Lösungen für diese Konflikte. Mithilfe der SLR werden konkrete Lösungen oder Lösungsansätze extrahiert und diese mit den Konflikten verbunden. Werden durch die SLR keine Lösungsansätze für bestimmte Probleme gefunden, werden eigens entwickelte Lösungsansätze vorgeschlagen.

1.3 Struktur der Arbeit

Im nächsten Kapitel werden die Grundlagen dieser Arbeit beschrieben. Dies beinhaltet auf der einen Seite die Grundlagen für traditionelles und agiles RE, sowie Gemeinsamkeiten und Unterschiede der beiden Methoden. Andererseits werden auch die Grundlagen der systematischen Literaturrecherche nach Kitchenham et al. [26] erklärt.

Im dritten Kapitel wird auf das Forschungsdesign dieser Arbeit eingegangen und damit der konkreten Anwendung der in den Grundlagen beschriebenen SLR.

Kapitel 4 und 5 beschreiben die Resultate der SLR. Kapitel 4 gibt eine Übersicht über alle gefundenen Konflikte. In Kapitel 5 wird eine Lösung für jeden dieser Konflikte gefunden, entweder als Ergebnis der SLR oder eigene Ansätze.

Im sechsten Kapitel werden verwandte Arbeiten vorgestellt, die ähnliche Bereiche des RE betreffen, aber nicht explizit in dieser Arbeit behandelt werden.

Abschließend wird im letzten Kapitel eine Zusammenfassung gegeben, sowie ein Ausblick für die zukünftige Forschung.

Kapitel 2

Grundlagen

Dieses Kapitel beschäftigt sich mit den allgemeinen Grundlagen des Requirements Engineering (kurz: RE) und beschreibt sowohl die traditionelle Umsetzung des RE, als auch die agile Umsetzung und geht auf Gemeinsamkeiten und Unterschiede zwischen diesen beiden ein. Anschließend wird auf die dieser Arbeit zugrunde liegende Systematische Literaturrecherche eingegangen.

2.1 Traditionelles Requirements Engineering

Requirements Engineering ist der Prozess zur Bestimmung von Anforderungen, um herauszufinden, was genau ein System tun soll und nicht, wie es dieses System umsetzen soll [16]. Dieser Prozess wird in zwei Hauptteile unterteilt: der Requirements Analyse und dem Requirements Management, die erneut in Unterkategorien aufgeteilt werden. Die Requirements Analyse unterteilt sich in Erhebung, Interpretation, Abstimmung, Dokumentation und Validierung Verifikation [44]. In diesem ersten Abschnitt des RE werden die Anforderungen der Stakeholder zusammengetragen, abgestimmt und im späteren Verlauf in verschiedenen Dokumenten festgehalten, wie zum Beispiel in einer Spezifikation. Das Requirements Management unterteilt sich in das Änderungsmanagement und das Tracing (Verfolgbarkeit). Diese beschäftigen sich um die Umsetzung der in der Analyse festgelegten Anforderungen und halten diese fest, um im späteren Verlauf der Entwicklung zurückverfolgen zu können, welche Anforderungen in welcher Weise umgesetzt wurden [44]. Die nachfolgenden Abschnitte werden diese Unterkategorien des RE ausführlicher erläutern.

2.1.1 Erhebung

Der erste Teil des RE ist die Requirements Erhebung. Die Requirements Erhebung versucht, Anforderungen zu entdecken und Systemgrenzen zu identifizieren, indem die betroffenen Stakeholder der Anwendung auf verschiedene Arten befragt werden [36]. Durch den unterschiedlichen Wissenstand und Anwendungsbereich der Stakeholder, kann die Qualität der erhaltenen Anforderungen stark variieren [16]. Um die Anforderungen so präzise wie möglich zu ermitteln, gibt es verschiedene Methoden. Interviews sind eine der wichtigsten Methoden, um Anforderungen zu erheben. Dabei unterscheidet man zwischen geschlossenen Interviews, bei denen die Fragen im Vorhinein festgelegt werden, und offenen Interviews, bei denen keine Fragen vorher geplant werden, sondern man eine offene Diskussion mit den Stakeholdern führt [36]. Weitere Methoden sind beispielsweise Use Cases, bei der eine bestimmte Interaktion mit dem System formal nachgebaut wird, oder Prototypen, damit die Stakeholder bereits vor der eigentlichen Entwicklung einen ersten Eindruck von der geplanten Anwendung bekommen, um auch tiefgreifendere Anforderungen zu planen. Diese Prototypen können zum Beispiel aus Papier angefertigt werden [36].

2.1.2 Analyse (Interpretation & Abstimmung)

Der zweite Schritt bei dem Requirementsengineering ist die Analyse der zuvor erhobenen Anforderungen. Hauptaspekt dieses Abschnittes ist das Identifizieren und Lösen jeglicher Konflikte und Inkonsistenten .[16]. Im Anschluss werden die entsprechenden Anforderungen durch die Stakeholder abgestimmt und in der Reihenfolge priorisiert, in der diese durch die Entwickler umgesetzt werden sollen.

2.1.3 Dokumentation

Die Dokumentation der abgestimmten Anforderungen ist nun der nächste Punkt. Der Hauptgrund der Dokumentation besteht darin, die Kommunikation zwischen den Beteiligten, insbesondere zwischen Auftragsteller und Entwickler, zu gewährleisten [36]. Die festgehaltenen Anforderungen sind die Grundvoraussetzung für die weitere Vorgehensweise mit der jeweils zu entwickelnden Software-Anwendung. Nach [36] soll das Anforderungsdokument bzw. Spezifikation folgende Eigenschaften erfüllen: eindeutig, vollständig, korrekt, verständlich, konsistent, prägnant und machbar.

2.1.4 Validierung/Verifikation

Durch die Validierung der Anforderungen wird sichergestellt, dass die Entwickler das korrekte Endprodukt entwickeln. Dabei wird vor allem darauf geachtet, dass die zuvor geschriebenen Dokumentationen, wie beispielsweise die fachliche Spezifikation, den Bedarf der Stakeholder widerspiegelt [16]. Dies wird ebenfalls sichergestellt, in dem die der Spezifikation zugrunde liegenden Anforderungen überprüft werden, wie beispielsweise User Stories oder Ablaufdiagramme. Dem gegenüber steht die Verifikation. Dabei wird sichergestellt, dass das Entwicklerteam das Produkt so entwickelt, wie die Stakeholder es festgelegt haben. Formale Anforderungen werden durch Notation oder der Überprüfung der zugrunde liegenden Modelle verifiziert [16]. Durch einheitliche Notationen wird beispielsweise eine automatisierte Verifikation erleichtert, da diese entsprechend konsistent in den Anforderungsdokumentationen ist.

2.1.5 Management

Beim Anforderungsmanagement wird Wert darauf gelegt, dass alle Anforderungen zentral gespeichert werden und alle Informationen zu diesen Anforderungen nachvollziehbar und verfolgbar sind. Dadurch wird eine Änderungs- und Versionskontrolle ermöglicht [36]. Zusätzlich werden die Beziehungen zwischen Anforderungen, Design und Implementation verwaltet, um Änderungen an das System nachvollziehen zu können.

2.2 Agiles Requirements Engineering

Das agile Requirements Engineering funktioniert vom generellen Aufbau des RE ähnlich zum traditionellen und es beinhaltet dieselbe Struktur der einzelnen Phasen. Jedoch die Prozesse, mit der die einzelnen Phasen umgesetzt werden, unterscheiden sich stark zwischen den beiden. Die Tabelle 2.1 gibt einen groben Überblick über die Prozesse der einzelnen Phasen.

2.2.1 Erhebung

Im Gegensatz zu dem traditionellen RE, bei dem die ganzen Anforderungen geplant werden, bevor man in die nächste Phase geht, werden beim agilen RE die Anforderungen iterativ ermittelt [16]. Das bedeutet, dass mehrere Besprechungen durchgeführt werden, um nach und nach alle ANforderungen zu ermitteln. Dabei wird auf verschiedene agile Praktiken zurückgegriffen,

RE Phase	traditionelle Umsetzung	agile Umsetzung
Erhebung	Definition aller Anforderungen	Iterative Anforderungsermittlung
Analyse	Konfliktbehebung	Iterative Anpassung und Priorisierung
Dokumentation	Detailliertee Dokumentation	User Stories
Validierung	Prüfung auf Vollständigkeit	Prüfung, ob Kundenwünsche erfüllt wurden
Management	Versionskontrolle	Management über Backlog

Tabelle 2.1: Vergleich zwischen traditionellem und agilen RE

wie beispielsweise Face-To-Face Kommunikation um die Qualität der Besprechungen zu verbessern, oder User Stories, mit denen die festgelegten Anforderungen dokumentiert werden. Des weiteren ist das Erstellen von Prototypen eine oft genutzte Maßnahme, um eine bessere Vorstellung von der Umsetzung der Anforderungen zu erlangen [16].

2.2.2 Analyse (Interpretation & Abstimmung)

Bei der Analysephase des agilen RE geht es zum einen darum, durch mehrere Iterationen die vorher bestimmten Anforderungen so anzupassen, dass sie vollständig sind und keine Konflikte mit anderen Anforderungen verursachen [16]. Des weiteren wird in dieser Phase die Priorisierung der Stories durchgeführt. Im Gegensatz zur traditionellen Priorisierung werden nicht alle Anforderungen mit einem Mal bestimmt, sondern es werden nur relevante Anforderungen für eine Iteration der Entwicklung priorisiert. Bei der agilen Priorisierung werden weniger Kriterien in den eigentlich Prozess miteinbezogen, da das Hauptaugenmerk auf den Wert der Stakeholder liegt und somit auf Kriterien wie Risiken oder Abhängigkeiten weniger Wert gelegt wird [16].

2.2.3 Dokumentation

Die Dokumentation im agilen RE erfolgt nach keinem festen Schema, da sie als unzumutbar angesehen wird und nicht kosteneffizient ist [16]. Die Entscheidung darüber, welche Dokumente angefertigt werden, hängt

von dem Entwicklerteam und der Projektleitung ab, aber generell wird Dokumentation vernachlässigt. Die einzige Form von Dokumentation, die in fast allen agilen Projekten genutzt wird, ist die Dokumentation von Anforderungen in Form von USer Stories [16]. In den User Stories wird eine Anforderung in Alltagssprache festgelegt, in der beschrieben wird, welche Rolle eine bestimmte Funktion durchführt und warum sie diese Funktion benutzt.

2.2.4 Validierung/Verifikation

Beim agilen RE gibt es keinen festgelegten Ablauf, der durch die Validierung durchgeführt wird. Um sicherzustellen, dass die umgesetzten Anforderungen den Anforderungen entsprechen, die bei der Anforderungserhebung festgelegt wurden, werden in regelmäßigen Abständen Review Meetings durchgeführt, in denen den Stakeholdern der aktuelle Stand der Software präsentiert wird. Anhand dessen kann beurteilt werden, ob die Stakeholder mit der Umsetzung zufrieden sind oder noch Änderungen vorgenommen werden müssen [16].

2.2.5 Management

Da alle Anforderungen in einem Backlog festgehalten werden, kann damit der aktuelle Stand und die bereits umgesetzten Anforderungen verwaltet werden. Das einzige Problem ist, dass der Umfang der Stories mangelhaft ist, um diese über den Backlog ausreichend verwalten zu können. Dies spiegelt sich auch in der Budgetplanung wieder, da agile Projekte nicht an ein festes Budget gebunden werden können, sondern entsprechen eher einem Aufand in Form von verfügbarer Zeit für die Entwicklung [16].

2.3 Systematische Literaturrecherche

Kitchenham et al [26] beschreiben Richtlinien, um im Bereich des Software Engineering eine systematische Literaturrecherche, kurz: SLR, durchzuführen. Diese beschreiben das Identifizieren, Evaluieren und Interpretieren von allen forschungsrelevanten Daten bezüglich bestimmter Forschungsfragen oder gezielte Themenbereiche [26]. Sie selbst bezeichnen die SLR nur als sekündare Studie, die der SLR zu Grunde liegenden Quellen sind die primären Studien. Es gibt viele verschiedene Gründe, eine systematische Literaturrecherche durchzuführen, jedoch sind die folgenden nach [26] die am meisten vorkommendsten: Zum einen wird eine SLR verwendet, um bestehende Informationen bezüglich einer Technologie oder Technik zusammenzufassen,

beispielsweise die Vor- und Nachteile von spezifischen agilen Methoden. Des weiteren wird eine SLR verwendet, um bestimmte Forschungslücken aufzudecken, damit bestimmte Gebiete für die zukünftige Forschung offen gelegt wird. Der dritte Grund ist das Bereitstellen von Hintergrundwissen, um neue Forschungsgebiete korrekt einzuordnen [26].

In dieser Arbeit wird eine SLR angewendet, um Stakeholder einer Softwareentwicklung Richtlinien zu bieten, wie sie mit bestimmten Problemen und Konflikten innerhalb eines hybriden Softwareprojekts umgehen können. Dies beinhaltet nicht nur das Beheben dieser Konflikte, wenn diese bereits aufgetreten sind, sondern dient auch zur Prävention von Konflikte, wenn beispielsweise ein Unternehmen sich als Ziel gesetzt hat, hybride Entwicklungsmethoden anzuwenden, damit sie gar nicht erst auftreten.

Die systematische Literaturrecherche nach Kitchenham et al. [26] ist in drei Phasen unterteilt: die Planung, die Durchführung und die Berichterstattung der Ergebnisse.

2.3.1 Planung der SLR

Die Planung der SLR ist die erste Phase. Einige Aktivitäten in dieser Phase benötigen mehrere Iterationen, besonders in Hinblick auf die Forschungsfragen, um eine bestmögliche Durchführung zu gewährleisten. Zu Beginn muss geprüft werden, ob eine SLR überhaupt die korrekte Methode ist [26]. Eine SLR bietet sich besonders an, wenn man einen allgemeinen Überblick über einen Themenbereich erlangen will, da dieser Anhand von individuellen Studien schwer umsetzbar ist.

Der nächste Schritt ist das Festlegen von Forschungsfragen [26]. Nach diesen Forschungsfragen richtet sich der ganze weitere Verlauf der SLR. In Form von einer oder mehrerer Fragen wird der zu untersuchende Inhalt der SLR dargestellt. Anhand dieser Forschungsfragen werden die zu untersuchenden primären Studien ausgewählt. Des weiteren lassen sich bestimmte Inklusions- und Exklusionskriterien bestimmen, um bestimmte Studien für den weiteren Verlauf in die SLR mitaufzunehmen, oder wenn ein Exklusionskriterium auf eine Studie zutrifft, wird diese im weiteren Verlauf nicht mehr beachtet. Diese Kriterien reichen von oberflächlichen Themen, wie die Sprache einer Studie, bis hin zu inhaltlichen Themen, um die Menge an vorliegenden Studien besser einzugrenzen [26].

Um die SLR durchführen zu können, wird ein Suchstring ermittelt. Der Suchstring ist eine mit AND und OR verknüpfte Reinhenfolge von Termen, durch den die Forschungsfragen in Form eines einzelnen, langen Strings dargestellt werden. Dieser beinhaltet auch Synonyme und Abwandlungen von den einzelnen Termen, um eine größtmögliche Abdeckung zu gewährleisten

[26].

Dieser Suchstring wird nun als Suchkriterium in verschiedenen digitalen Bibliotheken verwendet, die sich besonders auf den Bereich des Software Engineering beziehen. Diese sind zum Beispiel: Google Scholar, ACM Digital Library, ScienceDirect, Springer Link und IEEE Explore [13]. Je nach Bibliothek muss der Suchstring angepasst werden, um die Syntax der jeweiligen Bibliothek zu entsprechen, beispielsweise arbeitet Google Scholar ohne boolsche Ausdrücke und nutzt Klammerungsregeln um diese wiederzuspiegeln.

2.3.2 Durchführung der SLR

Wurde die Planungsphase der SLR erfolgreich abgeschlossen, kann nun die eigentliche Durchführung beginnen. Man beginnt damit, in den festgelegten digitalen Bibliotheken mithilfe des Suchstrings Ergebnisse zu filtern. Um die Qualität der erlangten Studient zu analysieren, werden Checklisten angefertigt [26]. Kitchenham et al. [26] haben zwei Checklisten angefertigt, um eine Aussage über die Qualität treffen zu können, jeweils eine für die quantitative und qualitative Analyse, und schlagen vor, die Listen oder einen Teil der Listen im Kontekt der eigenen Studie zu verwenden in Verbindung mit den selbst entwickelten Forschungsfragen.

Zusätzlich zu den Checklisten werden Extraktionslisten geführt [26]. Diese werden abhängig des zu untersuchenden Themas erstellt und dienen dazu, möglichst genau die extrahierten Daten aus den Studien festzuhalten und um unvoreingenommen die Studien voneinander zu betrachten. Im Idealfall wird die Datenextraktion von zwei oder mehr Forschern durchgeführt, um die externe Validität zu reduzieren und somit die Möglichkeit auszuschließen, dass Primärstudien nur von einem Forscher übersehen werden könnten [26].

Der letzte Schitt in der Durchführungsphase ist die Datensynthese. Bei der Datensynthese werden die gesammelten Daten analysiert und zusammengefasst. Sowohl die Daten aus den Checklisten als auch die Daten aus den Extraktionslisten werden tabellarisch festgehalten und gegenübergestellt. Durch die tabellarische Form können besser die Gemeinsamkeiten und Unterschiede in den Daten gegenübergestellt und verglichen werden [26].

2.3.3 Berichterstattung der SLR

Die letzte Phase der SLR ist die Berichterstattung. Dort werden die gewonnenen Informationen aus dem Set an Primärstudien und den Extraktionslisten zusammengefasst und je nach Anwendungsfall in ein gewünschtes Format gebracht. SLRs finden in sehr verschiedenen Bereichen

statt. Sie finden Anwendung in akademischen Konferenzen oder Journal Studien, Doktorarbeiten, bis hin zu Magazinen oder Webseiten [26]. Der am meisten vorkommende bereich von SLRs sind technische Bereiche in Abschlussarbeiten, wie eine Doktorarbeit.

Kapitel 3

Forschungsdesign

Im vorherigen Kapitel wurde in den Grundlagen erklärt, wo genau die Unterschiede zwischen dem traditionellen und agilen Requirements Engineering liegen und wie eine systematische Literaturrecherche nach Kitchenham et al [26] durchgeführt wird. Nur wenige Studien und Forschungen befassen sich mit dem Thema des hybriden RE, sondern befassen sich entweder mit dem traditionellen RE oder dem agilen RE oder machen Gegenüberstellungen der beiden RE Methoden. Es gibt jedoch sehr viele Unternehmen, die vor oder bereits in der Transformation in Richtung des agilen RE stecken und somit auch auf Komplikationen stoßen werden. Um eine Übersicht über diese Konflikte zu erstellen, wird in dieser Arbeit eine systematische Literaturrecherche durchgeführt.

Zum einen wird untersucht, welche Konflikte im Zusammenhang mit dem hybriden RE entstehen können und wodurch diese entstehen. Das zweite Ziel der SLR ist es, Lösungen für die gewonnenen Konflikte zu erhalten, um eine größtmögliche Anzahl an Konflikten zum einen zu lösen und zum anderen zu präventieren, damit diese gar nicht erst entstehen. Es ist nicht das Ziel, eine perfekte Lösungsidee für jedes der Probleme zu finden, sondern eine größtmögliche Abdeckung an Ansätzen zu finden, damit trotz eines Konflikts weiter gearbeitet werden kann.

3.1 Forschungsfragen

Um die vorher genannten Ziele der SLR zu erreichen, wurden die folgenden Forschungsfragen (RQ) gewählt:

Mit RQ1 wird zunächst ein Überblick darüber gegeben, welche verschiedenen Konflikte überhaupt bei dem hybriden RE auftreten können.

RQ2 geht der Ursache der Konflikte aus RQ1 auf den Grund, um

RQ1	Welche Konflikte gibt es beim Requirements Engineering in der hybriden Entwicklung?
RQ2	Wie bzw. warum sind diese Konflikte entstanden?
RQ3	Welche Lösungen bzw. Lösungsansätze gibt es für diese Konflikte?

Tabelle 3.1: Forschungsfragen

bei bestimmten Lösungen direkt an der Ursache des Problems anzusetzen. Des weiteren werden dadurch auch Zusammenhänge zwischen verschiedenen Konflikten klarer.

Mit RQ3 werden möglichst viele Lösungen oder Lösungsvorschläge gefunden. Damit wird untersucht, welche Konflikte noch keine konkrete Lösung haben, um entweder expliziter nach passenden Lösungen zu suchen, oder eigene passende Lösungen zu entwerfen.

3.2 SLR Anwendung

3.2.1 Festlegung des Suchstrings

Um den Inhalt der in Kapitel 3.1 genannten Forschungsfragen in Form eines langen Strings wiederzugeben, wurde der folgende Suchstring gewählt:

```
(RE OR ‘‘requirements engineering’’) AND software
AND ((‘‘hybrid development’’) OR ((‘‘agile
development’’ OR agile OR ‘‘extreme programming’’
OR scrum) AND (‘‘traditional development’’ OR
‘‘large-scale’’ OR ‘‘plan-based’’ OR
‘‘plan-driven’’))) AND (challenge OR challenges
OR issue OR issues OR problem OR problems)
```

Als ersten Termin beinhaltet der Suchstring das *Requirements Engineering* selbst, da die Hauptthematik der SLR von dem RE handelt. Da in der Literatur bestimmte Themen auf verschiedene Arten beschrieben werden, wurde mithilfe von Synonymen und Abkürzungen eine größtmögliche Schnittmenge beachtet. Besonders *Requirements Engineering* wird oftmals mit der Abkürzung RE verbunden und wird in der Literatur kaum noch vollständig ausgeschrieben [1] [22] [28] usw. Dadurch wurden bereits die Studien abgedeckt, die eine beliebige Schreibweise des RE beinhalten, da diese mit einem *OR* verknüpft werden.

Der nächste Term ist *software*. Damit werden die Primärstudien eingegrenzt, damit diese sich nur auf das Requirements Engineering im Bereich des Software Engineering beziehen, da RE auch in anderen Bereichen vorkommt.

Der nächste Bereich im Suchstring bezieht sich auf die *hybriden Entwicklungsmethoden*. Im Zusammenhang mit hybriden Methoden wird zusätzlich eine Verknüpfung aus agilen Methoden und traditionellen in den String mitaufgenommen, da oftmals nicht explizit von hybrid die Rede ist, sondern der Mischung aus agilen und traditionellen Methoden. In dem Zusammenhang wurden auch verschiedenste Synonyme für die agilen und traditionellen Methoden erwähnt, um eine möglichst große Schnittmenge aus den beiden Methoden zu erhalten. Explizite agile Methoden, wie Extreme Programming und Scrum, wurden ebenfalls in den Suchstring übernommen, da die Methoden gleichzusetzen sind mit allgemeinen agilen Methoden und werden meist im selben Kontext erwähnt.

Der letzte Teil des Suchbegriffs beinhaltet das Thema der Konflikte. Dort wurden erneut verschiedene Synonyme sowie die Plural Formen gewählt, um quantitativ bessere Daten aus der SLR zu erhalten.

3.2.2 Inklustions- & Exklusionskriterien

In der Tabelle 3.2 sind die Inklusions- (IK) und Exklusionskriterien (EK) festgehalten. Diese dienen dazu, die Artikel aus der SLR zu filtern [26]. Beinhaltet ein Artikel ein Inklusionskriterium, wird dieses in der weiteren Durchführung der SLR betrachtet. Trifft ein Exklusionskriterium auf einen Artikel zu, wird dieser nicht näher betrachtet.

Das erste Inklusionkriterium *IK1* geht allgemein darauf ein, ob ein Konflikt im Zusammenhang mit dem hybriden RE, bzw. einer Mischung aus agilem RE und traditionellen RE, beschrieben wird.

IK2 beschreibt, ob ein Paper den Umgang mit einem Konflikt im hybriden RE beschreibt. Dabei geht es zum einen um die Ursachen des Konflikts zum anderen aber auch um die Lösungen oder Lösungsvorschläge für bestimmte Konflikte.

Das erste Exklusionskriterium *EK1* beschreibt den Zugriff eines Papers in einer der digitalen Bibliotheken. Aufgrund von Zugriffsbeschränkungen sind einige Paper nicht einsehbar und müssten eingekauft werden um diese zu lesen. Diese werden im weiteren Verlauf nicht weiter betrachtet.

EK2 geht auf die Sprache der vorliegenden Artikel ein. In dieser Arbeit werden für die SLR nur Artikel in Betracht gezogen, die entweder auf deutsch oder auf englisch verfasst worden.

EK3 und *EK4* betrachten den Inhalt der Paper. Hat ein Paper beispielsweise gar keinen Bezug zu hybriden Entwicklungsmethoden, wird

IK1	Das Paper beschreibt einen Konflikt im Requirements Engineering in der hybriden Entwicklung.
IK2	Das Paper beschreibt den Umgang mit einem Konflikt im RE bei der hybriden Entwicklung.
EK1	Der Zugriff auf das Paper ist nur beschränkt möglich.
EK2	Das Paper ist nicht in deutsch oder englisch verfasst worden.
EK3	Das Paper hat keinen Zusammenhang mit hybriden Entwicklungsmethoden.
EK4	Das Paper hat keinen Zusammenhang mit dem Requirements Engineering.

Tabelle 3.2: Inklusions- und Exklusionskriterien

dieses aus der SLR entfernt. Des weiteren werden nach *EK4* keine Paper weiter behandelt, die keinen Bezug zum RE haben. Zwar sollten nach dem Suchstring gar keine Paper ohne hybriden oder RE Bezug gefunden werden, jedoch ist die Schnittmenge an zu untersuchenden Artikeln sehr groß, weshalb die beiden letzten Exklusionskriterien benötigt werden.

3.2.3 Auswahl der zu untersuchenden Bibliotheken

Für die Durchführung der SLR wurde Bezug auf die folgenden fünf digitalen Bibliotheken: Google Scholar, ACM Digital Library, IEEE Explore, ScienceDirect und SpringerLink. Google Scholar, ACM Digital Library, IEEE Explore und ScienceDirect wurden ausgewählt, da diese bei den SLR Richtlinien von Kitchenham et al. [26] erwähnt wurden. Zusätzlich wurde SpringerLink als fünfte Bibliothek miteinbezogen, da man dort Zugriff auf eine Vielzahl von Journal- und Konferenz-Paper hat.

3.2.4 Durchführung der systematischen Literaturrecherche

Nachdem die Planungsphase der SLR abgeschlossen ist, kann nun die Durchführung der SLR beginnen. Der entwickelte Suchstring wurde in allen ausgewählten Datenbanken als Suchkriterium eingesetzt, nachdem er in das jeweilige Format der Bibliothek gebracht wurde.

Bei der ersten Iteration wurden Anhand der Titel der gefundenen Paper und dem Einbeziehen der Inklusions- & Exklusionskriterien gefiltert. Dadurch wurden insgesamt *312* Paper als Zwischenresultat gefunden. Bevor die nächste Iteration startet, wurden die gefundenen Paper auf Duplikate

nach der 1. Iteration	312 Paper
nach der Duplikateliminierung	272 Paper
nach der 2. Iteration	136 Paper
nach der 3. Iteration	40 Paper

Tabelle 3.3: Durchführung der SLR

untersucht. Dabei wurden insgesamt 40 Duplikate gefunden, wodurch sich die Gesamtanzahl der Paper nach dem ersten Iterationsschritt auf *272* verringerte.

Die zweite Iteration beinhaltet das Untersuchen der Abstracts in Bezug auf Inklusions- und Exklusionskriterien. Dadurch hat sich die Menge an Paper halbiert - 136 Paper sind nach der zweiten Iteration verworfen worden, wodurch sich ein Resultat von *136* übrigen Papern nach der zweiten Iteration ergibt.

Die dritte und letzte Iteration geht auf den gesamten Inhalt der zuvor ermittelten Paper ein. Die relevantesten Studien für diese Arbeit wurden beibehalten und die übrig gebliebenen Paper wurden verworfen. Das Endresultat der SLR sind *40* gefundene Paper.

3.2.5 Analyse der gefundenen Daten

Mithilfe einer Extraktionsliste wurden tabellarisch die Inhalte der 45 Paper zusammengetragen. Dort wurde zum einen Bezug genommen auf mögliche auftretende Konflikte, sowie die Lösung von Konflikten. Diese Daten werden genauer in den Kapiteln 4 und 5 betrachtet.

3.2.6 Validitätsrisiken

Die Ergebnisse einer SLR sind immer mit Risiken für die Validität dieser Resultate verbunden, die die Glaubhaftigkeit der Ergebnisse in Frage stellen [26]. Hier wird Bezug auf die interne und externe Validität explizit genommen. Die Konstruktion wurde vorab mit Kapitel 3.2.1 genauer erklärt, bei dem Synonyme verwendet wurden, um möglichst viele Bereiche abzudecken.

Interne Validität

Die interne Validität der SLR wird sichergestellt durch das Bestimmen von Inklusions- und Exklusionskriterien, wodurch alle Paper gleichermaßen anhand der selben Kriterien betrachtet werden. Zusätzlich wurden die Resultate von dem Betreuer dieser Arbeit überprüft.

Externe Validität

Die externe Validität wurde bestmöglich betrachtet, jedoch kann bei einer SLR nicht sichergestellt werden, dass jeder Artikel oder jedes Paper berücksichtigt wurde, welches in Zusammenhang mit dem Forschungsfragen steht. Des weiteren konnte diese SLR nicht von mehreren Forschern durchgeführt werden, da dies eine Abschlussarbeit ist. Mithilfe des Suchstrings wurde ein breites Spektrum an einzubeziehenden Papern ermitteln.

3.2.7 Resultate der SLR

Die anschließende Tabelle gibt einen Überblick über die Resultate der SLR. Diese werden im weiteren Verlauf dieser Arbeit vertiefend betrachtet.

Titel	Autoren
Little Design Up-Front: A Design Science Approach to Integrating Usability into Agile Requirements Engineering	Adikari, Sisira und Mcdonald, Craig und Campbell, John
Quintessence of Traditional and Agile Requirement Engineering	Bukhari, Jalil
Agile requirements engineering practices and challenges: an empirical study	Ramesh, Balasubramaniam und Cao, Lan und Baskerville, Richard
Survey on Differences of Requirements Engineering for Traditional and Agile Development Processes	Alhazmi, Alhejab und Huang, Shihong
Quality Requirements in Large-Scale Distributed Agile Projects – A Systematic Literature Review	Alsaqaf, Wasim und Daneva, Maya und Wieringa, Roel
Understanding Challenging Situations in Agile Quality Requirements Engineering and Their Solution strategies: Insights from a Case Study	Alsaqaf, Wasim und Daneva, Maya und Wieringa, Roel
Agile Quality Requirements Engineering Challenges: First Results from a Case Study	Alsaqaf, Wasim und Daneva, Maya und Wieringa, Roel
Does a Hybrid Approach of Agile and Plan-Driven Methods Work Better for IT System Development Projects?	Imani, Takeomi
Review on Agile requirements engineering challenges	Elghariani, Kaiss und Kama, Nazri
Managing requirements volatility while "Scrumming" within the V-Model	Anitha, P.C. und Savio, Deepti und Mani, V. S.
Challenges and future trends in software requirements prioritization	Babar, Muhammad Imran und Ramzan, Muhammad und Ghayyur, Shahbaz A. K.
Non-functional Requirements Documentation in Agile Software Development: Challenges and Solution Proposal	Behutiye, Woubshet et al.
Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings	Bick, Saskia und Spohrer, Kai und Hoda, Rashina und Scheerer, Alexander und Heinzl, Armin

Tabelle 3.4: Übersicht der SLR Ergebnisse

Titel	Autoren
Handling requirements using FlexREQ model	Masood Butt, Saad und Ahmad, Wan Fatimah Wan
A Reflection on Agile Requirements Engineering: Solutions Brought and Challenges Posed	Inayat, Irum und Moraes, Lauriane und Daneva, Maya und Salim, Siti Salwah
Challenges of Aligning Requirements Engineering and System Testing in Large-Scale Agile: A Multiple Case Study	Gomes et al.
A Survey of Agile and Traditional Requirements Engineering	Heba Elshandidy und Sherif Mazen
Challenges and Practices in Aligning Requirements with Verification and Validation: A Case Study of Six Companies	Bjarnason et al.
Contemporary Requirements Challenges and Issues: An Empirical Study in 11 Organizations	Chen, Feng und Power, Norah und Collins, J. J. und Ishikawa, Fuyuki
Communication Patterns of Agile Requirements Engineering	Abdullah, Nik Nailah Binti und Honiden, Shinichi und Sharp, Helen und Nuseibeh
Factoring Requirement Dependencies in Software Requirement Selection Using Graphs and Integer Programming	Mougouei, Davoud
A Hybrid Approach of Requirement Engineering in Agile Software Development	Kumar, Manoj und Shukla, Manish und Agarwal, Sonali
Towards Guidelines for Preventing Critical Requirements Engineering Problems	Mafra et el.
Using a Hybrid Method for Formalizing Informal Stakeholder Requirements Inputs	Kitapci, Hasan und Boehm, Barry W.
Requirements engineering and agile software development	F. Paetsch und A. Eberlein und F. Maurer

Tabelle 3.5: Übersicht der SLR Ergebnisse

Requirements Engineering Challenges in Large-Scale Agile System Development	Kasauli et al.
A Mapping Study on Requirements Engineering in Agile Software Development	Heikkilä, Ville T. und Damian, Daniela und Lassenius, Casper und Paasivaara, Maria
Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson	Heikkilae, V.T. und Paasivaara, M. und Lasssenius, C
T-Reqs: Tool Support for Managing Requirements in Large-Scale Agile System Development	Knauss et al.
Organisation and communication problems in automotive requirements engineering	Liebel, Grischa und Tichy, Matthias und Knauss, Eric und Ljungkrantz, Oscar
Towards Building Knowledge on Causes of Critical Requirements Engineering Problems	Kalinowski et al.
Challenges of the Customer Organization's Requirements Engineering Process in the Outsourced Environment - A Case Study	Heli Hiisilä und Marjo Kauppinen und Sari Kujala
The Problem of Consolidating RE Practices at Scale: An Ethnographic Study	ohlrab, Rebekka und Pelliccione, Patrizio und Knauss, Eric und Gregory, Sarah C.
Agile Requirements Engineering in Practice: Status Quo and Critical Problems	Wagner et al.
How Are Hybrid Development Approaches Organized? A Systematic Literature Review	Prenner, Nils und Unger-Windeler, Carolin und Schneider, Kurt
What are Hybrid Development Methods Made Of? An Evidence-Based Characterization	Tell et al.
Transition to Agile Development - Rediscovery of Important Requirements Engineering Practices	Savolainen, Juha und Kuusela, Juha und Vilavaara, Asko
Addressing Challenges of Ultra Large Scale System on Requirements Engineering	Ahmed Safwat und M.B. Senousy
Management challenges to implementing agile processes in traditional development organizations	Boehm, B. und Turner, R.

Tabelle 3.6: Übersicht der SLR Ergebnisse

Kapitel 4

Resultate

In dem folgenden Kapitel werden die Ergebnisse aus der vorher genannten systematischen Literaturrecherche genauer vorgestellt und im Detail auf die verschiedenen spezifischen Konflikte eingegangen. Man wird direkt merken, dass diese Konflikte nicht eigenständig für sich existieren, sondern es stets Verbindungen zwischen Konflikte gibt und Herausforderungen aus anderen Problemen resultieren. Explizit wird in diesem Kapitel auf die Forschungsfragen RQ1 und RQ2 eingegangen.

4.1 Anforderungserhebung

Die Anforderungserhebung beschreibt den ersten großen Abschnitt des eigentlichen Requirements Engineerings und ist die Grundlage für das weiterführende RE und der späteren Entwicklung der zugeordneten Software-Anwendung. In diesem Teil des RE werden die meisten Anforderungen definiert und dementsprechend kommt es zu einer Vielzahl von Herausforderungen, die spezifisch für die Erhebung zutreffen. Bei den hier erhobenen Anforderungen unterscheide ich die Konflikte jeweils für unvollständige und unklare Anforderungen getrennt voneinander, da sich die Konflikte hier deutlich unterscheiden, auch wenn man Verbindungen zwischen einzelnen ziehen könnte.

4.1.1 Unvollständige Anforderungen

Mangelnde Erfahrungen im RE Prozess

Das Ermitteln von Anforderungen ist ein Prozess, der von Menschen durchgeführt werden muss. Es gibt keine Möglichkeit, diese automatisiert zu erheben, da die Anforderungen existieren, um ein bestimmtes Problem von

einer Gruppe von Personen auf eine Art zu lösen, die so genau wie möglich auf diese Personengruppe zugeschnitten ist. Daher spielt der Mensch selbst eine entscheidende Rolle für aufkommende Konflikte. Nach [23] ist der Mensch für 41% der Probleme schuldig, die mit unvollständigen Anforderungen zusammenhängen und ist somit der größte alleinige Verursacher von Problemem dieser Art. Dies beruht vor allem auf ein mangelndes Wissen [32]. Bei 2,9% ist dies mangelndes Wissen als Product Owner, bei 2,9% zu wenig theoretische Erfahrung im RE und bei über 29% fehlt die praktische Erfahrung von RE Praktiken [23].

Keine einheitliche Dokumentenstruktur

Ein weiteres großes Problem ist, dass Anforderungen in verschiedenen Projekten auf unterschiedliche Weisen festgehalten werden. Besonders da meistens die Stakeholder nicht nur an einem Projekt, sondern direkt an mehreren beteiligt sind, fehlt somit der Lernprozess und die Übersicht, die man aus vorherigen Projekten mitnehmen würde [23] [32]. Es existiert kein Template bzw. keine Vorlage, nach der man den Anforderungserhungsprozess dokumentiert, sondern die verschiedenen Teams nutzen jeweils andere Varianten für die Dokumentation [32]. Dadurch entsteht eine Vielzahl an Dokumenten und der Überblick über diese ist entsprechend schlecht.

Mangel an Dokumenten

Je mehr ein Unternehmen den Schritt in Richtung Agilität wagt, desto deutlicher ist dies am Umfang der Dokumente zu sehen [19] [22]. Nicht nur wird die Struktur bei den Dokumenten nicht einheitlich benutzt [32], sondern es werden auch Dokumente gar nicht mehr angefertigt [22]. Dies betrifft nicht nur das Ermitteln von Anforderungen, wo diese beispielsweise in Form von User Stories festgehalten werden, sondern zieht sich auch durch den weiteren Verlauf des Projekts. Seien dies mangelnde Teile der Spezifikation, Dokumentation der Testfälle oder Übersichten über die Architektur des zu entwickelnden Systems, je mehr man hybride Methoden bis hin zu agilen Methoden anwendet, desto deutlicher ist dies anhand der Dokumente erkenntlich [22]. Grund dafür ist der Fokus auf die zu entwickelnde Anwendung selbst, damit sie möglichst viele Anforderungen entspricht, weshalb weniger Zeit in die Anfertigung einer Dokumentation gesteckt wird [22] [19] [18].

Anpassungen von Anforderungen

Resultierend aus einer lückenhaften Dokumentation ergibt sich, dass sich die spätere Anpassung von Anforderungen schwierig gestaltet [15]. Besonders

wenn man mehr in die Richtung der Agilität geht, bei der generell schon die Dokumentation eine geringere Priorität hat [19] [22], und man Anforderungen in Form von User Stories festhält, sei dies in Form von physischen Karten oder digital in einem Backlog verwaltet, ist das Ändern und das dazugehörige Zurückverfolgen der Änderungen schwierig [15].

Keine Beschreibung von Fehlerinteraktionen

Ein weiteres großes Problem beim Erheben von Anforderungen ist die nicht vorhandene Beschreibung von Fehlerinteraktionen [25]. Besonders die Auftraggeber gehen von einer voll funktionsfähigen Anwendung aus und beachten deshalb nur korrekte Eingabewerte. Besonders dann, wenn eine Anwendung verbreitet wird, sodass verschiedene Menschen, auch unerfahrene Menschen, in Kontakt mit der Software kommen, muss berücksichtigt werden, wie sich die Anwendung bei einer ungewollten Interaktion verhält und welche Informationen an den jeweiligen Nutzer zurückgegeben werden, damit dieser mit dem Fehlverhalten umgehen kann und mit der Anwendung weiter arbeiten kann [25].

Kommunikation

Die Kommunikation spielt besonders zu Beginn des eigentlichen Requirements Engineerings eine große Rolle. Viele verschiedene Persönlichkeiten treffen aufeinander mit den unterschiedlichsten Erwartungen an das Endprodukt [23] [32]. Die Kommunikation bringt so viele Herausforderungen mit sich, dass sie im weiteren Verlauf noch in einem eigenen Abschnitt genauer betrachtet wird, jedoch spielt sie in der Erhebungsphase eine sehr wichtige Rolle.

Anforderungen werden erhoben, in dem die Stakeholder sich kommunikativ austauschen. Durch verschiedenste Persönlichkeiten kann es daher zu Meinungsverschiedenheiten kommen oder es wird schlichtweg nicht klar, welche Personen wieviel Wert auf bestimmte Funktionen legen [23] [32] [24], sodass das Endprodukt nicht den Wünschen aller Beteiligten entspricht.

Ableiten von Tests

Jede Anwendung muss ausgiebig getestet werden, bevor man diese an den Kunden geben kann. Mithilfe von Tests wird ein mögliches Fehlverhalten ausgeschlossen bei bestimmten Interaktionen der späteren Anwender, seien dies korrekte oder fehlerhafte Eingaben. Diese Tests müssen sich an bestimmte Vorgaben halten, die die Stakeholder beschlossen haben - die Anforderungen. Anhand von dokumentierten Anforderungen, zum Beispiel in Form einer Spezifikation oder User Stories [15], werden meist schon während

des Entwicklungsprozesses Tests geschrieben. Wenn jedoch die der Tests zu Grunde liegende Dokumentation unvollständig ist, wirkt sich das auf die Tests aus [17]. Testfälle können nicht nur übersehen werden, sondern betrachten die Auswirkung von einigen Interaktionen falsch, sodass es zu False-Negative Testergebnissen kommen kann, also, dass Tests als fehlerhaft gekennzeichnet wurden, obwohl sie einen korrekten Fall beschreiben [17].

4.1.2 Unklare Anforderungen

Der Unterschied zwischen unvollständigen und unklaren Anforderungen ist ein schmaler Grad. Jedoch sollte man trotzdem zwischen diesen unterscheiden. Unvollständige Anforderungen resultieren meistens darin, dass spätere Schritte im Requirements Engineering bzw. des ganzen Projektablaufs unter den mangelhaften Anforderungen leiden [23]. Unklare Anforderungen hingegen resultieren eher in Verständnisprobleme der beschriebenen Anforderungen.

Missverständnisse zwischen den Stakeholdern

Der erste große Punkt im Zusammenhang mit unklar beschriebenen Anforderungen sind auftretende Missverständnisse zwischen den Stakeholdern, wie beispielsweise die Projektleitung und die Entwickler selbst [17]. Unklar im Zusammenhang mit Anforderungen bedeutet nicht nur, dass aus der Formulierung der Anforderungen nicht die genaue Problematik klar wird, sondern dies bezieht sich auch auf versteckte Anforderungen, die nicht auf den ersten Blick eindeutig werden [32]. Durch verschiedene Blickwinkel auf eine Anforderung besitzen die Stakeholder unterscheidliche Meinungen zu einer bestimmten Anforderungen, auch bezüglich der späteren Priorisierung. Durch unklar definierte oder versteckte Anforderungen, die beispielsweise in anderen Anforderungen miteinbezogen worden, kommt es zu Missverständnissen zwischen den Stakeholdern, da zum Beispiel keine *Definition of Readiness & Definition of Done* gemeinsam festgelegt wurde [32]. Das bedeutet, dass nicht eindeutig festgelegt wurde, zum einen wann eine Anforderung soweit beschrieben wurde, dass sie *bereit* (ready) für die weitere Projektplanung und zum anderen, ab wann eine Anforderung als *abgeschlossen* (done) gilt in der späteren Entwicklungsphase.

Unklares User Story Format

Eine der ersten Maßnahmen im Übergang vom traditionellen RE in das hybride RE, bis in Richtung der Agilität, ist das Festlegen von Anforderungen

in Form von User Stories [19] [32] [17]. Mithilfe von User Stories werden Anforderungen an eine Software-Anwendung in einer Alltagssprache formuliert, die darauf eingeht, *wer was* tun möchte und *warum* er dies tun möchte. Sind die beteiligten Personen an dem Anforderungsermittlungsprozess nicht genügend geschult im Umgang mit Stories, besteht die Gefahr, dass die Stories zum einen falsch geschrieben werden, sodass der Sinn der User Stories entfällt, und zum anderen dass die Anforderungen im User Story Format fehlinterpretiert werden. Außerdem eignen sich User Stories nicht dafür, wenn die Anforderungen an Stakeholder weitergegeben werden müssen, die nicht am selben Standort sind [18], da sie in den meisten Fällen mithilfe eines Verwaltungstools digital festgehalten werden müssen und man dementsprechend die Stakeholder Zugriff auf das System ermöglichen muss.

Komplexität der Anforderungen

Software-Anwendungen werden immer umfangreicher und decken einen größeren Aufgabenbereich ab. In diesem Zusammenhang werden natürlich auch die Anforderungen an eine neue Anwendung stetig umfangreicher und komplexer [11]. Diese Komplexität muss sich in Richtung des hybriden und agilen RE in Form der User Stories wiederspiegeln. Da User Stories nur mithilfe von wenig Worten und durch Alltagssprache die Anforderungen ausdrücken sollen, sind sie eher ungeneigt, um komplexe Kundenwünsche festzuhalten [18], sondern richten sich an kurze und prägnante Umgangsformen mit dem zukünftigen System.

Anpassungen von Anforderungen

Nach Tukur et al. [46] sind Änderungen und Anpassungen an bereits bestehende Anforderungen eine der größten Herausforderungen eines jeden RE Prozesses. Auch mit längerer Vorausplanung des Projekts sind nicht zu Beginn Anforderungserhebung alle Anforderungen bestimmbar bzw. es ist nicht vollständig klar, wie sie umgesetzt werden sollen [32]. Dadurch sind Änderungen im weiteren Verlauf unausweichlich und bereits bei klar definierten Anforderungen sind Anpassung nur schwer zu kontrollieren [46]. Sind sie dazu noch nicht eindeutig festgelegt werden, müssen öfter als nötig Anpassungen durchgeführt werden und der Überblick geht verloren. Besonders da die Kunden ohnehin Anpassungen durchführen werden, weil sich beispielsweise der Kontext oder die Technologie ändert [46], würden zusätzliche, nicht notwendige Änderungen an Anpassungen das Budget der Planung übersteigen.

4.2 Priorisierung von Anforderungen

Mangelhafte Priorisierung

Mit dem Priorisieren von Anforderungen wird entschieden, welche Anforderungen zu Beginn der Entwicklung umgesetzt werden und welche Anforderungen an das Ende der Entwicklung geschoben werden, um möglichst viele wichtige Themen umzusetzen und um bei einer Änderung des Zeitplans nur Anforderungen zu streichen, die von weniger Bedeutung sind [20]. Bei einer mangelhaften Priorisierung ist das Problem, das bestimmte Anforderungen nicht in die Priorisierung miteinbezogen wurden [11], da sie durch die Anforderungsspezifikation nicht betrachtet wurden. Dadurch, dass die eigentlich Entwicklungsphase sehr kurz nach der Priorisierung bereits beginnt, lassen sich fehlende Anforderungen nur schwer in den weiteren Projektablauf eingliedern [32], da sich dadurch der ganze weitere Ablauf ebenfalls an die neue Priorisierung anpassen müsste.

Inkorrekte Priorisierung

Die Inkorrekte Priorisierung bezieht sich, im Gegensatz zu fehlenden Anforderungen, auf das falsche Priorisieren. Durch Unklarheiten der Stakeholder, was das eigentliche Ziel der Anwendung sein soll und in welchem Umfang die Anwendungsfunktionen umgesetzt werden sollen [22], wird eine falsche Reihenfolge der Umsetzung der Anforderungen festgelegt. Ebenso wie bei der mangelhaften Priorisierung lässt sich im Nachhinein die Reihenfolge nur schwer anpassen, da die Entwicklung bereits begonnen hat [32]. Wird die Entwicklung in Form von Sprints durchgeführt, sodass in regelmäßigen Abständen eine Besprechung mit den Stakeholdern stattfindet, lassen sich eventuelle Fehler in der Priorisierung korrigieren, jedoch wird dadurch das Grundproblem noch nicht gelöst [22] [33].

Priorisierung & Aufwandsschätzung von User Stories

Im hybriden RE und besonders auch im agilen RE, werden Anforderungen mittels User Stories festgehalten, damit diese verständlich und in einem kurzen Format beschrieben werden [18]. Im weiteren Verlauf des RE untergehen die Stories einen Prozess der Aufwandsschätzung. Dabei wird jeder Story ein Wert zugewiesen, den Story Points. Diese beschreiben den Umfang der damit beschriebenen Anforderung im Vergleich zu Anforderungen, die bereits in vergangenen Projekten umgesetzt wurden und einen ähnlichen Umfang haben. Die Aufwandsschätzung wird ebenfalls in den Priorisierungsprozess miteinfließen [24]. Da besonders bei Unternehmen, die sich gerade einen

Umstieg von traditionellem zum hybriden RE befinden, nicht jeder Beteiligte mit dem User Stories und der dazugehörigen Aufwandsschätzung vertraut ist, wird dadurch die Priorisierung der Anforderungen erschwert [24].

Re-Priorisierung

Das Problem der Re-Priorisierung von Anforderungen tritt bei Unternehmen besonders dann auf, wenn die Projektplanung eher in Richtung des traditionellen RE abläuft [22]. Dabei wird das Projekt komplett durchgeplant, bevor man in die Entwicklung geht und während der Entwicklung werden kaum bis gar keine Änderungen an der Planung vorgenommen, da in dieser Zeit die Entwickler keinen Kontakt mit den restlichen Stakeholdern haben. Dadurch lassen sich Änderungen, die kurzfristig entdeckt wurden und einen Einfluss auf die Priorisierung der bereits geplanten Anforderungen haben, nicht in den weiteren Planungsablauf miteinbeziehen [22].

4.3 Verwaltung des Backlogs

Mangelnde Erfahrung

Der Backlog ist eine der ersten Änderungen, die bei der Einführung von hybriden und agilen Entwicklungsmethoden umgesetzt wird [43]. In dem Backlog werden die erstellten User Stories abgelegt, die Verknüpfung unter ihnen wird hergestellt und der Aufwand wird abgeschätzt. Diese Methode des Festhaltens von Anforderungen ist bei dem traditionellen RE unbekannt und findet dort keine Anwendung [45]. Dadurch sind die Beteiligte des RE Prozesses nicht mit dem Umgang mit User Stories vertraut, da ihnen die nötige Erfahrung mit ihnen fehlt, sowohl theoretisch als auch praktisch, wodurch sich der Umgang mit den User Stories vorallem zu Beginn einer Umstellung als schwierig darstellt [45]. Deshalb muss zusätzliche Zeit und Budget eingeplant werden, um den RE Prozess vollständig durchzuführen.

Zu schnelle Transformation

Besonders Unternehmen mit IT-Bezug stehen stetig unter Druck, jederzeit auf dem neuesten Stand zu sein und moderne Methodiken in ihren Arbeitsalltag einfließen zu lassen [25] [42]. Dadurch kommt es zu einem überstürzten Drang zu einer Transformation, der besonders starken Einfluss auf aktuelle Unternehmen nimmt, die noch mit traditionellen RE Maßnahmen arbeiten [32]. Wird nicht genügen Zeit eingeplant und somit ein Umschwung regelrecht aufgezwungen, wirkt sich dies auf die verschiedensten Bereichen der RE

Prozesse aus, von der mangelhaften Dokumentation [32], über Kommunikationsprobleme [9] [23], bis zu Konflikten, die sich auf mangelde Erfahrung mit den neuen RE Prozessen beziehen [14] [31].

Inhalt der User Stories

User Stories sind von sehr großer Bedeutung für den weiteren RE Prozess und dem darauf aufbauendem weiteren Vorgehen [43]. Daher muss der Inhalt der Stories den Inhalt der Anforderungswünsche aller Stakeholder gerecht werden und darf keine Lücken beinhalten. Da am Prozess der Erstellung der User Stories Stakeholder aus den verschiedensten Bereichen beteiligt sind, ist es nicht einfach, den vollständigen Inhalt aller Anforderungen wiederzugeben, besonders weil im hybriden RE unterschiedliche Umgangsweisen mit Anforderungen und User Stories aufeinandertreffen [24].

Aktualität der User Stories

Nicht nur der generelle Inhalt der User Stories wirkt sich auf die weitere Planung und die spätere Entwicklung aus, sondern auch die Aktualität der Stories [45]. Im hybriden RE werden meist die Anforderungen nicht an einem Stück durchgeplant, sondern durchlaufen einen iterativen Prozess [46]. In diesem iterativen Prozess werden nicht nur neue Anforderungen stetig hinzugefügt, sondern es werden auch Änderungen an bereits bestehenden User Stories vorgenommen, die ebenso dokumentiert werden müssen. Jedoch bleiben diese Änderungen in den meisten Fällen auf der Strecke, da der Fokus auf das Festlegen von neuen Anforderungen besteht [45], wodurch sich Inkonsistenzen zwischen den Anforderungen bilden, da der Aktualisierungsstand zwischen ihnen variiert.

4.4 Konflikte mit Stakeholdern

Stakeholderidentifikation

Mit der Einführung von hybriden RE Prozessen wird zusätzlich zu neuen generellen Maßnahmen auch neue Rollen eingeführt, die von verschiedenen Stakeholdern eingenommen und ausgeführt werden müssen [43]. Je nach Anteil, wie stark ein Unternehmen mit hybriden Maßnahmen in die Richtung der traditionellen bzw. agilen Maßnahmen geht, wird beispielsweise Scrum angewendet, wodurch beispielsweise die Rolle des Scrummasters ausgeführt werden muss. Wenn nun ein Unternehmen die Transformation von traditionellem RE zu hybriden RE durchführt und demnach auch die zusätzlichen

Rollen einführt, fällt es den Stakeholdern schwer zu identifizieren, wer welche Rolle in dem RE Prozess einnimmt [14]. Werden zusätzlich in dem Planungsprozess noch externe Mitarbeiter einbezogen, die auch andere Planungsmethoden verwenden, wird der Prozess noch zusätzlich erschwert, da die nötige Erfahrung mit dem Umgang der Situation fehlt [46].

Mangelnder Anforderungsaustausch

Besonders in Unternehmen, die sich sehr start in der IT-Umgebung etabliert haben, gibt es eine sehr große Anzahl an unterschiedlichen Stakeholdern, vom IT Management über Systemarchitekten, bis zu den eigentlichen Entwicklern [20], welche die einzelnen Rollen des hybriden RE übernehmen. Durch die Vielzahl an Beteiligten bei der Bestimmung von Anforderungen, kommt es erschwert dazu, dass nicht die Vorstellungen aller Stakeholder in den Anforderungen berücksichtigt werden, da der Austausch mit allen Beteiligten über jede Anforderung nahezu unmöglich ist, um im zeitlichen Plan des Projekts zu bleiben [20] [46]

Unklare Verantwortlichkeiten

Zusätzlich zu der Identifikation der Rollen im hybriden RE muss auch festgelegt werden, welcher Stakeholder für welchen Bereich verantwortlich ist. Dabei wird nicht bestimmt, wer welchen Aufgabenbereich bei der Planung übernimmt, sondern wer bei der späteren Umsetzung des Systems für welchen Teil des Systems zuständig ist. Generell bei bereits erfahrenen Unternehmen mit dem hybriden RE Ablauf stellen die Verantwortlichkeiten eine Hürde dar, aber bei unerfahrenen Unternehmen, die bereits mit der Festlegung der einzelnen Rollen zu kämpfen haben, erschwert das Festlegen der Verantwortlichkeiten zusätzlich das weitere Vorgehen [31].

Mangelnde Unterstützung der Projektleitung

Die Projektleitung ist dafür zuständig, alle Prozesse des gesamten Projektes zu koordinieren. Dabei muss auf die verschiedenen Vorstellungen der Anforderungen und deren Umsetzung bei einer Vielzahl von Stakeholdern eingegangen werden. Da besonders bei Großprojekten die Stakeholder nicht jederzeit alle an einem Ort tätig sind, ist es für die Leitung schwierig, alle Beteiligten des Projektes miteinzubeziehen [46]. Besonders hybride und agile Projekte beinhalten mehrere kleinere Teams, die an unterschiedlichen Orten tätig sind, wodurch es für die Projektleitung nicht umsetzbar ist, jedem Team die benötigte Aufmerksamkeit zu schenken und dies anhand von Prioritäten geschehen muss [25].

4.5 Kommunikations- & Koordinationsprobleme

Koordinationsprobleme

Die Koordination von vielen verschiedenen Teams an unterschiedlichen Standorten ist besonders bei der Umstellung von traditionellem zu hybridem RE ein großer Konflikt [9]. Durch die Umstellung auf hybrid werden einzelne große Teams in mehrere kleinere Teams unterteilt. Dadurch erweitert sich die Koordination der Teams, da zum einen die Koordination innerhalb der einzelnen Teams gewährleistet werden muss und zusätzlich muss, wie bereits zuvor, die Koordination zwischen den einzelnen Teams verwaltet werden, welches sich durch eine größere Anzahl an Teams noch einmal deutlich erschwert [9]. Dadurch muss die Priorisierung, Aufwandsschätzung und die spätere Zuweisung der der Verantwortlichkeiten für jedes einzelne Team koordiniert und auf alle Teams gleichmäßig eingegangen werden [9]

Informationsaustausch zwischen Stakeholdern

Nicht nur der Austausch zwischen den einzelnen Teams wird durch das aufsplitten erschwert, zusätzlich muss jedes Team dem Rollenschema des hybriden RE entsprechen. Dadurch wird die Kommunikation zwischen den Führungsrollen der einzelnen Teams erschwert, da diese vorher in einem großen Team zusammen gearbeitet haben [24]. Daraus resultiert, dass zusätzlicher Aufwand nötig ist, damit spezifische Rollen, wie die Scrum Master der einzelnen Teams, auf einem aktuellen Stand sind [23] [5].

Aufteilung der Teams

Durch die Aufteilung in kleine Teams, vor allem zu Beginn einer Transformation, werden die Mitarbeiter aus ihrem alltäglichen Arbeitslesen gerissen, da sie dadurch in den meisten Fällen andere Kollegen besitzen als vorher [24]. Beispielsweise arbeiten bei den traditionellen Maßnahmen alle Systemarchitekten teilweise zusammen und um in die Richtung der hybriden Maßnahmen zu wechseln, wird jedem Team ein Systemarchitekt zugewiesen. Damit wird gewährleistet, dass in jeder Gruppe verschiedene Probleme eigenständig gelöst werden, ohne dabei auf Unterstützung von anderen Teams zurückzugreifen [32]. Jedoch sind damit neue Arbeitskollegen verbunden und die einzelnen Mitarbeiter benötigen eine gewisse Zeit für die Umstellung.

Mangelnder Austausch bezüglich Anforderungen

Aufgrund der neuen Teamstruktur hat sich die Anzahl der Personen, mit denen die Mitarbeiter zusammenarbeiten, deutlich verkleinert [32]. Dadurch haben die Teammitglieder auch automatisch weniger Kontakt mit anderen Teams, was sich vorallem bei der Erhebung der Anforderungen bemerkbar macht. Besonders wenn ein Team Änderungen an bereits bestehenden Anforderungen im Nachhinein vornimmt, wird dies durch die mangelnde Kommunikation mit anderen Teams nicht direkt klar und anderen Teams bemerken in den meisten Fällen gar nicht, ob und welche Anforderungen geändert wurden [24]. Besonders im hybriden RE treffen verschiedene Ansichten des RE aufeinander, weshalb die Teams strikter voneinander getrennt werden, damit die entsprechenden Maßnahmen besser eingeführt und umgesetzt werden können [32].

Ressourcenmangel

Besonders bei der Umstellung zu hybriden Entwicklungsmaßnahmen wird viel Budget benötigt, um die Umstellung so reibungslos wie möglich zu gewährleisten, indem in den meisten Fällen entsprechende Experten aus externen Quellen genutzt werden [46]. Dadurch muss an bestimmten Stellen gespart werden, und dies ist in den meisten Fällen die Kommunikation [32] [46] [24]. Es wird nicht genügend Zeit miteingeplant, die zusätzlich für den Austausch der Mitarbeiter benötigt wird, was zum einen zu Erfahrungsproblemen bei den Mitarbeitern führt [14], aber auch an der Ausformulierung der Anforderungen, besonders bei denjenigen, die von mehreren Teams gleichzeitig bearbeitet werden [46].

4.6 Nicht-funktionelle Anforderungen

Ignorieren von NFRs

Nicht-funktionelle Anforderungen, auch Qalitätsanforderungen genannt, sind Anforderungen, die nicht beschreiben, was genau eine Software tun soll, sondern wie die Umsetzung der Anforderungen in der Software selbst realisiert wird. Dies beinhaltet Werte wie Performance, Sicherheit, Wartbarkeit und Zuverlässigkeit [8]. Im Vergleich zum traditionellen RE werden NFRs im hybriden RE zum Großteil vernachlässigt, da der Fokus der Beteiligten im RE Prozess darauf liegt, welche Funktionen die Software am Ende beinhalten soll [4]. Dazu kommt, dass Anforderungen in dem Format der User Stories abgelegt werden. Diese sollen in kurzer und verständlicher Weise

Anforderungen wiedergeben und sind nicht für die Ablage von komplexen NFRs gedacht [15] [8].

Mangelndes NFR Verständnis

Die Nicht-Funktionellen Anforderungen werden jedoch nicht einfach nur ignoriert, weil sie vergessen wurden oder die Priorität nicht unbedingt bei ihnen liegt, sondern auch, weil den Beteiligten beim Requirements Engineering das nötige Verständnis fehlt, um NFRs festzuhalten [46]. Beim RE Prozess selbst wird mit den Kunden Rücksprache gehalten und es werden Anforderungen entsprechend nach Kundenwünschen festgelegt. Die Umsetzung der Anforderungen selbst, bleibt im Grunde den Entwicklern überlassen, da am Ende nur eine funktionierende Software von Interesse ist, die die nötigen Funktionen beinhaltet [40] [2] [3]. Da keine NFRs genau festgelegt wurden und die einzelnen Teams ohnehin schon Probleme haben, stetig im Austausch zu bleiben, kann es zu unterschiedlichen Umsetzungen der Anforderungen kommen [16].

Softwarearchitektur

Ein besonderer Teil, dem im RE Prozess der meisten Projekte wenig Beachtung zu Gute kommt, ist die Architektur der späteren Softwareanwendung [40]. Dies wird besonders ein großes Problem, wenn im späteren RE Prozess zusätzliche Anforderungen festgelegt werden, die Einfluss auf die Architektur nimmt, deren Entwicklung dann bereits begonnen hat [40], da die Anpassung der Architektur selbst sehr umständlich ist und in den meisten Fällen, je nach Anforderung, eine komplett neue Architektur entworfen werden muss. Dadurch verschiebt sich der komplette weitere Ablauf der Entwicklung [22].

4.7 Kosten- & Zeitermittlung

Schwierige Budgetplanung

Die Planung des Budgets ist ein Prozess im Requirements Engineering Prozesses, bei dem zum einen das Budget für die Planungsphase des RE berechnet wird, aber auch die Entwicklungsphase, nachdem das RE abgeschlossen wurde [32]. Durch zunehmende Agilität in dem hybriden RE Prozess kommt es wie bereits in den vorherigen Kapiteln erwähnt zu Anpassungen, da neue Anforderungen hinzukommen oder bestehende Anforderungen geändert werden. Zusätzlich zu der Gesamtbudgetplanung des Projekts müssen auch die einzelnen Sprintphasen gut durchstrukturiert

und geplant werden, welches sich schwierig gestaltet, da man zum einen auf langer Sicht planen muss, jedoch auch die kurz bevorstehenden Sprintphasen einbeziehen muss. Dadurch ist es schwierig, eine Balance zu finden zwischen einer Langzeitplanung und Kurzzeitplanung des Projektbudgets [39].

Kostensteigerung

Erfahrungswerte haben gezeigt, dass die Kostenermittlung zu Beginn eines jeden Projekts sich als schwierig gestaltet und in den meisten Fällen die Anfangs bestimmten Werte nachgebessert werden müssen, da es schwierig ist, am Anfang eines Projekts das komplette Ausmaß zu bestimmen [15]. Besonders bei der Planung der Entwicklung werden nur die aktuell ermittelten User Stories in die Planung miteinbezogen und anhand der zugewiesenen Story Points kann der Aufwand berechnet werden [16]. Da jedoch in den meisten Fällen weitere Stories erstellt werden, wird es zu einer Kostenerhöhung kommen [16]. Des weiteren ist die Kommunikation, sowohl innerhalb der Teams, als auch zwischen den Teams, eine Ursache für erhöhte Kosten eines Projekts, da sich aus mangelhafter Kommunikation ein Mehraufwand für alle Beteiligten ergibt [17].

Mangelndes Zeitmanagement

Der dritte Punkt im Zusammenhang mit der Kosten- & Zeitermittlung ist ein mangelhaftes Zeitmanagement. Zum einen ist dies bezogen auf die User Stories und der Aufwandsschätzung in Form von Story Points. Da die Story Points einen abstrakten Wert für einen Aufwand darstellen, der durch Erfahrungswerte in Form von Vergleichswerten mit ähnlich aufwendigen Anforderungen entsteht, spiegeln diese keinen genauen Wert wieder und sich besonders abhängig von der Erfahrung der Entwickler [32]. Zum anderen wird für die Entwicklungsphase selbst nur die direkte Umsetzung der Anforderungen eingeplant. Das bedeutet, dass keine zusätzliche Zeit eingeplant ist, um die Anforderungen zu verwalten und zu verstehen [31]. Besonders dadurch, dass die Anforderungen von vielen unterschiedlichen Stakeholdern geschrieben wurde, ist nicht direkt klar, was genau mit einer Anforderungen gemeint ist und wie diese auch technisch umzusetzen ist [40].

4.8 Ungleichmäßige Erfahrungsverteilung

Ungenügende Stakeholdererfahrung

Durch die große Vielzahl an zur Verfügung stehenden Maßnahmen und genutzten Maßnahmen, muss bei allen Stakeholdern ein gewisses Grundwissen über die RE Prozesse vorausgesetzt sein, um mit einem zufriedenen Ergebnis den RE Prozess abschließen zu können [14]. Nach Chen et al. [14] wurde bei der Hälfte der Befragten angemerkt, dass die Stakeholder nicht genügend Erfahrung besitzen, sowohl aus der Management Perspektive, als auch aus der technischen Perspektive. Dadurch ist der ganze weitere RE Prozess benachteiligt, da die Stakeholder im RE das Grundgerüst für das weitere Vorgehen festlegen. Ebenfalls sind die Product Owner davon betroffen, mit nicht genügendem Vorwissen in in weitere Verhandlung über die Anforderungen zu gehen, wodurch die Wünsche der Auftraggeber nicht in vollem und korrekten Maß an die weiteren Stakeholder gegeben werden [23] [12].

Unterschiedliche Erfahrungswerte

Bezüglich der Erfahrung gibt es nicht nur das Problem, dass nicht genügend vorhanden ist, sondern dass die Erfahrung nicht annährend gleichmäßig verteilt ist [31]. Gerade dadurch, dass in den hybriden Entwicklungsmethoden in kleineren Teams gearbeitet wird, muss gewährleistet sein, dass jedes Team eigenständig arbeiten kann. Jedoch fehlt der direkte Austausch zwischen den Führungsrollen der einzelnen Teams, da sie komplett getrennt voneinander arbeiten. Dadurch können Erfahrungswerte aus vergangenen Projekten nur schwer an die korrekten Stakeholder weitergeleitet werden, da man nicht mehr mit ihnen zusammenarbeitet [31]

Technische Anforderungen

Wie bereits in Kapitel 4.6 erwähnt, werden Nicht-Funktionelle Anforderungen meistens im RE Prozess nicht beachtet, müssen jedoch trotzdem festgelegt werden, um eine konsistente Entwicklungsphase zu gewährleisten. Das Problem ist aber, dass die Requirements Engineers, die die Anforderungen entsprechend entwerfen, nicht genügend mit den technischen Themen vertraut sind wie die Entwickler, die für die Umsetzung zuständig sind [46]. Dadurch werden mangelhafte technische Anforderungen festgelegt von Personen ohne oder mit nur wenigem technischen Hintergrund, weshalb diese zusätzlich im Entwicklungsprozess angepasst werden, damit die Software so

umgesetzt werden kann, wie von den funktionellen Anforderungen beschrieben [46].

4.9 Verifikation/Validierung

Testabdeckung

Mithilfe von Verifikation und Validierung wird bestimmt, dass die entwickelte Software Anwendung inhaltlich den Anforderungen der Kunden entspricht und die sie so umgesetzt wurden, wie es die Stakeholder festgelegt haben. Mithilfe von verschiedensten Tests wird die Umsetzung der Software mit den zugrunde liegenden Anforderungen verglichen und geprüft, ob das Endprodukt den Kundenanforderungen entspricht [10]. Durch unklare und unvollständige Tests, sowie undokumentierte Änderungen an Anforderungen kann eine vollständige Testabdeckung nicht hergestellt werden und somit kann kein vollständiges Urteil über die Verifikation und Validierung getroffen werden [10].

Fehlender Verifikationsprozess

Der Verifikationsprozess steht in direkter Verbindung zu der Testabdeckung bezüglich den Anforderungen [10]. Im hybriden und agilen RE werden verschiedene Prozesse gekürzt, zusammen mit anderen Prozessen werden neue Prozesse entworfen oder andere Maßnahmen entfallen komplett. Dazu gehört auch der Verifikationsprozess aus dem traditionellen RE [10]. Je mehr ein Unternehmen in Richtung des agilen RE arbeitet, desto weniger liegt der Fokus auf Prozesse wie der Verifikation, sondern auf ein möglichst umfangreiches Endprodukt. Dazu kommt, dass einzelne Teams ihre eigenen Prozesse entwickeln, wozu auch die Verifikation gehört, wodurch ein Chaos an verschiedenen Prozessen entsteht, die besonders in großen Projekten ineffizient funktionieren [10].

Verifikation von NFRs

Bei den Tests einer Anwendung liegt die eigentliche Funktionsweise im Vordergrund, also die funktionellen Anforderungen, im Vergleich zu dem technischen Hintergrund [10]. Dies liegt unter anderem auch daran, dass, wenn überhaupt NFRs beschrieben wurden, nicht auf die genauen Details der Umsetzung eingegangen wird, sondern diese nur grob beschrieben wurden. Daher lassen sich daraus keine genauen Tests für die Verifikation ableiten. Falls diese jedoch trotzdem mit Tests abgenommen werden müssen, ist es

einfacher, die Anforderungen selbst anzupassen an Systemarchitektur, als im Nachhinein das System so umzubauen, wie es in den NFRs beschrieben wurde [10].

Kapitel 5

Lösungen & Diskussion

In diesem Kapitel wird die Forschungsfragen RQ3 genauer behandelt. Das bedeutet, es werden Lösungen zu aus dem vorheringen Kapitel genannten Konflikte genannt, die zum einen aus der vorhandenen Literatur hervorgingen, und zum anderen eigenen Lösungsvorschläge beinhalten, da nicht zu jedem Konflikt eine Lösung in der Literatur genannt wurde. Des weiteren wird jeweils diskutiert, in welche Richtung, traditionell oder agil, ein Unternehmen gehen müsste, um möglichst gut die entsprechenden Konflikte abwenden zu können.

5.1 Anforderungserhebung

5.1.1 Unvollständige Anforderungen

Mangelnde Erfahrungen im RE Prozess

Das Problem der mangelnden Erfahrung kann nicht innerhalb eines kurzen Zeitraums bewältigt werden. Erfahrung wird nur gewonnen, entweder in dem die Mitarbeiter selbst die nötige Erfahrung sammeln, indem sie an eigenen Projekten mitwirken, oder indem über externe Quellen bereits erfahrene Stakeholder einbezogen werden [23] und ein entsprechender Erfahrungsaustausch mit ihnen statt findet. Falls bereits in anderen Teilen des Unternehmens erfahrene Mitarbeiter mit Bezug zu hybriden Entwicklungsmethoden vorhanden, empfiehlt sich ein direkter Austausch zwischen ihnen und den unerfahrenen Mitarbeitern.

Keine einheitliche Dokumentenstruktur

In der Literatur wurde keine konkrete Lösung genannt, jedoch liegt es auf der Hand, dass eine einheitliche Dokumentenstruktur festgelegt werden muss von den Verantwortlichen des jeweiligen Unternehmens. Dies beinhaltet zum einen, welche Dokumente als Pflichtdokumente im Laufe des Projekts angefertigt müssen. Welche genauen Dokumente damit gemeint sind, hängt vom jeweiligen Bereich ab. Beispielsweise muss ein Finanzdienstleistungsunternehmen, welches stetig amtlich geprüft wird, andere Dokumente vorlegen als beispielsweise eine Firma für Spieleentwicklung. Zum anderen muss festgelegt werden, welche Form und welchen Inhalt die Dokumente haben müssen, indem Vorlagen für jeden Dokumententypen angefertigt werden. Dadurch wird den Mitarbeitern eine Richtlinie gegeben, wie sie die Dokumente aufbauen müssen und zusätzlich kann man bei ähnlichen Projekten auf Dokumente zurückgreifen und spart sich zusätzlichen Arbeitsaufwand.

Mangel an Dokumenten

Ein Mangel an Dokumenten ist nicht immer unbedigt ein Konflikt, der auch behoben werden muss. Liegt der Fokus ausschließlich bei dem Endprodukt und die Dokumente werden nicht benötigt, um sie beispielsweise bei einer Prüfung vorzulegen, kann auf das anfertigen von bestimmten Dokumenten verzichtet werden [17]. Jedoch sollten Dokumente, um die Anforderungen entsprechend für die spätere Entwicklung festzuhalten, trotzdem angefertigt werden, damit Bezug auf sie genommen werden kann. Besonders User Stories sollten gründlich, einheitlich und ausführlich beschrieben werden, damit keine Missverständnisse entstehen [22].

Anpassungen von Anforderungen

Um eine bestmögliche Übersicht über Änderungen an Anforderungen zu erhalten, bieten sich verschiedene Projektmanagement Tools, in denen digital die User Stories abgelegt werden und zu jeder Anforderung eine Änderungshistorie gespeichert wird. Ein empfohlenes Werkzeug ist zum Beispiel JIRA [15]. Dort werden die Stories übersichtlich abgelegt und alle Änderungen sind direkt erkennbar. Verbunden mit einem Projektmanagement Tool ist natürlich zusätzlicher Aufwand bei der Ersteinrichtung und der Schulung der Mitarbeiter, jedoch wird auf lange Sicht besonders bei großen, herausfordernden Projekten dadurch an Zeit gespart [15].

Keine Beschreibung von Fehlerinteraktionen

Um zusätzlich zu der Masse an Anforderungen auch alle Interaktionen der Anwender, die einen Fehler verursachen könnte, abzudecken, eignet sich eine iterative Herangehensweise [22]. Die Masse an möglichen Werten ist zu groß, um diese innerhalb eines Durchlaufs alle festzulegen, weshalb man zu Beginn die grundlegenden Interaktionsmöglichkeiten festlegt und diese iterativ Stück für Stück abarbeitet [22]. Dadurch lässt sich eine größtmögliche Abdeckung erreichen, auch wenn dies mit einem größeren Zeitaufwand verbunden ist.

Ergänzend dazu bietet es sich an, bereits früh in der Anforderungsermittlung auf Prototypen zurückzugreifen [40]. Dadurch nimmt der theoretische Anteil des Prozesses ab, da man bereits einen ersten Entwurf vor Augen hat und dadurch zusätzliche Themen den Stakeholdern klar werden.

Kommunikation

Die Kommunikation ist generell ein schwieriges Thema, welches in der Literatur sehr viel Erwähnung findet [32] [40] [17]. Besonders die Kommunikation zwischen einzelnen Teams gestaltet sich schwierig. Daher muss man besonders bei der Anforderungserhebung so früh wie möglich auf Feedback der Stakeholder eingehen und einen iterativen Prozess einführen, damit so oft wie möglich Feedback gegeben werden kann. Dadurch minimiert sich der Anteil an Kommunikationsproblemen, da jeder in den Prozess regelmäßig miteinbezogen wird [32].

Ableiten von Tests

Da Anwendungstests anhand der im RE festgelegten Anforderungen entwickelten werden, müssen Anforderungen so gründlich wie möglich dokumentiert werden [17]. Ein einheitliches Template für eine Spezifikation erleichtert das Ableiten von Tests, da der selbe Prozess genutzt werden kann, der bereits bei vorherigen Projekten verwendet wurde. Daher muss nur noch auf die anwendungsspezifischen Testfälle eingegangen werden und spart somit Zeit bei der Erstellung der Tests [17].

Diskussion

Allgemein kann nicht festgelegt werden, ob ein hybrides Unternehmen bei Problemen mit unvollständigen Anforderungen eher in die Richtung des traditionellen oder des agilen RE gehen sollte, um diese Konflikte zu beheben, da es auf den speziellen Bereich des Unternehmens ankommt. Besonders bei kleineren Projekten, bei denen die Prozesse weniger stark aufeinander

aufbauen, kann eher eine Richtung in das agile RE empfohlen werden, da dadurch vorallem die Kommunikation durch iterative Prozesse verbessert wird. Bei Großprojekten hingegen, bei denen Bezug genommen werden muss auf vorher festgelegte Dokumente um ein weiteres Vorgehen zu gewährleisten, bieten sich traditionelle Methoden an, um auf vollständige und übersichtliche Dokumente zurückzugreifen.

5.1.2 Unklare Anforderungen

Missverständnisse zwischen den Stakeholdern

Um so wenig Missverständnisse wie möglich zu erreichen, müssen alle Anforderungen eindeutig für alle Beteiligten sein. Besonders hilft dabei die Form der User Stories, da die Anforderungen kurz und knapp in Alltagssprache festgelegt werden [32] [21]. Diese müssen entsprechen auf Vollständigkeit geprüft werden, zum Beispiel mit Hilfe von Checklisten [32]. Dadurch wird festgelegt, ab wann eine Anforderungen so beschrieben wurde, dass sie für den weiteren Verlauf verwendet werden kann und somit eine *Definition of Readiness* entschieden wird.

Gibt es inhaltliche Unklarheiten können Prototypen, in ditialer oder analoger Form, angefertigt werden, damit die Stakeholder bereits festgelegte Anforderungen in der Praxis sehen können, wodurch sich Änderungen an bestehenden oder neue Anforderungen bereits in einem frühen Stadium herausbilden [40].

Unklares User Story Format

Da nicht allen Stakeholdern im hybriden RE das User Story Format klar ist, muss es erst erlernt werden. Dabei hilft zum einen ein Austausch mit anderen Mitarbeitern, die bereits damit Erfahrungen gemacht haben, oder es werden Schulungen zum Anfertigen und Verstehen von User Stories besucht [24]. Zusätzlich muss sichergestellt werden, dass ein einheitliches Format der User Stories innerhalb aller Projekte verwendet wird. Dadurch entstehen keine Inkonsistenzen zwischen den Projekten und es muss nicht bei jedem neuen Projekt ein anderes Format allen Stakeholdern beigebracht werden [24].

Komplexität der Anforderungen

Mithilfe von User Stories lassen sich Anforderungen in einer übersichtlichen Form darstellen, wodurch komplexe Themen in einer einfachen Art und Weise beschrieben werden können. Jedoch lassen sich verschachtelte Anforderungen nur schwierig innerhalb einer User Story abspeichern, weshalb man User

Stories aufspalten muss [24]. Dabei kann eine User Story entweder auf mehrere Stories aufgeteilt werden und mittels Tools wie JIRA die Verknüpfung zwischen ihnen visualisieren, oder man unterteilt eine Story in einzelne Unteraufgaben, die bestimmte Bereiche und deren Umsetzung detaillierter beschreiben [24].

Anpassungen von Anforderungen

Damit Anpassungen an Anforderungen klar und deutlich für alle Beteiligten werden, muss dies nachverfolgbar sein, damit jede Änderung nachvollziehbar ist. Dafür eignen sich diverse Projektmanagement Tools, bei denen eine Änderungshistorie integriert ist. Dadurch wird klar, wann welche Änderungen von wem durchgeführt wurden und bei Unklarkeiten kann in einer späteren Iteration darauf zurückgegriffen werden [30].

Diskussion

Um Unklarheiten in Anforderungen zu vermeiden, eignen sich agile Methoden besser als traditionelle Methoden. Besonders durch die iterative Arbeitsweise und die Nutzung von User Stories fällt es auch unerfahreneren Stakeholdern leichter, Anforderungen zu verstehen und bei Unklarheiten kann direkt reagiert werden. Des weiteren werden digitale Projektmanagementwerkzeuge verwendet, um zum einen die User Stories digital abzulegen und zum anderen um eine Nachverfolgbarkeit zu gewährleisten, damit Anpassungen an Anforderungen deutlicher werden.

5.2 Priorisierung von Anforderungen

Mangelhafte Priorisierung

Eine gute und vollständige Priorisierung von Anforderungen ist nur möglich, wenn eine entsprechend vollständige Grundlage zur Verfügung steht, zum Beispiel in Form einer Spezifikation. Sind dort alle Anforderungen aufgeführt, werden diese auch bei der Priorisierung berücksichtigt. Besonders durch iteratives, kontinuerliches Planen kann eine hohe Abdeckung an notwendigen Anforderungen ermitteln werden [17]. Durch diese ist es auch im weiteren Verlauf einfacher, fehlende Priorisierungen nachzuholen.

Inkorrekte Priorisierung

Um eine Priorisierung so genau wie möglich festzulegen, werden in der Literatur mehrere konkrete Methoden genannt. Babar et al. [7] schlägt eine

Methode namens *Expert Driven Fuzzy Logic based Prioritization* vor, bei dem der Priorisierungsprozess über drei verschiedene Ebenen abläuft. Ebene 1 beschreibt die Priorisierung der Stakeholder als manuellen Schritt. Anhand der in Ebene 1 festgelegten Priorisierung wird in Ebene 2 jeder Anforderungen ein Wert zugewiesen, der sich aus der Priorisierung der Anforderung selbst und dem Bereich berechnen lässt, von dem der Stakeholder kommt, der die Anforderung festgelegt hat. In der dritten Ebene werden die Daten in ein neuronales Netz ein Eingabewert gegeben, welches mittels Training aus früheren Priorisierungsprozessen eine endgültige Entscheidung über die Wichtigkeit der Anforderungen bestimmt [7].

Nach Inayat et al. [22] sind Prototypen eine gute Unterstützung, um eine korrekte Priorisierung festzulegen, da man einen praktischen Bezug zu den Anforderungen entwickelt und so in einem frühen Stadium besser differenzieren kann zwischen essentiellen Anforderungen und weniger wichtigen Anforderungen. Dies wird kombiniert mit einer *Value based prioritization*, bei der jeder Anforderung ein Wert zugewiesen wird, der sich der Priorisierung und dem damit verbundenen Aufwand ergibt [22] [37].

Priorisierung & Aufwandsschätzung von User Stories

Die Schätzung von Aufwänden von User Stories liegt keine Berechnung zugrunde, sondern wird mittels Erfahrungswerten ermitteln. Ist dies nicht das erste Projekt, in dem mit User Stories gearbeitet wird, kann man auf ein vorheriges Projekt zurückgreifen, um Vergleiche ziehen zu können und die Aufwände anhand des Projekts ähnlich abschätzen kann. Ist dies das erste Projekt mit User Stories, kann auch auf ein Projekt zurückgegriffen werden, welches ohne Stories die Anforderungen beschreibt [41]. Anhand von von ähnlichen Aufwänden können nur die Stories angefangen geschätzt zu werden. Um eine genauere Schätzung zu erhalten, ist es ratsam, besonders bei unerfahrenen Stakeholdern den Prozess iterativ abzuarbeiten, indem mehrere Durchgänge der Schätzung durchgeführt werden. Da bei der Schätzung selbst ein Erfahrungsgewinn stattfindet, kann dieser auf die nächste Iteration angewendet werden.

Re-Priorisierung

Re-Priorisierung lässt sich umsetzen, indem mehr agile Maßnahmen benutzt werden als traditionelle [40]. Durch regelmäßige Iterationen in Form von Sprints, kann die Priorisierung entsprechend angepasst werden, falls Stakeholder einen Fehler in der im letzten Sprint festgelegten Priorisierung finden. Dadurch kann im nächsten Sprint die Änderung direkt umgesetzt werden

und hat nur wenig Einfluss auf den weiteren Verlauf [40].

Diskussion

Die Priorisierung ist eine Stärke von agilen Entwicklungsansätzen. Durch die iterative Arbeitsweise entsteht eine hohe Abdeckung an ermittelten Anforderungen, die priorisiert werden können. Dazu ist es ohne viel Aufwand möglich, Anpassungen an der Priorisierung in einer späteren Iteration vorzunehmen, ohne dass der weitere Ablauf darunter leidet. Die strikte Planung von traditionellen Methoden hingegen ist anfällig für Änderungen und die einzelnen Phasen des RE müssen daher gut strukturiert sein, damit keine Fehler auftreten.

5.3 Verwaltung des Backlogs

Mangelnde Erfahrung

Die Nutzung von einem Backlog ist eine Methode, die von den agilen Entwicklungsmethoden übernommen wurde. Die Erfahrung in der Anwendung des Backlogs muss daher mit Praxiserfahrung in konkreten Anwendungsfällen gesammelt werden. Unterstützend dazu wird die Hilfe von Experten im Umgang mit einem Backlog herangezogen, die den Stakeholdern mit dem Umgang unterstützen [24]. Dafür muss jedoch entsprechendes Budget in Form von Kosten und Zeit investiert werden.

Zu schnelle Transformation

Um eine möglichst reibungslose Transformation zu gewährleisten, benötigt es Expertenwissen im Umgang mit neuen Methoden, sowie entsprechende Einweisungen der Stakeholder, da sonst ein Projekt scheitern könnte [24]. Bekommt ein Unternehmen einen großen Auftrag zu Beginn einer Umstellung ist es ratsam, auf altbewährte Methoden vorerst zurückzugreifen, da sonst der Erfolg des Projekts gefährdet ist. Daher ist eine Umstellung ein langwieriger Prozess und sollte nicht überstürzt werden und lieber gut durchdacht werden, welche Methoden angepasst werden und bei welchen Methoden man auf den bisherigen Standard beruht.

Inhalt & Aktualität der User Stories

Durch eine iterative Arbeitsweise lässt sich ein vollständiger Inhalt der User Stories gewährleisten und zusätzlich bleiben sie immer auf dem neuesten

Stand [24]. Durch regelmäßige Treffen werden Lücken in Anforderungen früh entdeckt und können somit direkt angepasst werden, um immer auf einem aktuellen Stand zu sein. Zusätzlich lassen sich durch regelmäßige Treffen alle Stakeholder in den Prozess miteinbeziehen, wodurch die Anforderungen von jedem berücksichtigt werden.

Diskussion

Da der Backlog eine Methodik aus der agilen Entwicklung ist, lassen sich die Problematiken damit auch mit agilen Methoden beheben. Besonders eine iterative Vorgehensweise bietet sich im Zusammenhang mit der Nutzung eines Backlogs an. Traditionelle Methoden helfen zwar nicht mit dem direkten Umgang eines Backlogs, jedoch ist es bei zu großen Komplikationen ratsam, wieder in Richtung der traditionellen Methoden des Festhaltens von Anforderungen zu gehen, damit ein Scheitern eines kritischen Projekts vermieden werden kann.

5.4 Konflikte mit Stakeholdern

Mangelnder Anforderungsaustausch

Um den Austausch über die Anforderungen zwischen den Stakeholdern zu fördern, muss an der Kommunikation gearbeitet werden. Dies bedeutet, dass nicht nur die Kommunikation innerhalb eines Teams gefördert werden muss, sondern auch die Kommunikation zwischen Teams, damit möglichst viele Personen direkt am Prozess beteiligt werden [24]. Dazu bietet es sich an, einen On-Site Customer festzulegen, also ein Ansprechpartner, der bei Fragen vor Ort ist und direkt angesprochen werden kann. Dadurch muss nicht auf die nächste Besprechung gewartet werden bei Problemen, sondern kann dies direkt versuchen zu lösen [16].

Unklare Verantwortlichkeiten

Verantwortlichkeiten müssen festgelegt werden, damit klare Grenzen zwischen Aufgabenbereichen gezogen werden können. Dabei sollten Stakeholder auf die Erfahrung der Beteiligten aus zuvor abgeschlossenen Projekten vertrauen [40]. Beispielsweise sollten Entwickler, die in vorherigen Projekten an der Systemarchitektur gearbeitet haben, auch weiterhin an der Systemarchitektur arbeiten, da sie auf Erfahrungswerte im Umgang damit zurückgreifen können.

Mangelnde Unterstützung der Projektleitung

Der Projektleiter ist für die Koordination des Projekts zuständig. Jedoch muss er dies nicht alleine bewältigen, sondern kann sich auf erfahrene Mitarbeiter beziehen [40]. Besonders auf die Erfahrung der Entwickler sollte mehr eingegangen werden, da diese die Erfahrungen in der Praxis erlebt haben. Des weiteren hilft ein On-Site Customer, um die Stakeholder direkt unterstützen zu können [16].

Diskussion

Um Konflikte mit Stakeholdern auszulösen, hilft nur die Kommunikation. Ist man bereit, die zusätzliche Zeit in Anspruch zu nehmen, um ein klares Rollengerüst festzulegen, sollte man diesen Weg beibehalten. Ist man jedoch nicht bereit, muss zu der alten Projektstruktur zurückgekehrt werden. Jedoch bringen Investitionen wie ein On-Site Customer einen deutlichen Mehrwert für den weiteren Verlauf, der sich vor allem in der Qualität des Endprodukts wiederspiegelt.

5.5 Kommunikations- & Koordinationsprobleme

Koordinationsprobleme

Mithilfe des agilen Rollenkonzepts wird die Verantwortlichkeit im Vergleich zu traditionellen Methoden besser verteilt, da zusätzliche Rollen wie Scrum Master oder Product Owner festgelegt werden. Dazu kommt, dass durch agile Methoden das Team in mehrere kleine Teams aufgeteilt werden [32]. Dadurch wird die Koordination eines großen Teams auf die Koordination von mehreren kleinen Teams aufgeteilt, die zusätzlich durch eigene Führung in den jeweiligen Teams koordiniert werden. Dies schafft einen besseren Überblick über das große Ganze des Projekts, da die Verantwortung besser aufgeteilt ist [32].

Informationsaustausch zwischen Stakeholdern

Damit wichtige Informationen an Mitglieder von allen Teams weitergeleitet werden, muss zu Beginn des Projekts ein Informationsfluss festgelegt werden und im weiteren Verlauf des Projekts gemanaged werden, damit die Informationen immer an das bestimmte Ziel gelangen können [24]. Des

weiteren steigern Praktiken wie ein On-Site Customer und die Face-To-Face Kommunikation die Menge an gewonnen Informationen [22].

Aufteilung der Teams

Zwar sind hybride Methoden damit verbunden, eine neue Struktur der Teams festzulegen, jedoch sind damit auch viele Methoden verbunden, die die Kommunikation innerhalb eines Teams verbessern. Durch viele Iterationen kommt es auch zu mehr Kontakt zwischen den Mitarbeitern, kleinere Gruppen fördern den Zusammenhalt des Teams und Praktiken wie Face-To-Face Kommunikation erleichtern den Umgang mit einem neuen Team [22] [9] [24].

Mangelnder Austausch bezüglich Anforderungen

Um Anforderungen in einer hybriden Umgebung bestmöglich festzulegen, schlägt Bick et al. [9] ein Kommunikationsmodell vor, das sowohl die Kommunikation innerhalb, als auch die Kommunikation zwischen den Teams berücksichtigt. Dabei wird die Anforderungsbestimmung, die Priorisierung, die Schätzung und die spätere Zuweisung an Teams berücksichtigt und welche Aktivitäten einzelne Teams und welche das gesamte Projekt betreffen.

Ressourcenmangel

Kommunikation zwischen Mitarbeitern ist ein Prozess, der Zeit in Anspruch nimmt, besonders bei neu eingesetzten Methodiken. Dies muss in der vorher durchgeführten Planung berücksichtigt werden, damit der Zeitplan nicht abgeändert werden muss. Ist diese zusätzliche Zeit nicht vorhanden, ist es nicht sinnvoll, neue Methoden im RE einzuführen.

Diskussion

Die Kommunikation ist eine ganz klare Stärke des agilen RE. Durch die iterative Arbeitsweise findet mehr Kommunikation zwischen den Mitarbeitern statt und es wird eher auf Feedback reagiert und kann somit auch früher umgesetzt werden. Durch das Aufsplitten in kleinere Teams wird zusätzlich die Arbeitszufriedenheit erhöht, da eine größere Verbindung zu den direkten Mitarbeitern besteht. Eine Anpassung in Richtung des traditionellen RE würde eher einen Rückschritt bedeuten, da in weniger Kommunikation zwischen den Mitarbeitern bedeutet.

5.6 Nicht-funktionelle Anforderungen

Ignorieren von NFRs

Um nicht-funktionelle Anforderungen besser einzubeziehen, sollten traditionelle Methoden bei der Planung von Anforderungen durchgeführt werden [40]. Dadurch, dass die komplette Projektplanung vor der Durchführung des Projekts abgeschlossen wird, wird dadurch auch ein Großteil der NFRs berücksichtigt. Zusätzlich wird durch die ausführliche Dokumentation bei traditionellen Methoden sichergestellt, dass genügend Hintergrund zu den NFRs festgelegt werden, wodurch sich die spätere Umsetzung erleichtern lässt [8].

Mangelndes NFR Verständnis

Damit NFRs nicht nur festgehalten werden, sondern auch so beschrieben werden, dass sie in der Entwicklungsphase so wie geplant umgesetzt werden können, müssen Entwickler mit in den Prozess der Anforderungserhebung einbezogen werden [15] [40]. Wenn nur Theoretiker das Grundgerüst festlegen und selbst nicht genau wissen, ob dies überhaupt umsetzbar ist, hilft es bei der Entwicklung nicht weiter, da entweder eine suboptimale Umsetzung entwickelt wird, oder es müssen die Anforderungen an die Umsetzung angepasst werden, was auch nicht optimal ist [15].

Softwarearchitektur

Softwarearchitekten müssen genau so in den RE Prozess einbezogen werden, wie auch Entwickler. Die Softwarearchitekten sind nur von höherer Wichtigkeit, da sie das Grundgerüst bereits in einem sehr frühen Stadium der Entwicklung festlegen müssen und daher nicht die Zeit miteingeplant ist, Anforderungen neu zu entwickeln [40].

Diskussion

Nicht-funktionelle Anforderungen sind eine Schwachstelle von agilen Entwicklungsmethoden und es eignet sich deutlich besser, traditionelle Methodiken anzuwenden, um NFRs bestmöglich festzulegen. Durch die gründliche Planung eines Projekts bevor überhaupt die Entwicklung startet, können alle technischen Voraussetzungen bestimmt werden und müssen nicht im weiteren Verlauf angepasst werden.

5.7 Kosten- & Zeitermittlung

Schwierige Budgetplanung

Durch viele Schwankungen im Projektablauf mit hybriden Methoden, ist es schwierig, ein komplettes Projekt genau zu planen [34]. Daher bietet es sich an, das Projekt in kleinere Abschnitte aufzuteilen und jedem Abschnitt ein eigenes Budget zuzuweisen [32]. Dadurch gestaltet sich die Budgetplanung einfacher und Änderungen im Plan in einzelnen Bereichen lassen sich besser berücksichtigen [32].

Kostensteigerung

Eine Kostensteigerung lässt sich vermeiden, indem das Projekt so genau wie möglich durchgeplant wurde. Besonders der Aufwand der User Stories muss dem Aufwand der späteren Umsetzung so genau wie möglich entsprechen, da die Kosten entsprechend anhand der Story Points berechnet werden [15].

Mangelndes Zeitmanagement

Das Zeitmanagement wird anhand des Aufwands und der Priorisierung bestimmt [32]. Damit ein möglichst genauer Zeitplan entworfen werden kann, muss die Aufwandsschätzung und Priorisierung so genau wie möglich erfolgen und somit muss mehr Zeit in dies investiert werden. Werden nur vage Werte angegeben, kann auch kein genauer Zeitplan der Entwicklung festgelegt werden [32].

Diskussion

Sowohl agile als auch traditionelle Methoden eignen sich für eine bessere Budgetplanung. Welche genaue Methodik hängt von dem Projekt selbst ab. Wenn es möglich ist, das Budget in einzelnen Etappen zu bestimmen, eignen sich agile Methoden, da jeder Sprint für sich im Budget betrachtet werden kann und so ein exakteres Ergebnis bekommt. Muss jedoch das gesamte Budget zum Projektbeginn beauftragt werden, eignen sich traditionelle Methoden besser, da dort der vollständige Projektplan im RE selbst festgelegt werden kann mit dem entsprechenden Projektbudget.

5.8 Ungleichmäßige Erfahrungsverteilung

Bei einer Lösung für Probleme bei Erfahrungswerten ist keine direkte Unterteilung notwenig, wie bei den Konflikten selbst. Erfahrungen werden entweder

gesammelt, indem selbst Praktiken durchgeführt und bei der Durchführung selbst ein besseres Verständnis entwickelt, oder indem man von bereits erfahrenen Personen diese Erfahrungswerte weitergegeben bekommt [14]. Besonders bei komplett neuen Methodiken, die vorher keine Anwendung bei den Mitarbeitern gefunden habe, muss Zeit und auch Geld investiert werden, damit diese Praktiken erlernt werden können, um sie später selbst anwenden zu können. Das Einkaufen von Experten wird oft genutzt, damit diese anhand von Projekten im Unternehmen ihr Wissen an andere Mitarbeiter weitergeben könnnen [31].

Diskussion

Hier kann keine optimale Aussage getätigt werden, ob ein Weg in Richtung traditionell oder agil der bessere wäre. Ist man in der Lage, das zusätzliche Budget für Experten in Kauf zu nehmen, sollte dieses Budget investiert werden, da es sich zukünftig rentiert, neue Methoden zu erlernen. Ist man jedoch nicht in der Lage, dieses Budget bereitzustellen, sollte bei den zuvor genutzten Methoden geblieben werden, um ein Scheitern eines Projekts abzuwenden.

5.9 Verifikation/Validierung

Testabdeckung

Eine möglichst vollständige Testabdeckung kann gewährleistet werden, wenn die Anforderungen gründlich definiert wurden, sodass es möglich ist, direkte Testfälle aus ihnen abzuleiten [10]. Des weiteren können Anforderungen in einer Spezifikation so festgelegt werden, dass sie direkt in Form von Testfällen geschrieben werden, wodurch sie den Zweck von Anforderungen und Testfall mit einem Mal erfüllen [10].

Fehlender Verifikationsprozess

Mit einem Verifikationsprozess werden Tests mit den dazugehörigen Anforderungen in Verbindung gebracht. Wie genau diese Verifikation abläuft, muss in der Planungsphase festgelegt werden [10]. Beispielsweise kann ein Testplan angelegt werden, in der die Tests festgelegt werden, um diesen im weiteren Verlauf den Stakeholdern vorzustellen, damit eine Abnahme der umgesetzten Anforderungen gewährleistet werden kann.

Verifikation von NFRs

Um eine Verifikation von NFRs überhaupt zu ermöglichen, müssen diese bei der Anforderungserhebung beachtet werden und dokumentiert werden, gründlich genug, sodass sich Tests ableiten lassen. Anschließend lassen sich Entwicklertests herleiten, die von den Entwicklern durchgeführt werden und die NFRs auf die Umsetzung prüfen, wie sie dokumentiert wurden. Diese Tests werden dokumentiert und den Stakeholdern vorgestellt, sodass diese prüfen können, ob die Umsetzung abgenommen werden kann [10].

Diskussion

Verifikation und Validierung sind Prozesse, die bei den agilen Methodiken wenig beachtet wird. Agile Methoden legen Wert darauf, dass die umgesetzte Software so viele Anforderungen wie möglich beinhaltet und geht weniger auf den Dokumentationsprozess ein. Falls keine ausführliche Prüfung einer Software benötigt wird, können eher agile Methoden verwendet werden. Wird jedoch eine ausführliche Abnahme benötigt, empfiehlt es sich, auf traditionelle Methoden zurückzugreifen, da dort eine gründliche Abnahme mit der entsprechenden Dokumentation sichergestellt wird.

Kapitel 6

Verwandte Arbeiten

Zum Thema des hybriden Requirements Engineerings gibt es eine Vielzahl an spannender Literatur, die in dieser Arbeit nicht erwähnt wurde. Besonders, da das hybride RE sowohl traditionelle, als auch agile Entwicklungsansätze beinhaltet, kann auf diese ebenfalls zurückgegriffen werden.

Eine spezielle Quelle ist *Naming the pain in requirements engineering* [35]. Dies ist eine globale Initiative, die zwei Mal jährlich auf den aktuellen Status des Requirements Engineerings eingeht und beschreibt in dem Zusammenhang Probleme, Ursachen und Auswirkungen auf die Praxis. Dabei werden nicht explizit hybride Methodiken betrachtet, sondern RE als Ganzes - traditionell, hybrid und agil.

Eine weitere Quelle ist *How are Hybrid Development Approaches Organized* von Prenner et al. [38]. Dieses Paper geht weniger auf Konflikte im hybriden RE ein, sondern geht dem Grundkonzept des hybriden RE auf den Grund und welche expliziten Methodiken und Herangehensweisen verwendet werden. Dort werden *The Waterfall-Agile-Approach*, *The Waterfall-Iterations-Approach* und *The Pipeline-Approach* beschrieben [38].

Im Zusammenhang mit einer systematischen Literaturrecherche sollte auch die Methode des *Snowballings* erwähnt werden nach Wohlin et al. [47]. Diese Methode wird bei einer SLR angewendet, falls das Ergebnis, nachdem die SLR abgeschlossen wurde, nicht ausreichend für die weitere Analyse ist. Dafür wird unterstützend das Forward Snowballing und Backward Snowballing angewendet, bei denen für eine weitere SLR Iteration zusätzlich zum einen die Paper hinzugezogen, die von den Resultaten zitiert werden und zum anderen die Paper, in denen ein Resultat zitiert wurde.

Die letzte Arbeit, die hier Erwähnung findet, ist *Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond* von Kuhrmann et al. [29]. Dort werden Resultate der HELENA (Hybrid dEveLopmENt Approaches) vorgestellt. Dies beinhaltet verschiedene Praktiken im

Zusammenhang mit hybriden Entwicklungsansätzen, die sich jedoch nicht nur auf das RE beschränken, sondern auf den vollständigen Projektablauf.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Zusammenfassend wurde in dieser Arbeit ein Überblick über eine Vielzahl an Konflikten gegeben, die im Zusammenhang mit dem hybriden Requirements Engineering stehen. Die Ursachen dieser Konflikte beruhen sowohl auf traditionelle Methoden, als auch auf agile Methoden, sowie einer Kombination aus den beiden. Aber auch die Lösungen dieser Konflikte beruhen sowohl auf traditionelle Methoden als auch auf agile Methoden, je nach speziellem Anwendungsfall.

Eine große Ursache für Konflikte ist die mangelnde Kommunikation, beispielsweise bei der Bestimmung von Anforderungen. Eine gut durchdachtes Kommunikationssystem bildet die Grundlage für einen erfolgreichen Projektablauf und sorgt für eine bessere Arbeitsmoral.

Alles in allem sind traditionelle, hybride und agile Entwicklungsansätze für das Requirements Engineering wichtige Ansätze und alle müssen in Betracht gezogen werden, da jede Methodik ihre Vor- und Nachteile in bestimmten Bereichen hat. Welche Methodik genau angewendet werden soll, hängt daher von dem genauen Anwendungsfall und der Strategie eines Unternehmens ab.

7.2 Ausblick

Zukünftig werden immer mehr Unternehmen den Schritt wagen, traditionelle Methoden abzulösen und in Richtung des hybriden und agilen Requirements Engineerings zu gehen. Da eine Umstellung mit Komplikationen verbunden ist, muss im Vorfeld genau betrachtet werden, welche Konflikte auftreten können. In dieser Arbeit wurden nur die größten Konflikte analysiert, jedoch

gibt es noch viele weitere, kleine Konflikte, die in der Masse auch zu größeren Problemen werden können. Diese gilt es weiterhin zu analysieren und entsprechende Lösungen zu den Konflikten zu entwickeln, da noch nicht einmal eine Lösung zu allen größeren Konflikten ermittelt werden konnte.

Zusätzlich kann es durch die größere Menge an Unternehmen, die ihre Methodiken abändern, zu neuen Konflikten kommen, die aktuell noch gar nicht betrachtet werden. Dies bedarf ebenfalls weiterer Forschung, um diese zu analysieren und einen bestmöglichen Umgang mit ihnen zu entwickeln, damit ein Unternehmen, je nach speziellem Anwendungsfall, die für sich passenden Methodiken auswählen kann, ohne dabei auf unlösbare Probleme zu stoßen.

Literaturverzeichnis

[1] A. Alhazmi and S. Huang. Survey on differences of requirements engineering for traditional and agile development processes. In *2020 SoutheastCon*, pages 1–9, 2020.

[2] W. Alsaqaf, M. Daneva, and R. Wieringa. Agile quality requirements engineering challenges: First results from a case study. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 454–459, 2017.

[3] W. Alsaqaf, M. Daneva, and R. Wieringa. Quality requirements in large-scale distributed agile projects – a systematic literature review. volume 10153, pages 219–234, 02 2017.

[4] W. Alsaqaf, M. Daneva, and R. Wieringa. Understanding challenging situations in agile quality requirements engineering and their solution strategies: Insights from a case study. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 274–285, 2018.

[5] P. Anitha, D. Savio, and V. S. Mani. Managing requirements volatility while "scrumming" within the v-model. In *2013 3rd International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 17–23, 2013.

[6] L. B, K. Wiegers, and C. Ebert. The top risk of requirements engineering. *IEEE Software*, 18(6):62–63, 2001.

[7] M. I. Babar, M. Ramzan, and S. A. K. Ghayyur. Challenges and future trends in software requirements prioritization. In *International Conference on Computer Networks and Information Technology*, pages 319–324, 2011.

[8] W. Behutiye, P. Karhapää, D. Costal, M. Oivo, and X. Franch. Non-functional requirements documentation in agile software development: Challenges and solution proposal. In M. Felderer, D. Méndez Fernández,

B. Turhan, M. Kalinowski, F. Sarro, and D. Winkler, editors, *Product-Focused Software Process Improvement*, pages 515–522, Cham, 2017. Springer International Publishing.

[9] S. Bick, K. Spohrer, R. Hoda, A. Scheerer, and A. Heinzl. Coordination challenges in large-scale software development: A case study of planning misalignment in hybrid settings. *IEEE Transactions on Software Engineering*, 44(10):932–950, 2018.

[10] E. Bjarnason, P. Runeson, M. Borg, M. Unterkalmsteiner, E. Engström, B. Regnell, G. Sabaliauskaite, A. Loconsole, T. Gorschek, and R. Feldt. Challenges and practices in aligning requirements with verification and validation: A case study of six companies. *Empirical Softw. Engg.*, 19(6):1809–1855, Dec. 2014.

[11] E. Bjarnason, K. Wnuk, and B. Regnell. Overscoping: Reasons and consequences — a case study on decision making in software product management. In *2010 Fourth International Workshop on Software Product Management*, pages 30–39, 2010.

[12] B. Boehm and R. Turner. Management challenges to implementing agile processes in traditional development organizations. *IEEE Software*, 22(5):30–39, 2005.

[13] P. Brereton, B. A. Kitchenham, D. Budgen, and M. K. Mark Turner. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, Volume 80, Issue 4:571–583, 2007.

[14] F. Chen, N. Power, J. J. Collins, and F. Ishikawa. Contemporary requirements challenges and issues: An empirical study in 11 organizations. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1592–1599, New York, NY, USA, 2019. Association for Computing Machinery.

[15] K. Elghariani and N. Kama. Review on agile requirements engineering challenges. In *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, pages 507–512, 2016.

[16] H. Elshandidy and S. Mazen. A survey of agile and traditional requirements engineering. *International Journal of Scientific & Engineering Research*, 4(9):473–482, Sept. 2013.

[17] F. Gomes De Oliveira Neto, J. Horkoff, E. Knauss, R. Kasauli, and G. Liebel. Challenges of aligning requirements engineering and system testing in large-scale agile: A multiple case study. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pages 315–322, 2017.

[18] V. Heikkilae, M. Paasivaara, and C. Lasssenius. Managing the requirements flow from strategy to release in large-scale agile development: a case study at ericsson. In *Empirical Software Engineering*.

[19] V. T. Heikkilä, D. Damian, C. Lassenius, and M. Paasivaara. A mapping study on requirements engineering in agile software development. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, pages 199–207, 2015.

[20] H. Hiisilä, M. Kauppinen, and S. Kujala. Challenges of the customer organization's requirements engineering process in the outsourced environment - a case study. In *21st International Working Conference (REFSQ), Essen, Germany, March 23-26, 2015*, Lecture Notes in Computer Science, pages 214–229. Springer, 2015. VK: Kauppinen, M.

[21] T. Imani. Does a hybrid approach of agile and plan-driven methods work better for it system development projects? *International Journal of Engineering Research and Applications*, 07:39–46, 03 2017.

[22] I. Inayat, L. Moraes, M. Daneva, and S. S. Salim. A reflection on agile requirements engineering: Solutions brought and challenges posed. In *Scientific Workshop Proceedings of the XP2015*, XP '15 workshops, New York, NY, USA, 2015. Association for Computing Machinery.

[23] M. Kalinowski, R. Spínola, T. Conte, R. Prikladnicki, D. Méndez Fernández, and S. Wagner. Towards building knowledge on causes of critical requirements engineering problems. 07 2015.

[24] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar, and B. Kanagwa. Requirements engineering challenges in large-scale agile system development. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 352–361, 2017.

[25] H. Kitapci and B. W. Boehm. Using a hybrid method for formalizing informal stakeholder requirements inputs. In *Fourth International Workshop on Comparative Evaluation in Requirements Engineering (CERE'06 - RE'06 Workshop)*, pages 48–59, 2006.

[26] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. In *Information and Software Technology*, 2007.

[27] J. Klünder, R. Hebig, P. Tell, M. Kuhrmann, J. Nakatumba-Nabende, R. Heldal, S. Krusche, M. Fazal-Baqaie, M. Felderer, M. F. Genero Bocco, S. Küpper, S. A. Licorish, G. Lopez, F. McCaffery, Özcan Top, C. R. Prause, R. Prikladnicki, E. Tüzün, D. Pfahl, K. Schneider, and S. G. MacDonell. Catching up with method and process practice: An industry-informed baseline for researchers. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 255–264, 2019.

[28] E. Knauss, G. Liebel, J. Horkoff, R. Wohlrab, R. Kasauli, F. Lange, and P. Gildert. T-reqs: Tool support for managing requirements in large-scale agile system development. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 502–503, 2018.

[29] M. Kuhrmann, P. Diebold, J. Münch, P. Tell, V. Garousi, M. Felderer, K. Trektere, F. McCaffery, O. Linssen, E. Hanser, and C. R. Prause. Hybrid software and system development in practice: Waterfall, scrum, and beyond. In *Proceedings of the 2017 International Conference on Software and System Process*, ICSSP 2017, page 30–39, New York, NY, USA, 2017. Association for Computing Machinery.

[30] M. Kumar, M. Shukla, and S. Agarwal. A hybrid approach of requirement engineering in agile software development. In *2013 International Conference on Machine Intelligence and Research Advancement*, pages 515–519, 2013.

[31] G. Liebel, M. Tichy, E. Knauss, and O. Ljungkrantz. Organisation and communication problems in automotive requirements engineering. In *Requirements Engineering*, volume 23, pages 145–167, 2018.

[32] P. Mafra, M. Kalinowski, D. Méndez Fernández, M. Felderer, and S. Wagner. Towards guidelines for preventing critical requirements engineering problems. In *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 25–29, 2016.

[33] S. Masood Butt and W. F. W. Ahmad. Handling requirements using flexreq model. In *2012 IEEE International Conference on Computer Science and Automation Engineering*, pages 661–664, 2012.

[34] D. Mougouei. Factoring requirement dependencies in software requirement selection using graphs and integer programming. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ASE 2016, page 884–887, New York, NY, USA, 2016. Association for Computing Machinery.

[35] D. Méndez Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetro, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayebi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, and R. Wieringa. Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. *Empirical Software Engineering*, 08 2016.

[36] F. Paetsch, A. Eberlein, and F. Maurer. Requirements engineering and agile software development. In *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* IEEE Comput. Soc.

[37] F. Paetsch, A. Eberlein, and F. Maurer. Requirements engineering and agile software development. In *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, pages 308–313, 2003.

[38] N. Prenner, C. Unger-Windeler, and K. Schneider. How are hybrid development approaches organized? a systematic literature review. In *Proceedings of the International Conference on Software and System Processes*, ICSSP '20, page 145–154, New York, NY, USA, 2020. Association for Computing Machinery.

[39] N. Prenner, C. Unger-Windeler, and K. Schneider. Goals and challenges in hybrid software development approaches. In *Proceedings of the International Conference on Software and System Processes*, ICSSP '20, page 145–154, New York, NY, USA, 2021. Association for Computing Machinery.

[40] B. Ramesh, L. Cao, and R. Baskerville. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20(5):449–480, 2010.

[41] A. Safwat and M. Senousy. Addressing challenges of ultra large scale system on requirements engineering. *Procedia Computer Science*, 65:442–449, 2015. International Conference on Communications, management, and Information technology (ICCMIT'2015).

[42] J. Savolainen, J. Kuusela, and A. Vilavaara. Transition to agile development - rediscovery of important requirements engineering practices. In *2010 18th IEEE International Requirements Engineering Conference*, pages 289–294, 2010.

[43] T. Sedano, P. Ralph, and C. Péraire. The product backlog. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 200–211, 2019.

[44] I. Sommerville and P. Sawyer. *Annals of Software Engineering*, 3:101–130, 1997.

[45] M. K. Stefan Wagner, Daniel Méndez Fernández and M. Felderer. Agile requirements engineering in practice: Status quo and critical problems. *Vol. 21 No. 1 (2018): April 2018 Issue including selected works from CIbSE 2017 and LACLO 2016*, 2018.

[46] M. Tukur, S. Umar, D. Budgen, and J. Hassine. Requirement engineering challenges: A systematic mapping study on the academic and the industrial perspective. *Arabian Journal for Science and Engineering*, Volume 46, Issue 4:3723–3748, 2021.

[47] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, New York, NY, USA, 2014. Association for Computing Machinery.