Freie Universität Berlin

Bachelorarbeit am Institut für Informatik der Freien Universität Berlin

Arbeitsgruppe Software Engineering

Probleme im Requirements Engineering in der Praxis

Charlene Raddatz

Matrikelnummer: 5189222

charlene.raddatz@fu-berlin.de

Betreuer/in: Prof. Dr. Lutz Prechelt

Eingereicht bei: Prof. Dr. Lutz Prechelt

Berlin, 3. September 2021

Zusammenfassung

Hintergrund Die Ergebnisse von NaPiRE[12] zeigen, dass Unternehmen auf der ganzen Welt trotz teilweise oder komplett agilem Vorgehen Schwierigkeiten mit Anforderungsänderungen während der Softwareentwicklung haben. Die agilen Prozesse sollten diesen aber vorbeugen. Es gilt, hier zu untersuchen, warum trotzdem Probleme mit Anforderungsänderungen auftreten.

Ziel Es soll ein Einblick in die reale Welt der Anforderungserhebung und der Probleme mit Anforderungsänderungen gewonnen werden. Im Zusammenhang dazu wird auch untersucht, inwiefern agile Methoden zur Vorbeugung von schwerwiegenden Folgen durch Änderungen in den Anforderungen angewandt werden.

Methoden Mit semistrukturierten, qualitativen Interviews werden Gesprächspartner in unterschiedlichen Unternehmen und Positionen befragt. Die Analyse der Daten erfolgt anhand der Grounded Theory Methodology, welche außerdem dazu verwendet wird, die Interviews mit den bereits erhaltenen Kenntnissen anzupassen, um aus den darauffolgenden Interviews für die Erkenntnisse relevante Daten zu erheben.

Ergebnisse Ein Großteil der befragten Teilnehmer sagte aus, dass agile Methoden verwendet würden und nach Möglichkeit agil gearbeitet werde. Allerdings traten vor allem Richtungsänderungen mit stärkeren Folgen immer noch häufig auf. Eine Ursache dahinter ist die unvollständige Anwendung agiler Methoden, die vermutlich einen Teil der Probleme hätten verhindern können. Jedoch ließen nicht alle der genannten Schwierigkeiten sich mit korrekt angewandten agilen Prozessen verhindern.

Schlussfolgerungen Durch agile Methoden könnte vielen der Schwierigkeiten mit Anforderungsänderungen während der Entwicklung vorgebeugt werden. Dafür müssen diese aber korrekt und strikt durchgeführt werden. Die Akzeptanz der anwendenden Teams, sowie des Managements und der Kunden sollte angestrebt werden.

Abstract

Background The survey results from NaPiRE[12] show that companies still struggle with changing requirements despite using agile methods or working completely agile. The agile methods should prevent teams from experiencing issues when faced with sudden changes in requirements.

Goal This research provides insight into the problems arising in requirements engineering in real companies. Additionally, the use of agile methods is discussed because of their ability to prevent most of the consequences of sudden requirement changes. This paper aims to seek the underlying causes for this conflict.

Methods Qualitative, semi-structured interviews were used to study the obstacles in requirements engineering with which the surveyed teams were faced. Data was analysed with the grounded theory methodology, which was also used to update the interview questions to adjust the following interviews to results which were already found.

Result Most of the questioned participants said that their teams used at least some agile methods. Despite that, requirement changes, especially moving targets, commonly caused problems for the project and teams. One reason for this is the incomplete implementation of agile methods because some of the obstacles could have been prevented with strictly agile methods. Nevertheless, not all stated problems are caused by incomplete implementation and have other explanations.

Inference Agile methods could prevent many problems caused by changing requirements. But they need to be used correctly and strictly. The acceptance by the team, the management as well as the clients should be targeted.

Eidesstattliche Erklärung

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

3. September 2021

Charlene Raddatz

Inhaltsverzeichnis

1 Einführung

Die Anforderungserhebung bildet einen der ersten Schritte einer erfolgreichen Projektabwicklung. Ob ein Unternehmen für die interne IT oder für Kunden entwickelt, eine Erhebung der Anforderungen und die Erfüllung dieser müssen in jedem Fall durchgeführt werden. Da Arbeitsstunden in der Informationstechnik meist die teuerste Ressource eines Projektes sind, sind Änderungen an den Anforderungen während der Entwicklung eine Gefahr für den Erfolg eines Projektes. Bei Ungnauigkeiten oder kompletten Richtungsänderungen müssen mehr Arbeitsstunden investiert und teilweise sogar bereits entwickelte Artefakte entsorgt werden.
Das Prinzip des iterativen oder agilen Vorgehens arbeitet bewusst und aktiv mit Anforderungsänderungen.[1] Von den durch das Projekt NaPiRE befragten Unternehmen gaben 75% an, zumindest teilweise agil (hybrid, rather agile, agile) zu arbeiten. [12] Allerdings gehören unvollständige Anforderungen (incomplete or hidden requirements), Zieländerungen (moving targets) und ungenaue Anforderungen (underspecified requirements) zu den Top 5 der genannten Probleme in den Umfragen[12], obwohl diese durch das agile Vorgehen umgangen werden sollten. Hier stellt sich die Frage, warum es trotzdem zu diesen Problemen kommt.

Aufgrund dieser Fragestellung entstand die folgende Hypothese, welche in der vorliegenden Arbeit näher untersucht werden soll:

> Auftretende Probleme mit Anforderungen der drei Kategorien *Unvollständige Anforderungen (incomplete or hidden requirements), Zieländerungen (moving targets), ungenaue Anforderungen (underspecified requirements)* ließen sich mit agilen Methoden der Softwareentwicklung umgehen. Somit liegt eine Ursache für diese Probleme darin, dass nicht oder unvollständig agil gearbeitet wird.

Dies ist keine neue Frage in der Forschung. Ziel ist es, mit der vorliegenden Arbeit einen weiteren Aspekt des Problems zu betrachten.
Diese Bachelorarbeit gewährt einen Einblick in die reale Situation in Unternehmen bezüglich der Anforderungserhebung. Es werden Probleme in der Anforderungserhebung beleuchtet und Ursachen für diese vorgestellt. Die Gemeinsamkeiten und Unterschiede der befragten Unternehmen werden miteinbezogen, um so Aussagen über die Ursprünge der Schwierigkeiten zu treffen. Ziel ist es, die genannte Hypothese zu beleuchten und einen Eindruck zu gewinnen, ob und in welchem Maß diese bei den befragten Unternehmen zutrifft. Außerdem gewährt die Arbeit einen Einblick in die Arbeitsweisen realer Unternehmen und die Vorgehensweisen in den Teams. Unter anderem wird auch gezeigt, wie Agilität in der Realität umgesetzt wird.

1.1 Grenzen der vorliegenden Arbeit

In dieser Arbeit werden die Ergebnisse von Interviews mit unterschiedlichen Gesprächspartnern aus den befragten Unternehmen vorgestellt, die Vorgehensweise bei der Datenerhebung und der Analyse beschrieben, sowie die gefundenen Artefakte er-

läutert. Die Unterschiede und Gemeinsamkeiten der Unternehmen und Teams, sowie der beschriebenen Situationen werden beleuchtet.
Bei den Gesprächen handelt es sich um semistrukturiert geführte Interviews.
Nicht Teil der Arbeit ist eine empirische Erhebung von Daten in großem Umfang. Es handelt sich um eine durch qualitative Interviews erhobene Datenmenge, welche vor allem Ursachen für reale Beispiele der Probleme beleuchtet und stichprobenartig Einblick in Unternehmen gewährt.
Zusätzlich werden keine Handlungsempfehlungen oder Lösungsvorschläge für die gefundenen Probleme aufgezeigt. Es handelt sich lediglich um eine Nennung der Probleme und deren Ursachen, sowie die Herleitung der Verbindung dieser.

1.2 Vergleichbare Arbeiten

Um herauszufinden, wie sich diese Ausarbeitung in bereits veröffentlichte Arbeiten einbettet, wurde nach passender Literatur gesucht. Dafür führte eine Schlagwortsuche bei Google Scholar(https://scholar.google.com/) und im universitätsinternen Bibliotheksportal (https://fu-berlin.primo.exlibrisgroup.com/discovery/search?vid=49KOBV_FUB:FUB) auf Englisch und Deutsch zu Ergebnissen. Zu den gesuchten Schlagwörtern zählten zum Beispiel *requirements survey* und *Anforderungen interview*. Neben diesen wurde nach ähnlichen Formulierungen auf Deutsch und Englisch gesucht.

Zu Anforderungsmanagement existiert umfangreiche Literatur, deswegen beschränkte sich die Suche auf Veröffentlichungen, welche sich auch mit der Situation in realen Unternehmen befassten. Zusätzlich sollten die gesuchten Werke mit einer umfragen- oder interview-ähnlichen Erhebungsform erarbeitet worden sein. Nach dem Durchsehen einiger Arbeiten und deren Quellen zeigte sich, dass vor allem zwei Artikel interessant für diese Einordnung sind:

Kassab et al.[11] führten eine Umfrage durch, welche sich mit Projekteigenschaften, Vorgehensweisen, Unternehmen und Beteiligten des Anforderungsmanagements beschäftigte. Für die Analyse wurden mehrere Umfragen der letzten zehn Jahre verglichen und Veränderungen aufgezeigt. Diese Arbeit vergleicht Ergebnisse mehrerer Jahre miteinander und arbeitet die Unterschiede heraus. Der Schwerpunkt der hier vorliegenden Arbeit besteht in der Ergründung von Ursache-Wirkungs-Beziehungen von Herausforderungen mit Anforderungen. Es werden nur aktuell erhobene Daten verwendet.
Kassab et al.[11] arbeiteten mit einem standardisierten Fragebogen, welcher quantitative Daten der Antworten erfasst, wohingegen in der vorliegenden Arbeit auf Grundlage der Grounded Theory Methodology, welche in Kapitel 2 näher erläutert wird, in jedem Interview mit einem abgewandelten Fragenkatalog gearbeitet und qualitative Daten erhoben wurden.

Im Januar 2021 veröffentlichten Cristina Palomares et al.[14] einen Artikel, welcher eine interview-basierte Untersuchung der Anforderungserhebung in Unternehmen und

die Analyse der Ergebnisse beschrieb. Es wurden den Teilnehmern drei Fragen gestellt, welche die verwendeten Techniken, die beteiligten Rollen und die Herausforderungen während der Anforderungserhebung ermittelten. Die anschließende Analyse befasste sich mit den quantitativen Ergebnissen der Umfrage. Diese Arbeit setzt den Schwerpunkt auf die Anforderungserhebung, während meine Untersuchung auch das Anforderungsmanagement während laufender Projekte beinhaltete. Palomares et al. [14] führten die Befragung bei einer größeren Anzahl an Unternehmen durch, beschränkten sich dabei aber auf drei Fragen.
Diese sind auch in den hier verwendeten Interviews enthalten, aber der Schwerpunkt lag vor allem auf Beispielen, anhand derer die Ursache-Wirkungs-Beziehungen von den genannten Herausforderungen abzulesen sind und es wurde ein umfangreicherer Fragebogen verwendet. Somit liegt der Unterschied darin, dass Palomares et al.[14] das Vorkommen von Herausforderungen vor allem während der Erhebung ergründeten, während hier nach den Ursachen dieser Herausforderungen gesucht wurde. Ein weiterer Unterschied liegt darin, dass in der hier vorliegenden Arbeit eine kleinere Datenmenge analysiert wurde.

Um einen besseren Überblick zu geben, folgt eine Erläuterung des Aufbaus dieser Arbeit.
In Kapitel 2 werden zentrale Begriffe sowie Konzepte eingeführt und erklärt.
Der Abschnitt 3 beschreibt, warum die verwendete Vorgehensweise gewählt und welche Alternativen ausgeschlossen wurden.
Die eigentliche Durchführung der Befragungen wird in Kapitel 4 beschrieben.
Der Abschnitt 5 beschreibt das Vorgehen zur Analyse der Daten und welche Ergebnisse gefunden wurden.
Teil 6 der Arbeit enthält eine Zusammenfassung der Ergebnisse.
Abschließend zeigt Kapitel 7 einen Ausblick, was anschließend an diese Arbeit noch untersucht werden kann.

In der folgenden Arbeit werden die Gesprächspartner*innen immer *Gesprächspartner* genannt, da die Gruppe der Befragten ausschließlich aus männlichen Interviewpartnern bestand. Es wird deswegen für die bessere Lesbarkeit auf einen genderneutralen Begriff verzichtet.

2 Grundlagen

Um das Verständnis für die vorliegende Arbeit zu verbessern, werden in diesem Abschnitt wichtige Grundbegriffe und Konzepte erklärt. Grundsätzlich werden die Begriffe den meisten Lesern bekannt sein, doch die für das Verständnis benötigten Details sind hier noch einmal beschrieben.

2.1 Zentrale Begriffe

Interview Laut dem Duden handelt es sich bei einem Interview um eine „gezielte Befragung (von ausgewählten Personen) zu statistischen Zwecken“. [8] Allgemein ist das also ein Gespräch mit einer Person, in dem gezielt Fragen zu einem Sachverhalt

gestellt werden. Die Unterhaltung wird in manchen Fällen durch eine Aufnahme oder Mitschrift dokumentiert. Interviews lassen sich in *strukturiert*, *semistrukturiert* oder *unstrukturiert* sowie in *qualitativ* oder *quantitativ* einteilen.[7]

Anforderungen Eine Definition nach IEEE Std. 610.12-1990[10] beschreibt Anforderungen mit drei Kriterien:

1. „A condition or capability needed by a user to solve a problem or achieve an objective."
2. „A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents."
3. „A documented representation of a condition or capability as in (1) or (2)."

Anforderungen beschreiben also zum Beispiel zu erfüllende Bedingungen eines Systems, welche ein Nutzer benötigt, um ein Problem zu lösen. Diese bilden die Ziele, welche während der Entwicklung in einem Projekt erreicht werden sollen. Es gibt zwei Arten von Anforderungen. Funktionale Anforderungen beschreiben, was genau das Produkt können soll.[3] Dies sind zum Beispiel die Funktionen des Produktes, die ein Nutzer verwenden kann. Im Gegensatz dazu beschreiben nichtfunktionale Anforderungen Qualitätsanforderungen an das Produkt.[3] Ein Beispiel ist die Antwortgeschwindigkeit einer Software. Das ist keine Eigenschaft, die eine direkte Funktion für den Nutzer bietet, trotzdem aber für die korrekte Funktionalität entscheidend sein kann.

In dieser Arbeit sind Anforderungen zu implementierende Artefakte oder Forderungen an das System, welche durch die Beteiligten festgelegt werden. Diese können von internen oder externen Beteiligten sowie den Kunden direkt gefordert werden.
In der These wurden bereits drei Arten von Problemen mit Anforderungen vorgestellt. Sie werden hier noch einmal erklärt und voneinander abgegrenzt.

Unvollständige oder versteckte Anforderungen Von unvollständigen oder versteckten Anforderungen wird in dieser Ausarbeitung gesprochen, wenn im Laufe einer Entwicklung zu den bereits erhobenen neue Anforderungen hinzukommen, welche zuvor noch nicht bedacht wurden, oder weitere Teile einer Anforderung nicht betrachtet wurden und diese erst während der Bearbeitung auffallen. Dies kann zu einem Problem werden, wenn die neuen Anforderungen/Anforderungsteile anderen Anforderungen widersprechen oder Mehrarbeit im Projekt nötig machen.

Zieländerungen Zu Zieländerungen innerhalb eines Projektes kann es kommen, wenn sich die Anforderungen der Kunden stark ändern oder Veränderungen am Markt eine Richtungsänderung des Projektes nötig machen. Das Ziel der Entwicklung ändert sich, Anforderungen werden komplett neu erarbeitet oder stark geändert, Projektteile kommen hinzu oder werden ganz weggelassen. Im schlimmsten Fall muss die komplette bisherige Entwicklung wiederholt werden.

Ungenaue Anforderungen Ungenaue Anforderungen tauchen auf, wenn Funktionen oder Leistungsfaktoren zwar gewünscht sind, diese aber nicht genau genug bedacht oder kommuniziert wurden. So kann es während der Entwicklung vorkommen, dass Klärungsbedarf entsteht oder den Entwicklern Wissen über die genaue Beschaffenheit der gewünschten Anforderung fehlt. Dies kann zu Entwicklungen führen, welche nicht zu der benötigten Lösung vom externen/internen Kunden passen.

Im Folgenden wird agile Softwarentwicklung vorgestellt, welche darauf basiert, mit Änderungen an Anforderungen zu arbeiten und diese „selbst spät in der Entwicklung willkommen“[1] zu heißen. Die drei zuvor genannten Probleme mit Anforderungen sollten also mit agilen Prinzipien umgangen werden können.

2.2 Agile Softwareentwicklung

Bei der agilen Softwareentwicklung handelt es sich um eine Menge von Konzepten, welche ein Vorgehen in der Softwareentwicklung beschreiben. Nach dem Gabler Wirtschaftslexikon[2] steht Agilität im Zusammenhang mit Unternehmen für die „Gewandtheit [...] von Organisationen und Personen bzw. in Strukturen und Prozessen.“Bei Veränderungen sollen Firmen und Projekte in der Lage sein, flexibel auf Veränderungen zu reagieren. Dies steht im Gegensatz zu der herkömmlichen Vorgehensweise, ein Projekt zu Beginn möglichst genau zu planen und die meisten Schritte bereits zu Beginn der Durchführung zu kennen.

Das *Manifest für Agile Softwareentwicklung*[1] legt hierfür vier Werte und 12 Prinzipien fest.

1. „**Individuen und Interaktionen** mehr als Prozesse und Werkzeuge“
2. „**Funktionierende Software** mehr als umfassende Dokumentation“
3. „**Zusammenarbeit mit dem Kunden** mehr als Vertragsverhandlung“
4. „**Reagieren auf Veränderung** mehr als das Befolgen eines Plans“

Einige der Prinzipien, welche für diese Arbeit von Interesse sind, lauten wie folgt:

- „Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.“
- „Heisse Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.“
- „Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.“
- „Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.“

- „Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.“
- „In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.“

Der Fokus dieser Prinzipien liegt somit sowohl auf der kontinuierlichen und flexiblen Entwicklung als auch auf der regelmäßigen, engen Kommunikation im Team und mit den Beteiligten. Das Team soll sich selbst organisieren und die Prozesse auch während der Entwicklung laufend verbessern.

Ein bekanntes Modell für agile Softwareentwicklung bildet *SCRUM*. Rubin et al.[24] beschreiben, dass Anforderungen an das Produkt in einem *Product Backlog* festgehalten, priorisiert und anhand dieser Einstufung abgearbeitet werden. Durch dieses Vorgehen wird gewährleistet, dass das Produkt die wichtigsten Funktionen enthält und Anforderungen erfüllt, selbst wenn nicht alle Punkte der Liste abgearbeitet werden konnten. Das Arbeitn am Projekt erfolgt in kurzen Zyklen, den sogennanten *Iterationen*. Innerhalb dieser Sprints sollen funktionsfähige Artefakte entstehen. Zu Beginn jeder Iteration wird festgelegt, welche Aufgaben in dieser Zeit abgearbeitet werden sollen. Das Team arbeitet dabei selbstorganisiert. Bei Abschluss einer Iteration werden die Ergebnisse mit den Beteiligten besprochen. (*Sprint-Review*) Wenn jetzt Unzufriedenheiten auftauchen, werden bei Bedarf die Anforderungen angepasst oder die Prioritäten überarbeitet. Zusätzlich beschreiben Rubin et al.[25] eine *Sprint-Retrospektive*, welche nach der Review stattfinden sollte. Hier werden Probleme in den Prozessen angesprochen und bei Bedarf an den betroffenen Stellen nachgebessert. Ein weiteres Konzept bildet die tägliche Absprache namens *Daily SCRUM*, in der in kurzer Zeit von jedem Teammitglied erklärt wird, was seit dem letzten Treffen erledigt wurde, was jetzt erledigt werden soll und wo es Probleme gab.

Die genannten Vorgehensweisen zeigen, wie die Prinzipien des agilen Arbeitens in einem Projekt umgesetzt werden können. Die Sprints und die Review gewährleisten die kontinuierliche Auslieferung von Ergebnissen sowie die regelmäßige Rückmeldung der Beteiligten und die Anpassungsfähigkeit an Änderungen. Daily SCRUMs ermöglichen das tägliche Zusammenarbeiten sowie den Austausch von Angesicht zu Angesicht. Durch die interne Planung der Sprints ist das Team selbstorganisiert. Die Retrospektive sorgt für die regelmäßige Reflexion und Verbesserung der Prozesse.

Neben SCRUM existieren viele weitere Modelle, in dieser Arbeit wird aber zu einem großen Teil über SCRUM-basierte Vorgehensweisen gesprochen, weswegen die Vorstellung dieser Methode ausreichend ist, um einen Einblick in die Umsetzung agiler Prinzipien zu erhalten.

2.3 Grounded Theory Methodology

Nach Corbin und Strauss[26] handelt es sich bei der *Grounded Theory Methodology* (GTM) um eine Methodik, mit der eine Theorie basierend auf erhobenen und analysierten Daten entwickelt werden kann. Während der Erforschung der betrachteten

Thematik entwickelt sich die Theorie laufend weiter anhand von „[...] continous interplay between analysis and data collection.“[26] Durch wiederholten Vergleich der neuen mit den bereits erhobenen Daten wird die Theorie laufend weiterentwickelt. Ziel ist es, möglichst offen an die Daten heranzugehen und anhand dieser erst eine Theorie zu entwickeln, welche sich auf die Daten gründet. [27] Urquhart[27] beschreibt, dass GTM vor allem qualitative Daten verwendet. Dies passe zu den induktiven Herangehensweisen, welche verwendet werden, um von einzelnen Vorkommen in den Daten zu generellen Schlussfolgerungen zu gelangen.

Im Folgenden werden die Grundlagen der GTM vorgestellt.

Coding Ein grundsätzliches Vorgehen von GTM ist das *Coding*. Textabschnitten oder einzelnen Daten werden Kategorien zugeordnet, die sogenannten *Codes*. Sie beschreiben einen Sachverhalt an dieser Textstelle, welcher auch an anderen Textstellen vorkommt und diese dadurch verbindet.[27] Zum Beispiel kann eine Textstelle eine Ursache eines Problems mit Anforderungen enthalten. Der Textabschnitt erhält den Code *Ursache Probleme mit Anforderung*. Wenn dieser Code nun auch auf andere Textstellen angewandt werden kann, können die Unterschiede und Zusammenhänge dieser Vorkommen untersucht werden.

Constant Comparison *Constant comparison* bildet einen Grundstein der Vorgehensweise. Daten, welche einer Kategorie zugeordnet werden, werden immer wieder mit Daten der gleichen Kategorie verglichen.[27] Auf diese Weise erhält Kategorie X entweder eine genauere Definition der Gemeinsamkeiten innerhalb dieses Konzeptes oder es bilden sich neue Unterklassen, weil weitere Unterschiede aufgefallen sind, welche sich nochmals anders beschreiben lassen.

Memoing Memos werden während der Arbeit an den Daten erstellt. Sie enthalten die Ergebnisse der Analyse.[4] Dies kann auf verschiedene Arten repräsentiert werden. Sie können Erkenntnisse enthalten, die Grenzen der Konzepte und Kategorien stecken, vergleichen, Beziehungen zwischen den Kategorien aufzeigen oder Abläufe beschreiben.[4]

Theoretical Sampling *Theoretical Sampling* beschreibt die Datenerhebung, welche auf der Basis von bereits aus den Daten erarbeiteten Konzepten neue Daten erhebt. Während der Analyse der vorhandenen Daten fallen Lücken in den Daten auf oder neue Fragen stellen sich über bereits untersuchte Sachverhalte. Anhand dieser Erkenntnisse wird die darauffolgende Datenerhebung angepasst. So entwickelt sich die Datenerhebung selbst mit den Ergebnissen der Analyse weiter und Konzepte können tiefgehender untersucht werden, als es mit bereits erhobenen Daten oder der gleichen Datenerhebung wie zuvor möglich wäre.[5]

Es gibt mehrere Arten, die Grounded Theory Methodology durchzuführen. Für die vorliegende Arbeit wurde die in drei Phasen unterteilte Variante von Glaser[27] beziehungsweise von Corbin und Strauss[6] verwendet. Die Analyse teilt sich in drei

Schritte und kann, wie bereits beschrieben, starten, sobald die ersten Daten erhoben wurden.
Die erste Phase ist das sogenannte *offene Codieren* (open coding). Die Daten werden Schritt für Schritt durchgegangen und mit Codes versehen, nur eine grobe Idee einer Richtung zu haben oder ohne ein bestimmtes Ziel zu verfolgen. Hier können auch noch sehr viele, unterschiedliche Codes entstehen, welche die Abschnitte aus unterschiedlichsten Perspektiven kategorisieren. Wichtig ist hier, möglichst offen an die Daten heranzutreten.
In der zweiten Phase, dem *axialen Codieren* (axial coding, Corbin und Strauss[6]) beziehungsweise dem *theoretischen Codieren* (theoretical coding, Glaser[27]) , erhalten die bereits verteilten Codes gemeinsame Kategorien, welche größere Konzepte darstellen. Diese weisen schon auf eine Theorie hin und beginnen diese zu formen. Neue Codes werden im Hinblick auf diese Theorie entworfen. Aber auch in diesem Schritt ist das Vergleichen der Daten untereinander das wichtigste Mittel der Analyse.
Die dritte Phase nennt sich *selektives Codieren* (selective coding, Corbin und Strauss[6]. Ein einzelnes oder mehrere Hauptkonzepte wurden gefunden, welche den Kern der Theorie bilden. In dieser Phase werden die Daten im Hinblick auf diesen Schwerpunkt analysiert. Die Codierung fokussiert sich nun auf Kategorien, die für das Hauptkonzept relevant sind. Die für das Hauptkonzept uninteressanten Kategorien werden beiseitegelegt. Nachfolgend erhobene Daten werden jetzt unter den Gesichtspunkten des Hauptkonzeptes gesammelt.(*Theoretical Sampling*)

3 Wahl der Vorgehensweise

Um die vorliegende Fragestellung zu ergründen, suchte ich nach einer geeigneten Vorgehensweise. Hier fiel die Wahl auf qualitative, semistrukturierte Interviews auf Basis der *Grounded Theory Methodology*. Diese wurde in Abschnitt 2 bereits vorgestellt. Zu dieser Entscheidung führten folgende Überlegungen:

Wegen des eingeschränkten Umfangs einer Bachelorarbeit bot sich eine qualitative statt einer quantitativen Datenerhebungsmethode an. In der vorgesehenen Zeit ließ sich nur eine begrenzte Anzahl kooperierender Firmen finden. Zudem schränkte die vorherrschende Covid-19-Pandemie die Bereitschaft von Unternehmen ein, Zeit in Interviews mit mir zu investieren. So musste ich eine Methode verwenden, mit der anhand weniger Datensätze aussagekräftige Ergebnisse erzielt werden konnten.

Zusätzlich lag primär die Ergründung der Ursachen von Schwierigkeiten in der Anforderungserhebung und nicht die Ergründung der Häufigkeiten im Fokus. Standardisierte Fragebögen oder quantitative Interviews bieten wenig Raum für firmenspezifische Antworten auf die Frage nach Art und Ursprung der Probleme. Dieses Medium schränkt zusätzlich den Umfang der Antworten ein. Da sowohl die Struktur jedes Unternehmens als auch die Probleme mit Anforderungen unterschiedlich sind, war es nötig, schon während der Datenerhebung auf vielversprechende Inhalte näher einzugehen und mehr Details oder reale Beispiele zu erfragen.
Weiterhin bietet die qualitative Methode die Mögichkeit für Gesprächspartner, an-

hand der gestellten Fragen meine Schwerpunkte zu erkennen und darauf vertiefend einzugehen.

Quantitative Fragebögen schieden als Medium aus, da die Fragen anhand der *Grounded Theory Methodology* im Laufe der Bearbeitung angepasst werden sollten. Eine hohe Anzahl an Fragebögen kann nur ausgegeben werden, wenn die Fragen zu Beginn der Erhebung feststehen. Eine Nachkorrektur des Fragenkataloges würde eine schrittweise Durchführung der Befragung erfordern, was bei einer quantitativ hohen Menge an Daten in der relativ kurzen Bearbeitungszeit der Bachelorarbeit zu Problemen führen würde.

Um genaue Ursachen entstandener Probleme finden zu können, mussten nach Möglichkeit in den Unternehmen mehrere Interviews mit unterschiedlichen Personen geführt werden. Durch den Perspektivwechsel ließen sich andere Herausforderungen oder Ursachen für die Probleme finden. Mitarbeiter in verschiedenen Positionen sind unterschiedlich stark an Situationen, in denen Probleme mit Anforderungen auftreten, beteiligt und haben so unterschiedliche Blickwinkel auf die gleiche Problematik. Ein Anforderungsmanager zum Beispiel wird nicht auf die gleichen Herausforderungen stoßen wie ein Entwickler und auch nicht die gleichen Ursachen sehen.
Die Bereitschaft von Unternehmen, mehrere Mitarbeiter mit mir sprechen zu lassen, war selten vorhanden. Ich versuchte, dieses Problem zu lösen, indem ich möglichst viele Unternehmen anfragte.
Es war bei zwei kooperierenden Unternehmen möglich, mehrere Interviews zu führen.

Eine weitere Schwierigkeit bei der Befragung war, dass Fragen zum Vorgehen häufig mit Erläuterungen über den Soll- (*Wir würden [...] machen.*) und nicht über den Ist-Zustand (*Wir haben [...] gemacht.*) beantwortet wurden. So kam es vor, dass erdachte Strategien anstatt real verwendeter Methoden genannt wurden. Dieses Problem löste ich, indem ich nach konkreten Beispielen fragte, anhand derer der Umgang mit genannten Problemen in kürzlich real entstandenen Situationen beschrieben wurde. Dies minimierte die Gefahr, idealisierte Beschreibungen für das Vorgehen in Teams zu erhalten. Die aus den Gesprächen erhaltenen Daten variierten je nach Wissen und Mitteilungswillen der Gesprächspartner in ihrer Tiefe.

4 Durchführung der Interviews

Am Anfang der Bearbeitung stand die Hypothese, die in Abschnitt 1 bereits aufgeführt wurde.
Die drei Kategorien von Problemen mit Anforderungen entstammen den Erkenntnissen aus den Datenerhebungen von *NaPiRe*.
Von den Befragten gaben 75% an, dass ihr Team beziehungsweise Unternehmen *agile*, *rather agile* oder *hybrid* arbeitet. Die drei Kategorien *Unvollständige Anforderungen (incomplete or hidden requirements)*, *Zieländerungen (moving targets)* und *Ungenaue Anfor-*

derungen (underspecified requirements) zählten zu den *Top 5 Rated Problems*.[12]
Es gilt, herauszufinden, warum trotz teilweise oder komplett agil organisierten Teams weiterhin Schwierigkeiten mit den genannten Problemen auftreten.

Anhand dieser Überlegung erstellte ich das erste Interview, welches den Fokus auf die Vorgehensweise des Teams sowie die entstandenen Probleme legt (siehe Entwurf Abb. 1). Für die Fragen 2 bis 7 fragte ich explizit nach Beispielen, um einen Einblick in die tatsächlichen Abläufe zu erhalten.

Nach einer groben Transkription der geführten Interviews begann ich die eigentliche Analyse. Mit der GTM passte ich die Fragen nach bereits entstandenen Interviews an die neuen Erkenntnisse und Schwerpunkte an. Über die Bearbeitungszeit wuchs der Fragenkatalog, da weitere Fragen hinzukamen und neue Schwerpunkte durch die bisherigen Interviews motiviert wurden.

Grob fügte ich folgende Konzepte im Laufe der Analyse zusätzlich zu den bereits im ersten Entwurf (siehe 1) abgedeckten in darauffolgende Interviews ein:

- Formulierung der Anforderungen
- Beteiligte am Erhebungsprozess
- Kommunikation über Anforderungen und Änderungen, Reaktionen auf Änderungen
- Verwendung theoretischer Modelle
- Distanz zwischen technischer und kundenseitiger Perspektive

Anhang 2 zeigt den Entwurf, welcher für die letzten beiden Interviews mit nur kleinen Anpassungen je nach Gesprächspartner verwendet wurde und somit den finalen Entwurf darstellt. Anhand der Fragen lassen sich viele der Konzepte erkennen, die im Laufe der Bearbeitung hinzugekommen sind. Zum Beispiel ist mit der dritten Frage das Konzept fest in die Interviews integriert worden, welches die Formulierungen der Anforderungen und deren Auswirkung untersucht.
Je nach Gesprächspartner und bereits erhobenen Informationen zu dem Unternehmen und dem Team übersprang ich im Gespräch einzelne Fragen oder beleuchtete sie noch näher.
Zum Beispiel muss nicht noch einmal über das Ziel des Projektes gesprochen werden, wenn ein Gesprächspartner aus demselben Unternehmen zuvor schon von diesem berichtet hatte. Allerdings kann ein Projektleiter eventuell mehr zu den verwendeten theoretischen Modellen oder der Kommunikation mit dem Kunden sagen als ein Entwickler. Auch hatten unterschiedliche Gesprächspartner unterschiedlich tiefe Einblicke in Gespräche mit Kunden und Beteiligten sowie die Entscheidungsfindung. So ließen sich auch Erkenntnisse daraus gewinnen, wenn zwei Mitarbeiter in verschiedenen Positionen aus dem gleichen Projekt über die gleiche Situation sprachen.

Insgesamt führte ich acht Interviews in fünf Unternehmen.
Als Erstes sprach ich mit einem Berater (Entwickler) eines Consulting-Unternehmens mit Spezialisierung auf DevOps (UDevOps).[15] Das Unternehmen baut seit einigen Jahren einen Standort in Deutschland auf. In dieser Firma führte ich im Laufe der Bearbeitungszeit noch zwei weitere Befragungen mit beiden Teamleads (Teamlead I, Teamlead II) durch.[22, 23]
Das zweite Interview beantwortete ein Marketing-Verantwortlicher eines Softwareentwicklungsunternehmens (USoftware). Dieses implementiert Anwendungslösungen für Kunden. [19]
Der dritte kooperierende Konzern erstellt Gebäudeautomatisierungen (UGebäudeAuto). Dort sprach ich mit einem zuständigen Systemingenieur.[21]
Außerdem interviewte ich den Geschäftsleiter eines Unternehmens, welches eine Software für automatisierte Softwaretests entwickelt (UAutoTests). [18]
Abschließend führte ich zwei Interviews in einem Unternehmen, welches eine Softwarelösung für die Versorgungsbranche bereitstellt (UVersorgung). Ich sprach dort mit einem Entwickler und einem Projektleiter. [17][20]

Den Kontakt zu Gesprächspartnern stellte ich auf unterschiedlichen Wegen her.
Die Möglichkeit zu fünf Interviews in zwei Unternehmen erhielt ich durch den Kontakt zu Kommilitonen und Bekannten. Ein weiterer Kontakt wurde durch einen wissenschaftlichern Mitarbeiter an der Universität hergestellt. Für weitere Interviews durchsuchte ich Internetseiten, welche Stellenanzeigen von Unternehmen enthielten, die für Bachelorarbeiten mit Studenten zusammenarbeiten. Da bei diesen schon Bereitschaft an der Zusammenarbeit mit Studenten zu erkennen war, bot es sich an, speziell nach solchen Unternehmen zu suchen.
Bei der Auswahl der Unternehmen versuchte ich, Einblicke in möglichst viele unterschiedliche Unternehmen anzustreben. Meiner Vermutung nach sollten sich Unternehmen, welche zum Beispiel nach Auftrag Software für ein Unternehmen entwickeln, nicht so stark voneinader unterscheiden wie von einem Unternehmen, welches interne Software entwickelt und eine Standardlösung anbietet.

Wenn sich ein Unternehmen beziehungsweise ein Gesprächspartner bereit erklärte, ein Interview mit mir zu führen, wurde ein gemeinsamer Termin gesucht. Die Gespräche zeichnete ich mithilfe der Software *OBS*[13] auf. Sie fanden alle online über unterschiedliche Videokonferenz-Tools statt. Zu Beginn jeder Aufnahme holte ich die Erlaubnis der Gesprächspartner ein, dieses Interview aufzuzeichnen und damit für die vorliegende Arbeit weiterzuarbeiten. Gespeichert wurde nur der Ton des Gesprächs.
Die Interviews dauerten je nach Detailgrad des Gesprächs sowie der Mitteilungsbereitschaft und Wissensmenge des Gesprächspartners 30 bis 90 Minuten. Je nach Verlauf des Gesprächs passte ich die Fragen an oder fügte weitere hinzu, um möglichst die Bereiche genauer zu beleuchten, in die der jeweilige Gesprächspartner den genauesten Einblick hat. Weiterhin legte ich einen Schwerpunkt darauf, möglichst viele Beispiele zu erhalten. Dabei ging es vor allem um einen realen Einblick, der Abläufe, Ursachen und Wirkungen nicht idealisiert darstellt, sondern wie sie wirklich in einem dynamischen Umfeld ablaufen.

Im Anschluss an das Interview sicherte ich die Aufnahme und fügte sie in das Projekt in MAXQDA[9] ein. Diese Software wird im folgenden Abschnitt (Kapitel 5) kurz vorgestellt.

5 Analyse und Ergebnisse

Dieser Abschnitt erläutert die Analyse der Daten und deren Ergebnisse.

Die Analyse der Interviews führte ich, wie in Abschnitt 2 beschrieben, in den drei Phasen der Grounded Theory Methodology nach Corbin und Strauss[6] durch.
Die Textabschnitte der groben Transkriptionen versah ich mit verschiedenen Codes, welche die Inhalte in Kategorien einteilten. Die Gesamtmenge der Codes, welche zum Ende der Bearbeitung aufgestellt und verwendet wurden, können in Anhang 3 eingesehen werden.

Die Transkription der Interviews, die Bearbeitung der Daten und die Codierung führte ich mithilfe der Software *MAXQDA*[9] durch. Dort können Transkriptionen erstellt und direkt mit den Audiodateien verbunden werden. Zusätzlich lassen sich Codes erstellen, gewelche direkt Textstellen zugeordnet werden können. Codes können benannt, einem Farbcode zugeordnet und in Hierarchien eingeteilt werden. *MAXQDA* bietet auch die Möglichkeit, die Häufigkeiten der Codes und deren Verbindungen zueinander mit Tools zu visualisieren. Das vereinfacht es, den Überblick zu behalten und die Analyse durchzuführen.

Zu Beginn arbeitete ich nach der Methode des *offenen Codierens* alle bis zu diesem Zeitpunkt geführten Interviews durch und sammelte Textabschnitte mit ersten Code-Ideen und ihren Varianten.
Besonders sinnvoll erschien es hier, nach Strategien, Ursachen und Wirkungen zu suchen.
Nach der ersten Phase ließen sich folgende Klassen für Codes finden: *Sprache*, *Probleme mit Anforderungen*, *Formulierung der Anforderungen*, *Art des Projektes*, *Kommunikation*, *Vorgehensweise* und *Beteiligte am Erhebungsprozess*. Zu jeder Klasse gehören mehrere Codes, welche die Unterschiede als Unterklassen formulieren. Der Code *Sprache* zum Beispiel beinhaltete zwei weitere Codes, welche die Unterscheidung treffen, ob erklärte Sachverhalte mit Beispielen untermauert oder eher allgemein beschrieben wurden. Die Idee war hier, Informationen über die Realitätsnähe der Aussagen zu erhalten. Womöglich werden Sachverhalte in allgemeiner Sprache beschönigt, der Soll-Zustand und nicht der Ist-Zustand wird beschrieben.
Das Konzept *Probleme mit Anforderungen* enthielt genau die drei Arten von Problemen, welche bereits in Abschnitt 1 genannt wurden. Diese markierten also auch die Textabschnitte, welche besonders interessant sein könnten, da die Herkunft eben dieser Probleme untersucht werden soll.
In der Klasse *Formulierung der Anforderungen* findet man mehrere Codes, welche die Beschaffenheit der erhobenen oder an die Entwickler*innen weitergereichten Anforderungen beschreiben. Hier unterschied ich zwischen Anforderungen aus Kunden-

perspektive, aus technischer Perspektive, ob sie als Problem oder als Lösung formuliert waren und ob Abnahmekriterien fehlten. Die Idee für diese Klasse kam mir, da in mehreren Interviews Probleme mit den unterschiedlichen Perspektiven und dem Vokabular zwischen den Kunden und dem Unternehmen genannt wurden.
Mit *Art des Projektes* wurden mehrere Eigenschaften beschrieben, die ein Projekt haben kann. Hierzu gehören Codes über die Zielgruppe und zum Projektkontext. Diese Klasse enthält mögliche Ursachen für Probleme, die durch die Beschaffenheit des Projektes entstanden sein können.
Die Klasse *Kommunikation* enthielt zu diesem Zeitpunkt drei Codes, welche Ursachen für Schwierigkeiten in der Kommunikation zwischen den Beteiligten des Projektes markieren. Beispielsweise beschreibt der Code *Kommunikation: technisches vs. fehlendes technisches Wissen* die Problematik, dass Probleme in der Kommunikation auftreten, da einigen Beteiligten das technische Fachwissen fehlt, um die Thematik zu verstehen.
Die Strategie sowie die Arbeitsstruktur wird mit dem Code *Vorgehensweise* kategorisiert. Diese Klasse enthält die Arbeitsweise während des Projektes im Team und zeigt auch die Agilität beziehungsweise das Fehlen dieser auf. *Beteiligte am Erhebungsprozess* beschreibt die Menge unterschiedlicher Perspektiven, die aus dem entwickelnden Team an der Erhebung der Anforderungen beteiligt sind. Die Idee ist hier, dass verschiedene Perspektiven, vor allem die Kombination aus technischer und kundenseitiger Perspektive, einen ausführlicheren Blick auf das zu entwickelnde Produkt ermöglichen. Das Fehlen der technischen Perspektive zum Beispiel kann dazu führen, dass Anforderungen erhoben werden, welche im vorhandenen System gar nicht umsetzbar sind.
Ein Beispiel bildet die Klasse *Probleme mit Anforderungen*, welche die Codes *ungenau*, *Richtungs-/Zieländerungen* und *versteckte Anforderungen* enthält. Nach der Zuteilung zu Textabschnitten fiel auf, dass nur wenige Gesprächspartner ausführliche Beispiele über versteckte Anforderungen erläutert hatten, wohingegen viele Beispiele von Richtungsänderungen genannt wurden. Am Ende der Bearbeitung wurden Richtungsänderungen von sieben der acht Gesprächspartner genannt, während nur drei Gesprächspartner etwas über versteckte Anforderungen berichtet haben. Dies muss keine verlässliche Aussage über die reale Häufigkeit und die Schwere der Probleme mit versteckten Anforderungen sein, allerdings zeigt dies, dass die Gesprächspartner mehr über Richtungsänderungen berichten konnten. Somit ist dieser Code auch für die Analyse interessanter gewesen. Aufgrund dieser Erkenntnis wurde in darauffolgenden Interviews mehr nach dem Problem der Richtungsänderungen gefragt.
Mit diesen Klassen konnte ich nun unterschiedliche Vorkommnisse von Problemen, Beispiele und Vorgehensweisen in Zusammenhang bringen und sie miteinander vergleichen.

In der zweiten Phase, dem axialen Kodieren, untersuchte ich Ursache-Wirkung-Beziehungen anhand der bereits zugeteilten Codes. Zu diesem Zeitpunkt erhielten die Codes erste kurze Definitionen, um Textabschnitte transparenter und eindeutiger zuordnen zu können.
Hier fand das *Memoing* erstmals statt, welches in Abschnitt 2 bereits beschrieben wurde. Diese Memos enthielten erste Abgrenzungen der Kategorien voneinander, Defini-

tionen der Klassen und Fragen, welche sich im Zusammenhang mit diesen Kategorien stellten.
Mit diesen Definitionen verglich ich auch noch einmal jede Zuteilung, um fehlerhafte Codierungen zu entfernen oder Klassen aufzuteilen, falls sich innerhalb der Beispiele auffällige Unterschiede finden ließen.
Im Laufe der Analyse wurden die Memos weiter beschrieben, definiert und überarbeitet. Es kamen neue Grenzen hinzu oder Kategorien wurden in Unterkategorien geteilt.

Die Codes *Kommunikation: Stakeholder* und *Vorgehensweise: semi-agil* erhielten aufgrund diverser Überschneidungen weitere Unterklassen, um die Einteilungen aussagekräftiger zu machen. So kamen bei *Vorgehensweise: semi-agil* neben Weiteren die Codes *SCRUM Master oder Product Owner fehlen*, *Zwischenschieben von Tasks während Sprint* und *Retrospektive weggelassen/angepasst* hinzu. Der Code *Art des Projektes: nicht alle benötigten Funktionen im Team*, welcher bis zu diesem Zeitpunkt zur Klasse *Art des Projektes* gehörte, wurde auch zu *Vorgehensweise: semi-agil* verschoben, da das autonome Team in agilen Arbeitsweisen ein Grundprinzip ist und alle beteiligten jeden Tag zusammen arbeiten sollten. [1]

Um die eigentlichen Ursachen-Wirkungs-Beziehungen zu untersuchen, erhielten die realen Beispiele, welche die Gesprächspartner dargestellt hatten, jeweils mindestens einen neuen Code, welcher das Problem mit der Herkunft des Problems in Verbindung brachte. Bei der ersten Bearbeitung entstanden so zwölf Klassen. Bei Abschluss der Analyse waren es 15.
Die häufigsten Codes dieser Ursache-Wikungs-Klassen zu diesem Zeitpunkt waren *causal: Entscheidungsgewalt nicht im Team -> Fremdentscheidungen* (Vorkommen: 13), *causal: nicht alle Funktionen im Team -> Kommunikationsprobleme* (Vorkommen: 4) und *causal: fehlendes technisches Wissen Verantwortliche -> Fehlentscheidungen* (Vorkommen: 13).
Da das autonome Team, welches alle benötigten Funktionen intern haben sollte, ein Grundprinzip der agilen Arbeit bildet, ist anhand dieser Codes zu erkennen, dass in den Teams, welche in diesen Interviews beschrieben wurden, nicht allen agilen Prinzipien gefolgt wird.
Der Code *causal: Entscheidungsgewalt nicht im Team -> Fremdentscheidungen* lässt sich mit folgendem Beispiel gut veranschaulichen:
Der Projektleiter von UVersorgung[20] beschrieb eine Situation, in der ein großer Kunde sich von diesem UVersorgung hinsichtlich der Standardisierung seiner Prozesse beraten ließ. Der Ansprechpartner dort verließ das Unternehmen und ein neuer wurde eingesetzt, welcher sich nun erst einlesen musste und von den Zuarbeitenden beraten wurde. Diese sprachen sich gegen Standardisierungen und für spezielle Entwicklungen aus. Aufgrund dessen wurden Spezialentwicklungen angefordert. Übergangsweise wurde dem Kunden dann die Standardsoftware zur Verfügung gestellt und Schritt für Schritt an dessen Wünsche angepasst. Bei den Workshops zum Einführen der erarbeiteten Lösung kam dann die Frage auf, warum nicht die Standardlösung verwendet wurde und teure Zusatzentwicklungen bezahlt werden mussten. Das Unternehmen hatte den Kunden zu Beginn dahingehend beraten, es wurden aber trotz dessen weitere Entwicklungen angefordert. Die Entscheidungsgewalt über das Vorgehen für die

Entwicklung der Lösung lag nicht im Team, was zu einer problematischen Entscheidung von außen führte. ([20] 18:12-22:24)
Für den Code *causal: nicht alle Funktionen im Team -> Kommunikationsprobleme* beschrieb Teamlead II von UDevOPs[23] einen Fall, in dem der Ansprechpartner eines Kunden mehrere Teams hatte, welche aber alle extern waren. Kritik an Prozessen wurde dann nicht gut angenommen, da sie über mehrere Ecken weitergetragen werden musste.([23] 52:20-55:08) Hier führte das nicht umgesetzte agile Prinzip der direkten Kommunikation von Angesicht zu Angesicht[1] zu Schwierigkeiten in der Kommunikation.
Das Problem *causal: fehlendes technisches Wissen Verantwortliche -> Fehlentscheidungen* wurde in folgendem Beispiel deutlich:
Projektleiter II aus UDevOps[16] arbeitete in einem früheren Unternehmen als neuer Projektleiter. Dort erhielt er die Anweisung, die komplette Plattform innerhalb von drei Monaten an eine andere Stelle zu migrieren. Der Entscheidungsträger/die Entscheidungsträgerin war vermutlich auf einen Marketingtrick hereingefallen, denn dies war in dieser Zeit nicht möglich. Obwohl ihm vermutlich von technisch versierten Mitarbeitern davon abgeraten wurde, hatte dieser das kurze Zeitfenster angeordnet. Die Migration ließ sich nicht rechtzeitig abschließen und das Projekt benötigte mehrere Anläufe.([23] 45:16-50:24) Aufgrund des fehlenden Wissens entschied der Entscheidungsträger/die Entscheidungsträgerin falsch. Das Team konnte nicht selbstorganisiert arbeiten, da Entscheidungen zum Vorgehen von außen getroffen wurden. Hier wurde das agile Prinzip des selbstorganisierten Teams nicht durchgesetzt.[1]

Zusätzlich fiel der Code *causal: SCRUM falsch angewandt -> Probleme im Team* (Vorkommen: 3) auf, da dieser Aussagen markierte, welche zeigten, dass die falsche Anwendung von SCRUM zu Problemen während der Entwicklung führen kann und damit für die betrachtete Fragestellung entscheidend ist.
Zwei Textstellen beschrieben hierbei SCRUM Dailies, welche nicht als knappe Meetings zum Statusabgleich verwendet wurden, sondern lange Diskussionen des Teams darstellten. ([22] 1:24:51-1:26:48; [23] 37:09-37:27)
Das letzte Beispiel zeigt, dass im Team des Unternehmens UAutoTest keine Dailies nach SCRUM durchgeführt werden, da diese von den Mitarbeiter*innen nicht angenommen wurden. Die Zeitabschätzung im Team wirft noch Probleme auf und Entwickler müssen bei dringenden Problemen auf Kundenseite diese zuerst lösen. ([18] 29:54-32:12)
Dieser Abschnitt erhielt aus diesem Grund auch den weiteren Code *causal: Entwickler arbeiten an mehreren Stellen -> Probleme Entwickl*, welcher beschreibt, dass Probleme in der Entwicklung auftreten, da Teammitglieder noch in anderen Projekten oder bei Kunden aushelfen müssen. So können sie nicht komplett in das Projekt eingeplant werden und fallen eventuell sogar aus. Trotz Miteinberechnung gewisser Ausfallzeiten kam es bei UAutoTest zu Fehleinschäzungen des Zeitaufwands, was zusätzlich mit den fehlenden regelmäßigen Rücksprachen zusammenhängen könnte.

UAutoTest war das einzige befragte Unternehmen, welches ausschließlich für das eigene interne Produkt entwickelt. Aus diesem Grund kam hier eine weitere Ursachen-Wirkung-Beziehung hinzu: Kunden dürfen zwar Wünsche an das Produkt formulie-

ren, falls diese für das Standardprodukt aber nicht sinnvoll erscheinen, werden diese Anforderungen bei Bedarf angepasst oder ganz weggelassen. So kann es passieren, das etwas implementiert wird, was nicht zur Vorstellung des Kunden passt. ([18] 15:06-18:04) Mit einem entsprechenden Code ließen sich drei Beispiele finden, anhand derer dieses Problem zu erkennen ist. ([18] 15:06-18:04, 35:44-37:55, 40:58-47:40) Da dieses Problem auf Unternehmen beschränkt ist, die nur für ihr eigenes Produkt entwickeln, ließen sich keine weiteren Beispiele innerhalb der anderen Interviews finden.
Ähnlichkeiten zu UVersorgung ließen sich trotzdem erkennen, da dieses Unternehmen ein internes Produkt vertreibt, bei Bedarf aber auch Entwicklung betreibt, welche für die Kunden eine individuelle Anpassung der Software an ihre Prozesse ermöglicht. ([17] 5:50-7:11)
Die Abhängigkeit von Kundenwünschen kommt hier hinzu, da nicht primär für das interne Produkt entwickelt wird, sondern das Produkt bei Bedarf an den Kunden angepasst wird. Bei der Analyse der beiden geführten Interviews in diesem Unternehmen kamen drei Ursache-Wirkung-Codes hinzu, welche beschreiben, dass die Distanz zwischen technischer und kundenseitiger Perspektive zu Missverständnissen, die Abhängigkeit nach außen, zum Beispiel zum Kunden, zu Wartezeiten und nicht weitergetragene Informationen zu Unklarheiten führen können.

Insgesamt vier Beispiele ließen sich für den Fall finden, dass der Aufwand für Projekte falsch eingeschätzt wurde und dies zu hohem Ressourcenaufwand oder Richtungsänderungen führte, um die Projekte trotzdem erfolgreich durchführen zu können.
Einen spannenden Fall bildet hier die bereits zum Thema fehlendes Wissen von Entscheidungsträger*innen aus dem Gespräch mit Teamlead II von UDevOps[23] behandelte Situation: In einem früheren Unternehmen sollte Teamlead II ein Projekt umsetzen, welches ohne große Planung von Führungskräften angesetzt wurde. Dieses sollte wider besseres Wissen in zu kurzer Zeit umgesetzt werden. Das Projekt scheiterte, da die Informationen aus dem Fachgebiet nicht zu den Führungskräften durchgedrungen waren oder von diesen ignoriert wurden. Eine Verlängerung des Projektes wurde nötig. ([23] 45:08-50:32)

Ein weiterer klarer Zusammenhang innerhalb der Codes bildete eine zu genaue Planung von Projekten, welche zu hohem Planungsaufwand und wenig Flexibilität innerhalb des Projektes führte.(Vorkommen: 3;[15] 14:20-15:32; [23] 11:58-15:50;[21] 1:07:24-1:08:20) Eine sehr genaue Planung steht für eine wasserfallartige Herangehensweise, da in agilen Arbeitsweisen eher inkrementell je nach Stand und neuen Erkenntnissen während der Sprints geplant wird.
Mehrere Befragte antworteten, dass Ungenauigkeiten in den Anforderungen alltäglich seien. Beispielsweise beschrieb der Entwickler von UDevOps[15], dass dies ständig passiere und meist nur das Ziel in den Anforderungen der Kunden beschrieben wird. Dies sei allerdings teilweise direkt im Team klärbar.([15] 27:52-28:40)
Es ließen sich keine genaueren Beispiele mit Abläufen und Ursprüngen erfahren.
Auch hier gab es Zusammenhänge zu gescheiterten Projekten, jedoch sagten mehrere Gesprächspartner aus, dass der Arbeitsalltag in Projekten daraus bestehe, mit ungenauen Anforderungen zu arbeiten. Laut dem Teamlead I von UDevOps[22] sei dies

„normalerweise eigentlich der Regelfall“. Vor allem bei neuen Ideen erkenne man erst während der Entwicklung, wie genau die Umsetzung aussehen muss.([22] 00:59:20-01:00:30)

Nach Analyse der Code-Häufigkeiten und den Zusammenhängen zwischen Codes fiel auf, dass vor allem der Code *Probleme mit Anforderungen: Richtungsänderungen* viele klare Ursachen und Zusammenhänge hervorbrachte. Dieser Code erhielt folgende Definition:

In einem Projekt muss die Richtung geändert werden, da sich das Ziel geändert hat oder Anforderungen hinzugekommen sind, welche eine starke Richtungsänderung fordern. Das bedeutet, dass das Projekt neu aufgerollt werden muss oder Projektteile hinzukommen beziehungsweise weggelassen werden müssen. Das Projekt konnte ohne eine starke Änderung nicht erfolgreich weiterlaufen.

Die dritte Phase, das selektive Codieren, begann damit, dass dieser Code weitere, genauere Einteilungen erhielt und die Richtungsänderungen in Projekten in Unterkategorien geteilt wurden, welche den Ursprung der einzelnen Beispiele näher beleuchteten. Hier ließen sich sechs Ursprünge finden: Anforderungen wurden falsch/unvollständig umgesetzt, fehlende Flexibilität bei neuen Erkenntnissen, fehlende Planung, nicht alle Perspektiven wurden bei der Planung miteinbezogen, Fehleinschätzungen beziehungsweise fehlendes Wissen, Stakeholder.

Ein von den Richtungsänderungen unabgängiger Code beschrieb Textstellen, in denen gescheiterte Projekte auftraten.
Gescheitert bedeutet hier, dass ein Projekt oder eine Entwicklung abgebrochen werden musste, beziehungsweise es nur mit einer starken Veränderung, Verlängerung oder Budgeterhöhung möglich war, das Projekt erfolgreich abzuschließen.
Beispiele, welche diesen Code erhielten, waren für die Analyse besonders interessant, da hier die stärksten Auswirkungen von Problemen mit Anforderungen beschrieben wurden. Die meisten Zusammenhänge mit diesem Code zeigten Textstellen, die eine Richtungsänderung in den Anforderungen aufzeigten. ([19] 14:05-18:48, 24:11-25:00; [22] 1:01:10-1:01:41, 1:01:40-1:06:09; [23] 21:04-21:28, [20] 18:12-22:24)
Das bedeutet, dass vor allem Richtungsänderungen innerhalb dieses Datensatzes zu schwerwiegenden Problemen geführt haben.
Bei drei dieser fünf Vorkommnisse innerhalb der Daten handelte es sich um eine Fehleinschätzung oder fehlendes Wissen, als Ursache des Scheiterns.([19] 24:11-25:00; [22] 1:01:10-1:01:41; [23] 21:04-21:28) Ein Beispiel dafür zeigte der Marketingverantwortliche von USoftware[19]: Er beschrieb einen Fall in einer ehemaligen Firma, bei dem eine Software für den freien Markt auf Wunsch eines Kunden entwickelt werden sollte, dieser aber falsch eingeschätzt wurde und nicht sie daher nicht auf dem Markt landete. ([19] 24:11-25:00) Hier führte eine Fehleinschätzung des Marktes zum Scheitern des Projektes.

Ein Beispiel eines aus einem anderen Grund als einer Fehleinschätzung gescheiterten Projektes brachte Teamlead I von UDevOps[22] vor.

In einem Kundenunternehmen wurde die Datenbankstruktur zur einfacheren Bedienung für die internen Entwickler angepasst. Die Datenbank wurde zu einer NoSQL-Datenbank umstrukturiert. Nun sollte die Datenbank durch UDevOps in die Cloud migriert werden, was aufgrund der abstrahierten Struktur zu Problemen führte. Nun mussten alle vorherigen Änderungen wieder rückgängig gemacht werden, was mit hohem Aufwand verbunden war. ([22] 1:01:40-1:06:09) An diesem Beispiel lässt sich gut erkennen, dass die fehlenden Überlegungen aus anderen Perspektiven bei der ersten Umstrukturierung der Datenbank zu Problemen bei der Migration in die Cloud geführt hatten. Es wurden nicht alle Perspektiven beachtet, weshalb eine Richtungsänderung nötig geworden war. Stakeholder von außen führten zum Scheitern des Projektes.

Der Marketingverantwortliche von USoftware[19] beschrieb auch einen Stakeholder als Ursprung eines gescheiterten Projektes.
Es sollte für ein Unternehmen eine Software entwickelt werden. Nach der Erarbeitung der gewünschten Lösung musste diese nur noch beim Kunden eingeführt werden. Als dies getan werden sollte, stimmte die interne Informationstechnik diesem Vorhaben aber nicht zu. Sie schien nichts von einem solchen Vorhaben zu wissen. ([19] 14:05-18:48) Hier wurde der Stakeholder *interne IT* bei der Planung nicht beachtet. Dies führte zur fehlenden Kooperation dieser und dem Scheitern des Projektes.
Zusätzlich scheint hier kein regelmäßiges Feedback nach den agilen Prinzipien eingeholt worden zu sein und nicht in regelmäßigem Austausch mit den Fachexperten gestanden zu haben.[1] Mit diesen Maßnahmen wäre die interne IT womöglich früher miteinbezogen worden.

Aus diesen Beispielen lässt sich schließen, dass das Scheitern von Projekten auch von außen herbeigeführt werden kann. Hier bestehen häufig Widersprüche zum agilen Prinzip der selbstorganisierten Teams, welche in regelmäßigem Austausch mit allen Beteiligten stehen sollten.[1] Ab Projektbeginn sollte ein agiles Team bevollmächtigt sein, Entscheidungen selbst zu fällen und die Richtung vorzugeben. Dass hier Einfluss von außen genommen werden konnte, zeigt, dass höchstens semi-agil gearbeitet wurde.

Zusätzlich zu den Ursachen erhielt die Klasse der Richtungsänderungen auch noch Unterklassen, welche die Wirkung der Richtungsänderung beschrieben. Insgesamt ließen sich drei verschiedene Auswirkungen in den Daten finden: *Wirkung: Neustart des Projektes/Wiederholung*, *Wirkung: Budget-, Zeiterhöhung* und *Wirkung: Scheitern des Projektes/Abbruch*. Das Scheitern des Projektes ist in diesen Codes nicht mit einer starken Budgeterhöhung oder einer Verlängerung gleichgesetzt, da hier eine genauere Unterteilung der Wirkung vorgenommen werden sollte. Mit den Wirkungscodes und den Herkunftscodes zeigt sich nun für jede Richtungsänderungen in den Daten ein klarer Ablauf.

Um die aufgestellte These genauer zu untersuchen, erhielten Textstellen, die eine Richtungsänderung aufwiesen und direkt in Verbindung mit fehlender oder falsch angewandter Agilität standen, den Code *Richtungs-/Zieländerungen: Verbindung zu feh-*

lender Agilität.
Beim Durchsehen der in Frage kommenden Stellen fiel auf, dass ein großer Teil der Vorkommnisse aufgrund von Kunden- beziehungsweise Stakeholderentscheidungen oder Änderungen am Markt auftrat. Diese Fälle erhielten noch einmal einen Code (*Richtungs-/Zieländerung: Richtungsänderung von außen*), welcher dem zuvor genannten teilweise gegenübersteht. Beispielsweise erläuterte der Projektleiter von UVersorgung[20] in diesem Zusammenhang folgende Situation: Ein Kunde benötigte viele zusätzliche Entwicklungen für die Schnittstellen zu einem Partner im Ausland. Kurz vor Ende des Projektes, als UVersorgung viele Sonderentwicklungen bereits implementiert hatte, entschied sich der Kunde dann aus mikropolitischen Gründen, doch einen anderen Anbieter zu nehmen, wodurch alle bis dahin entwickelten Lösungen nicht mehr benötigt wurden.([20] 26:18-28:00)

Zwar wurden diese Ursachen in anderen Codes schon indirekt beschrieben, allerdings sollte hier noch einmal die Verbindung zu den Richtungsänderungen beschrieben werden und eine kleinere Unterteilung innerhalb des näher untersuchten Bereichs einen besseren Überblick gewähren.
Drei der zwölf markierten Textstellen wiesen eine Verbindung zu fehlender Agilität auf. Zwei dieser beschriebenen Sachverhalte wurden aber allgemein beschrieben und es handelte sich nicht um konkrete Beispiele. Das dritte Beispiel beschrieb der Entwickler von UDevOps[15]. In einem Projekt mit einem großen Kunden plante dieser die Projektphasen, zu jeder Phase gehörte ein Endsystem. Diese Phasen sollten nacheinander abgearbeitet werden. Ohne Vorwarnung wurden dann Projektphasen gestrichen und an kundeninterne Mitarbeiter gegeben, schließlich wurden sie doch wieder von UDevOps entwickelt. ([15] 10:03-11:50) Die Vermutung liegt hier nahe, dass keine enge Kommunikation mit dem Kunden während der Entwicklung stattgefunden hat, eventuell hat das regelmäßige Feedback des Kunden gefehlt. Oder das Problem war, dass das Team selbst nicht komplett für die Entwicklung zuständig war, da die Entscheidungsgewalt immer noch beim Kunden lag. Dieser hatte Teile des Projektes während der Durchführung an andere Entwickler abgegeben, wodurch nicht mehr alle Beteiligten in einem Team arbeiteten.
Dieses Vorkommnis lässt sich aber auch nicht mit hundertprozentiger Sicherheit auf fehlende Agilität zurückführen. Das Team selbst hat so weit wie möglich agil gearbeitet, wenn aber der Kunde nicht bereit war, regelmäßig Feedback zu geben, zum Beispiel über eine Review, lag hier das Problem nicht in der fehlenden Agilität des Teams, sondern in der fehlenden Bereitschaft des Kunden, Entscheidungsgewalt über die Entwicklung oder das Vorgehen an das Team abzugeben. Der Kunde selbst hatte noch genug Entscheidungsgewalt, um Verantwortlichkeiten zu entziehen oder durchführungsbezogene Entscheidungen zu treffen.
Somit war das Team aber auch nicht mehr selbstorganisiert, da nicht alle Entscheidungen nach bestem Wissen für das Projekt getroffen werden konnten. Dies widerspricht dem agilen Vorgehen.[1] Trotzdem ist hier nicht ganz klar, ob das Team selbst für die fehlende Agilität verantwortlich gemacht werden kann.

Sechs der zwölf Textstellen ließen sich nicht komplett durch fehlende Agilität, sondern auch durch den Einfluss von außen erklären.
Ein Beispiel bildet die Situation, die der Marketingverantwortliche von USoftware be-

schrieb:
In einer ehemaligen Firma wurde eine Software für einen großen Kunden entwickelt. Als die Lösung dann in der internen IT beim Kunden ausgerollt werden sollte, wusste die IT selbst nichts von dem Projekt und ließ Tests im internen System nicht zu. ([19] 14:20-15:15) Hier wurde ein interner Stakeholder des Kunden bei der Planung und Durchführung des Projektes ignoriert, das Scheitern wurde durch das Hinzukommen dieses Stakeholders zum Ende des Projektes verursacht. Dies ist eine Situation, in der die Richtungsänderung auch von außen herbeigeführt wurde und nicht nur durch fehlende Agilität im Entwicklerteam erklärt werden kann. Wäre komplett agil vorgegangen worden, hätte es vielleicht schon früher Softwareprototypen gegeben, welche in der internen IT getestet worden wären. So wäre der fehlende Stakeholder eventuell schon früher involviert gewesen.
Das eigentliche Problem lag aber an der fehlenden Einbindung des Kunden der eigenen IT beziehungsweise die fehlende Kommunikation des Entwicklerteams mit der kundeninternen IT. Laut dem agilen Manifest sollen Fachexperten täglich mit den Entwicklern zusammenarbeiten, was hier nicht der Fall war.[1]

Ein weiteres, interessantes Beispiel erläuterte der Projektleiter von UVersorgung[20] im Zusammenhang mit Richtungsänderungen.
Das Beispiel wurde zu einem früheren Zeitpunkt der Arbeit bereits vorgestellt, ist aber auch hier von Bedeutung. Der Kunde wollte die Branchenlösung für sich anwenden, es wurde eine Lösung mit der Standardanwendung entwickelt. Als der verantwortliche Mitarbeiter das Unternehmen verließ, veranlasste der neue Verantwortliche spezielle, teurere Entwicklungen anstelle der Standardlösung. Als die Software daraufhin bei den Mitarbeitern mithilfe von Workshops eingeführt wurde, folgten Fragen nach dem Sinn der Sonderentwicklungen, da die Standardlösung ausgereicht hätte.([20] 18:13-20:11) Hier wurde zu Beginn richtig beraten, der Kunde selbst änderte aber auf eigenen Wunsch die Richtung, nach der Entwicklung fiel auf, dass der ursprüngliche Weg besser gewesen wäre. Dieses Problem hätte sich durch Agilität vermutlich nicht verhindern lassen können, da die Änderung trotz vorheriger Beratung vom Kunden ausging. Auch der plötzliche Wechsel des Verantwortlichen ist nicht mit fehlender Agilität zu erklären.

Es fällt auf, dass ein Teil der Richtungsänderungen womöglich durch richtig angewandte Agilität verhindert worden wäre, dass einige der Vorkommnisse aber selbst durch Umstrukturierung im Team nicht hätten verhindert werden können. Zumindest die Situationen, in denen Kunden Entscheidungen für das Projekt getroffen haben, hätten zwar durch das Prinzip der autonomen Teams, welche alle benötigten Funktionen und die Entscheidungsgewalt innehaben, verhindert werden können, alleredings lässt sich solch eine Situation in realen Unternehmen selten herstellen. Vor allem, wenn für externe Kunden entwickelt wird, ist es meist nicht möglich, alle direkt beteiligten Parteien zu jeder Zeit in das Projekt einzubinden. Der Entwickler von UVersorgung beschrieb dies folgendermaßen: „Sie wollen halt ein System haben und haben [...] einen Dienstleister, der sich darum kümmert. Sie wollen sich eigentlich gar nicht damit befassen.“([17] 15:02-15:10) Die Aufgabe des Entwickelns wird an das Unternehmen abgegeben. Es soll möglichst ohne Mehraufwand für den Kunden erledigt

werden. Kunden haben selbst ein Tagegeschäft und eigene Aufgaben zu bewältigen.

Keines der befragten Unternehmen gab an, komplett agil zu arbeiten. Es wurde meist von einigen agilen Prozessen oder einer angepassten agilen Methodik gesprochen. Aufgrund dieser Erkenntnis entstand ein weiteres Konzept, *Ursache für fehlende Agilität*, welches den Grund dafür näher beleuchten sollte. Insgesamt 13 unterschiedliche Textstellen wurden hiermit beschrieben.
Anschließend wurden die markierten Textstellen verglichen und Unterklassen hinzugefügt, welche die Ursachen genauer beschrieben.
Es ließen sich fünf verschiedene Ursprünge finden: *Projektstruktur* (Vorkommen: 3), *Falsche/Unvollständige Anwendung* (Vorkommen: 4), *Fehlende Kooperation der Mitarbeiter/des Managements* (Vorkommen: 4), *Fehlende Kooperation der Kunden/Stakeholder* (Vorkommen: 3), *Laufender Prozess zu mehr Agilität* (Vorkommen: 2).

Die Projektstruktur beschreibt als Ursache für fehlende Agilität den Projektkontext. Beispielsweise nannte der Entwickler von UDevOps[15] den Grund der Anpassung von SCRUM an das eigene Team als Maßnahme, da es sich um ein kleines Team handele und SCRUM strenge Vorgaben setze.([15] 33:28-33:45)
Der Systemingenieur von UGebäudeAuto[21] beschrieb im Gegensatz dazu, dass das agile Arbeiten im Unternehmen für alle Teams grundsätzlich möglich sei, der Termin der Fertigstellung beim Unternehmen klar im Vordergrund stehe. Es handele sich außerdem um Bereichsteams.([21] 32:54-40:43) Das Vorgehen kann also nicht so flexibel gehandhabt werden und die Teams haben nicht alle benötigten Funktionen in sich vereint.
Falsche oder unvollständige Anwendung beschreibt Teamlead I von UDevOps[22] in einem Beispiel:
Bei einem Kunden dauerten die SCRUM Dailies teilweise bis zu 50 Minuten und wurden schon vom Team selbst „Monster-Dailies“genannt. Es wurden nicht die typischen Fragen beantwortet, stattdessen wurden gemeinsam die E-Mails gelesen.([22] 01:24:50-01:26:50)
Die Methodik der SCRUM Dailies wurde falsch angewandt und führte zu einer schlechten Arbeitsatmosphäre.

Die fehlende Kooperation von Mitarbeitern ließ sich bei UAutoTest[18] erkennen: Es wurde versucht Daily StandUp Meetings einzuführen, allerdings wurden diese vom Team nicht angenommen.([18] 29:53-30:15)

Sowohl der Entwickler als auch Teamlead I von UDevOps[15][22] nannten hingegen Kunden, welche entweder im Gegensatz zum Team wasserfallartig gearbeitet haben oder sich nicht für die regelmäßige Review interessiert haben, weshalb nicht agil gearbeitet werden konnte.

Der Entwickler und der Projektleiter von UVersorgung[17][20] beschrieben, dass aktuell agile Methodiken in die Prozesse des Teams integriert werden. Aus diesem Grund arbeitet das Team zwar aktuell noch nicht vollständig agil, beugt aber mit ersten agilen Vorgehensweisen bekannten Problemen vor.

6 Zusammenfassung

Die zugrunde liegende These, die den Anstoß für diese Arbeit gegeben hat, lautet:

> Auftretende Probleme mit Anforderungen der drei Kategorien *Unvollständige Anforderungen (incomplete or hidden requirements), Zieländerungen (moving targets), ungenaue Anforderungen (underspecified requirements)* ließen sich mit agilen Methoden der Softwareentwicklung umgehen. Somit liegt eine Ursache für diese Probleme darin, dass nicht oder unvollständig agil gearbeitet wird.

Viele Gründe für die Probleme mit Anforderungen in Unternehmen konnten gefunden werden und wurden in Abschnitt 5 aufgeführt.

Ungenaue und versteckte Anforderungen werden in den befragten Firmen meist nicht mehr als schwerwiegendes Problem betrachtet. Dies liegt vermutlich daran, dass die zumindest teilweise agilen Prozesse Problemen durch kleinere Änderungen vorbeugen.

Vor allem wurden die Richtungs- beziehungsweise Zieländerungen in Projekten als häufiges Problem genannt und schienen auch die schwerwiegensten Folgen mit sich zu bringen. Sie können zu höheren Projektkosten, einer längeren Projektdauer und teilweise sogar zur Wiederholung ganzer Projektteile führen.
Es fällt auf, dass die unvollständige Anwendung agiler Prinzipien häufig als Ursache für Richtungsänderungen genannt werden konnte oder diese zumindest hätte verhindern können. Vor allem die fehlende Entscheidungsgewalt im entwickelnden Team führte sehr häufig zu Problemen mit Richtungsänderungen.
Aber auch fehlende regelmäßige Rückmeldungen schienen oft Probleme mit Änderungen in den Anforderungen nach sich zu ziehen.
Diese beiden Ursachen lassen sich mit der korrekten Umsetzung aller agilen Prinzipien vermeiden. Somit scheinen in diesen Fällen tatsächlich die unvollständig oder gar nicht angewandten agilen Prozesse eine Ursache für die weiterhin auftretenden Probleme darzustellen. Die vollständige und korrekte Durchführung agiler Prinzipien würde die Auswirkungen der Richtungsänderungen eventuell verhindern.

Allerdings gab es auch einige Fälle, in denen die fehlende Agilität nicht für die Probleme mit den Anforderungen verantwortlich gemacht werden kann. Wenn Änderungen auf dem Markt oder Kundenforderungen hinzukommen, lassen sich diese manchmal nicht mit agilen Prozessen abfedern, vor allem wenn sie ohne vorherige Kommunikation direkt vom Kunden festgelegt werden. Wenn trotz gegensätzlicher Beratung eine Forderung gestellt wird, muss sich das Team häufig anpassen.

7 Ausblick

Anhand der in dieser Arbeit beschriebenen Erkenntnisse eröffnen sich weitere Forschungsmöglichkeiten.

Die Ursachen für Probleme mit Anforderungen könnten in großem Umfang qualitativ untersucht werden. Mit einer größeren Menge an befragten Unternehmen sowie mehreren Interviews innerhalb der Unternehmen selbst ließen sich noch genauere Aussagen über die Herkunft verschiedener Probleme finden.

Zusätzlich bietet es sich an, Handlungsempfehlungen für die vorliegenden oder ähnlichen Vorkommnisse für die Unternehmen zu formulieren. Das Lesen dieser Arbeit könnte einzelne Unternehmen bereits bei der Erkennung der problematischen Arbeitsweisen unterstützen, das Finden der zugehörigen Lösungen ist anschließend firmenspezifisch nötig.
Auch Kunden könnten von der Anpassung an das agile Vorgehen ihrer Dienstleister profitieren, da eine schnelle Anpassung der Anforderungen zu Wettbewerbsvorteilen führen kann.[1] Allerdings setzt das auch voraus, Projekte in enger und regelmäßiger Zusammenarbeit mit dem Dienstleister durchzuführen. Das in Abschnitt 5 genannte Problem, dass Kunden häufig die Verantwortung abgeben und mit dem Projekt so wenig wie möglich zu tun haben möchten, steht dazu im Gegensatz.
Vermehrte Aufklärung über die Vorteile agiler Maßnahmen würde vermutlich die Kooperation der Mitarbeiter, der Kunden und des Managements stärken. Im Umkehrschluss könnte es zu fehlender Kooperation kommen, wenn die Sinnhaftigkeit neuer Maßnahmen nicht angemessen kommuniziert wird. Die Übergabe von Verantwortung des Managements an die Teams wurde mehrmals als Schwierigkeit genannt. Es ist aber bei agilem Vorgehen sinnvoll, das Team selbst entscheiden zu lassen, wie das Projekt am besten durchzuführen ist, da es den tiefsten Einblick hat.

Literatur

[1] Cockburn et al. *Manifest für Agile Softwareentwicklung*. URL: https://agilemanifesto.org/iso/de/manifesto.html.

[2] Prof. Dr. Oliver Bendel. *Agilität*. Springer Gabler. URL: https://wirtschaftslexikon.gabler.de/definition/agilitaet-99882.

[3] Stahn Bühne und Andrea Herrmann. *Handbuch Requirements Management nach IREB Standard*. International Requirements Engineering Board, 2015, S. 19–20.

[4] Juliet M. Corbin und Anselm L. Strauss. *Basics of Qualitative Research (3rd ed.): Techniques and Procedures for Developing Grounded Theory*. Los Angeles; London; New Delhi; Singapore; Washington DC; Boston : SAGE, 2008, S. 117–142.

[5] Juliet M. Corbin und Anselm L. Strauss. *Basics of Qualitative Research (3rd ed.): Techniques and Procedures for Developing Grounded Theory*. Los Angeles; London; New Delhi; Singapore; Washington DC; Boston : SAGE, 2008, S. 143–158.

[6] Juliet M. Corbin und Anselm L. Strauss. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Los Angeles; London; New Delhi; Singapore; Washington DC; Boston : SAGE, 2015.

[7] Lea Genau. *Die verschiedenen Interviewformen erklärt*. Scribbr B.V. URL: https://www.scribbr.de/methodik/interviewformen/.

[8] Bibliographisches Institut GmbH. *Interview, das*. Duden. URL: https://www.duden.de/rechtschreibung/Interview.

[9] *Homepage MAXQDA*. VERBI Software – Consult -Sozialforschung GmbH. URL: https://www.maxqda.de/.

[10] IEEE. „IEEE Standard Glossary of Software Engineering Terminology“. In: *IEEE Std 610.12-1990* (Dez. 1990), S. 62.

[11] Colin Neill Kassab Mohamad und Phillip Laplante. „State of Practice in Requirements Engineering: Contemporary Data..“ In: *Innovations in systems and software engineering* 10.4 (2014), S. 235–241.

[12] Prof. Dr. Daniel Mendez. *NaPiRE Data Visualisation*. Naming the Pain in Requirements Engineering (NaPiRE). URL: http://www.re-survey.org/#/explore.

[13] *OBS Studio*. URL: https://obsproject.com/de.

[14] Cristina et al. Palomares. „The State-of-Practice in Requirements Elicitation: An Extended Interview Study at 12 Companies..“ In: *Requirements engineering* 26.2 (2021), S. 273–299.

[15] Charlene Raddatz. *Qualitatives Interview mit Entwickler 1.0, UDevOps*. . Apr. 2021. URL: https://git.imp.fu-berlin.de/charler98/bachelorsthesis_sources/-/blob/main/Interviewaufnahmen/UDevOps/Entwickler/UDevOpsEntwickler1.mp3.

[16] Charlene Raddatz. *Qualitatives Interview mit Entwickler 2.0, UDevOps*. . Apr. 2021. URL: https://git.imp.fu-berlin.de/charler98/bachelorsthesis_sources/-/blob/main/Interviewaufnahmen/UDevOps/Entwickler/UDevOpsEntwickler2.mp3.

[17] Charlene Raddatz. *Qualitatives Interview mit Entwickler, UVersorgung*. Aug. 2021. URL: https://git.imp.fu-berlin.de/charler98/bachelorsthesis_sources/-/blob/main/Interviewaufnahmen/UVersorgung/Entwickler/UVersorgungEntwickler.mp3.

[18] Charlene Raddatz. *Qualitatives Interview mit Geschäftsleiter, UAutotest*. Mai 2021. URL: https://git.imp.fu-berlin.de/charler98/bachelorsthesis_sources/-/blob/main/Interviewaufnahmen/UAutoTest/UAutoTestGeschaeftsleiter.mp3.

[19] Charlene Raddatz. *Qualitatives Interview mit Marketing-Verantwortlichem, USoftware*. . Apr. 2021. URL: https://git.imp.fu-berlin.de/charler98/bachelorsthesis_sources/-/blob/main/Interviewaufnahmen/USoftware/USoftwareMarketingverantwortlicher.mp3.

[20] Charlene Raddatz. *Qualitatives Interview mit Projektleiter, UVersorgung*. Aug. 2021. URL: https://git.imp.fu-berlin.de/charler98/bachelorsthesis_sources/-/blob/main/Interviewaufnahmen/UVersorgung/Projektleiter/UVersorgungProjektleiter.mp3.

[21] Charlene Raddatz. *Qualitatives Interview mit Systemingenieur, UGebäudeAuto*. Apr. 2021. URL: https://git.imp.fu-berlin.de/charler98/bachelorsthesis_sources/-/blob/main/Interviewaufnahmen/UGebaeudeAuto/UGebaeudeAutoSystemingenieur.mp3.

[22] Charlene Raddatz. *Qualitatives Interview mit Teamlead I, UDevOps*. . Mai 2021. URL: https://git.imp.fu-berlin.de/charler98/bachelorsthesis_sources/-/blob/main/Interviewaufnahmen/UDevOps/TeamleadI/UDevOpsTeamlead1.mp3.

[23] Charlene Raddatz. *Qualitatives Interview mit Teamlead II, UDevOps*. . Juni 2021. URL: https://git.imp.fu-berlin.de/charler98/bachelorsthesis_sources/-/blob/main/Interviewaufnahmen/UDevOps/TeamleadII/UDevOpsTeamleadII.mp3.

[24] Lichtenberg K. Rubin KS. *Essential Scrum: Umfassendes Scrum-Wissen Aus Der Praxis.* MITP, 2014, S. 33–35.

[25] Lichtenberg K. Rubin KS. *Essential Scrum: Umfassendes Scrum-Wissen Aus Der Praxis.* MITP, 2014, S. 50–61.

[26] A. Strauss und J Corbin. *Grounded Theory Methodology: An Overview*. In N. Denzin und Y. Lincoln Handbook of Qualitative Research. 1st ed, 1994, S. 273–285.

[27] Cathy Urquhart. *Grounded Theory for Qualitative Research: a Practical Guide / Cathy Urquhart*. London: SAGE, 2013.

A Anhang

Grobentwurf des Interviews für die Bachelorarbeit zum Thema "Probleme im RE-Prozess" von Charlene Raddatz

1. Kurze Vorstellung:
   a. Welche Position besetzen Sie im Unternehmen?
   b. Was sind Ihre Aufgaben?
   c. Wie groß ist Ihr Team und wie ist es zusammengesetzt?
2. Wie arbeiten Sie in Ihrem Team vor Beginn der eigentlichen Durchführung eines Projektes? Wie werden die Anforderungen erstellt?
3. Welche Herausforderungen treten in der Anforderungserhebung bei Ihnen auf?
4. Welche sind besonders schwerwiegend und warum? In welchem Zusammenhang sind sie aufgetreten?
5. Wie hat Ihr Team diese überwunden?
6. Wie werden den genannten Schwierigkeiten entgegengewirkt?
7. Wie wird vorgegangen, wenn sich Ziele oder Anforderungen stark ändern?

1

Abbildung 1: Erster Entwurf des Interviews

Entwurf des Interviews für die Bachelorarbeit zum Thema "Probleme im RE-Prozess" von Charlene Raddatz (Teamlead)

1. Eine kurze Vorstellung:
   a. In welchem Projekt/in welchen Projekten arbeiten Sie? Wenn es mehrere sind: Gibt es Wechselwirkungen unter den Projekten oder wirkt sich das auf Ihre Arbeit aus?
   b. Welche Position besetzen Sie im Unternehmen?
   c. Was sind Ihre Aufgaben?
   d. Wie groß ist Ihr Team und wie ist es zusammengesetzt?
   e. Wie arbeitet Ihr Team allgemein? Wir wird vorgegangen? Wie läuft die Kommunikation? Gibt es Iterationen? *Gerne mit Beispielen*
   f. Haben Sie firmeninterne oder externe Kunden?
2. Wie arbeiten Sie in Ihrem Team vor Beginn der eigentlichen Implementierung? Wie werden Anforderungen erhoben? Wer arbeitet daran alles mit? Welche Fachgebiete/Erfahrungen haben die beteiligten Personen? Wie wird vorgegangen? Wer spricht mit wem? Wer schreibt die erarbeiteten Anforderungen letztendlich auf? Wird bei Bedarf nachkorrigiert? Wenn ja, wer nimmt das vor und wie wird es kommuniziert? (*Anforderungserhebung mit dem Kunden möglichst genau beschreiben*)
3. Wie sind Anforderungen zu Beginn bei Ihnen formuliert? Wie sind sie im laufenden Projekt formuliert? Welche Formulierung wirft die meisten Probleme beziehungsweise die meiste Nacharbeit auf? Warum? Werden theoretische Modelle oder Konzepte befolgt? Bringen diese Modelle aus Ihrer Sicht etwas?
4. Wie arbeiten Ihre Kunden (im Gegensatz zu Ihnen)? Wasserfall, iterativ, agil? Wie läuft die Kommunikation mit den Kunden? Gab es hier schon Konflikte? *(Gerne Beispiele)*
5. Welche Herausforderungen treten (in der Anforderungserhebung oder im Laufe des Projektes mit den Anforderungen) bei Ihnen häufig auf?
6. Welche sind besonders schwerwiegend? Warum das? Und in welchem Zusammenhang sind sie aufgetreten? (*Möglichst genau beschreiben, gerne ein reales Beispiel ausführlich erläutern*)
7. Wie haben Sie oder Ihr Team diese überwunden? (*Konkretes Beispiel, am besten das zeitlich Letzte*)
8. Hat die Distanz zwischen technischer und Kunden-Perspektive schon mal zu Problemen in der Anforderungserhebung geführt? Welche Herangehensweise wirft mehr/weniger Probleme auf?
9. Wie werden den genannten Schwierigkeiten entgegengewirkt?

1

   a. Wenn iterative Methoden verwendet werden, warum treten die Probleme immer noch auf? Woran liegt das Ihrer Meinung nach? (*Agil hier bei Bedarf definieren*)
   b. Wenn keine iterativen Methoden angewandt werden, warum können diese nicht verwendet werden?
10. Wie werden Änderungen in den Anforderungen an den Kunden/an das Team kommuniziert? Welche Reaktionen sind zu erwarten?
11. Wie wird vorgegangen, wenn sich Ziele oder Anforderungen stark ändern? Wird bei neuen Erkenntnissen (z.B. moving targets) bei Bedarf auch die Software neu aufgesetzt? Wie ist dazu die Bereitschaft im Team? Oder wird versucht inkrementell zu erweitern?
12. Hatten Sie diese typischen Schwierigkeiten? Wie sind sie entstanden und wie werden sie bei Auftreten ausgeräumt? (*Nur falls vorher keine Probleme genannt*)
   a. Unvollständige oder zuvor versteckte Anforderungen (Incomplete or hidden requirements)
   b. Unterbestimmte Anforderungen, zu locker, nicht genau genug (Underspecified requirements)
   c. Sich verändernde Ziele (Moving targets)
13. Wenn nein, was wirkt diesen in Ihrem Team entgegen?

2

Abbildung 2: Finaler Entwurf des Interviews

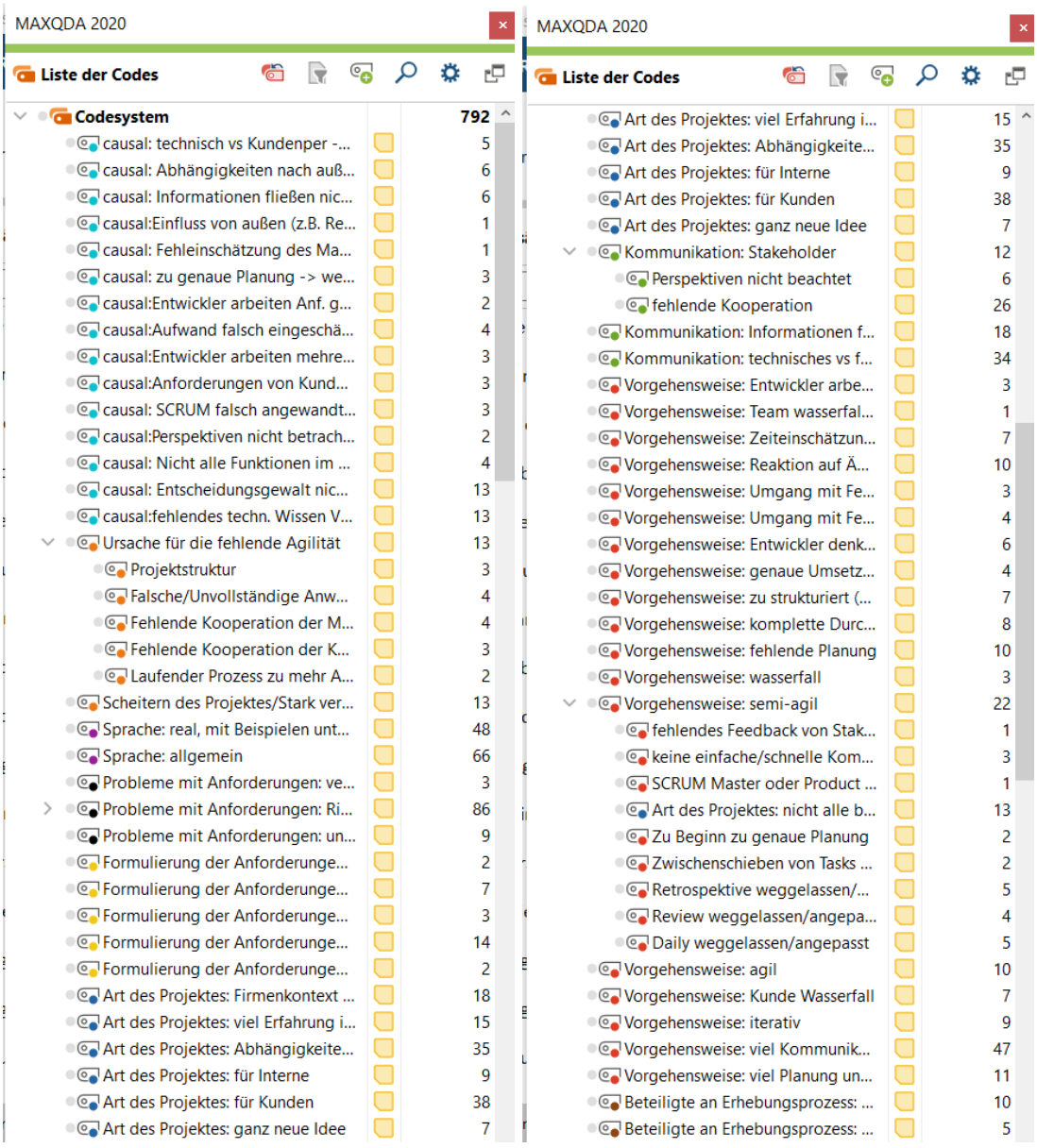

Abbildung 3: Liste der verwendeten Codes