# Unterstützung der Balance von agiler und plan-basierter Entwicklung durch ein Entwicklungsframework

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

**Doktor-Ingenieur**
(abgekürzt: Dr. Ing.)

genehmigte Dissertation von Herrn

**M. Sc. Nils Prenner**
geboren am 28.04.1992 in Hannover

**2023**

1. Referent: Prof. Dr. Kurt Schneider
2. Referent: Prof. Dr. Jan Rellermeyer
Vorsitzender der Prüfungskommission: Prof. Dr. Bernardo Wagner

Tag der Promotion: 04.05.2023

# Zusammenfassung

Softwarefirmen benötigen Stabilität und Flexibilität für ihre Projekte gleichermaßen und kombinieren daher häufig agile und plan-basierte Ansätze zu hybriden Ansätzen. Plan-basierte Ansätze unterstützen Stabilität, während agile Ansätze erlauben, flexibel zu bleiben. Um beide Ziele zu erreichen, müssen Projektteams beide Ansätze ausbalancieren und der Entwicklungsprozess muss an ihren Kontext angepasst werden. Dies bedeutet eine Schwierigkeit für Firmen. Es existiert eine Forschungslücke hinsichtlich eines strukturierten Vorgehens für die Kombination von beiden Ansätzen. Definierte allgemeine Entwicklungsframeworks, wie Scrum oder das V-Modell, unterstützen Firmen dabei, bestimmte Ziele in ihrer Entwicklung zu erreichen. Daher wird in dieser Arbeit ein allgemeines Entwicklungsframework für die Balance von beiden Ansätzen entwickelt. Für die Entwicklung dieses allgemeinen Frameworks muss erforscht werden, (1) an welchen Stellen im Entwicklungsansatz eine Balance hergestellt werden muss, (2) welche Ziele die Balance beeinflussen und (3) wie eine hergestellte Balance in ein Framework überführt werden kann. Zur Erforschung dieser Aspekte werden eine zweifache Mappingstudie und eine Interviewstudie durchgeführt. Es werden fünf Aktivitäten identifiziert, für die agile und plan-basierte Entwicklung balanciert werden muss und 13 agile und 12 plan-basierte Ziele, die die Balance beeinflussen. Im Rahmen der Interviewstudie wird untersucht, wie Firmen eine Balance in diesen Aktivitäten herstellen. Dabei zeigt sich, dass Projektteams die beiden Grundbedürfnisse nach Stabilität und Flexibilität abwägen. Um ein allgemeines Framework zu schaffen, werden hybride Entwicklungsprozesse von Firmen untersucht. Diese Analyse ergibt drei generelle Vorgehensmodelle, nach denen Firmen in hybriden Entwicklungsprozessen arbeiten. Die gesammelten Erkenntnisse werden verwendet, um ein allgemeine Entwicklungsframework zu entwickeln. Das Framework ermöglicht es Firmen eine Balance von Stabilität und Flexibilität zu nutzen, indem Aktivitäten sowohl zu Beginn eines Projektes als auch kontinuierlich während des Projektes durchgeführt werden. Das Framework gibt Empfehlungen an Firmen darüber, inwieweit sie Aktivitäten zu Beginn eines Projektes durchführen müssen, um genug Stabilität zu erreichen und wann sie in eine kontinuierliche Durchführung von Aktivitäten übergehen müssen, um ausreichend flexibel zu sein. Zusätzlich werden auf Basis eines risikobasierten Konzeptes, unterschiedliche Handlungsalternativen definiert, um die Balance an den spezifischen Kontext einer Firma anzupassen. Das Framework und die Handlungsalternativen werden anschließend durch Experteninterviews validiert.

**Schlagwörter:** Agile Softwareentwicklung, Plan-basierte Softwareentwicklung, Hybride Softwareentwicklung, Balance, Entwicklungsframework

# Abstract

Software companies need simultaneously stability and flexibility for their software projects, therefore, they often combine agile and plan-based approaches to hybrid approaches. Plan-based approaches support stability, while agile approaches support flexibility. To reach both goals, project teams have to balance both approaches and adapt their development process to their context. Companies have difficulties with both. There is a research gap regarding a structured approach for the combination of agile and plan-based development. Defined general development frameworks, such as Scrum or the V-model, support companies in achieving certain goals for their development. Therefore, a general development framework for the balance of both approaches is developed in this thesis. For the development of this general framework it is necessary to investigate (1) at which points in the development approach a balance has to be established, (2) which goals influence the balance, and (3) how an established balance can be transferred into an actual development framework.

To investigate these aspects a double mapping study and an interview study with 15 participants is conducted. Five activities are identified that project teams have to balance and 13 goals for agile and 12 goals for plan-based development that influence the balance in hybrid approaches. How companies create a balance in these activities, is investigated by conducting an interview study. The results show that project teams weigh the two basic needs of stability and flexibility against each other. To create a general framework, the hybrid development processes of companies are examined. The analysis results in three general process models that are used by companies in hybrid development processes. Based on the gathered knowledge, a general framework is created.

The framework enables companies to use a balance of stability and flexibility. For that, the frameworks enables companies to use the same activities at the beginning of a project and continuously during the project. The framework provides recommendations for companies regarding to what degree they have to use the activities at the beginning of a project to reach enough stability. Further, it provides recommendations regarding when companies should move to a continuous implementation of activities to be sufficiently flexible. Additionally, on the basis of a risk-based concept, different alternatives courses of action are defined in order to adapt the balance to the specific context of a company. The framework and the alternative courses of action are then validated by expert interviews.

**Keywords:** agile software development, plan-based software development, hybrid software development, balance, development framework

# Danksagung

Diese Arbeit wäre ohne die unglaubliche Unterstützung von vielen anderen Menschen nicht möglich gewesen. Ich danke allen, die mich auf dieser Reise begleitet und mit Rat und Zuspruch unterstützt haben.

Besonders danken möchte ich Prof. Dr. Kurt Schneider. Mein Traum war es in der Forschung zu arbeiten. Danke, dass Sie diesen Traum wahr gemacht haben. Ich habe von Ihnen viel über die Forschung und das Software Engineering gelernt. Sie hatten immer Zeit, um Feedback oder Rat zu geben. Danke für diese unermüdliche Unterstützung und Ihre Zeit.
Ich danke Prof. Dr. Jan Rellermeyer und Prof. Dr. Bernardo Wagner, dass sie sich die Zeit nehmen, um diese Arbeit zu unterstützen.

Ich bedanke mich bei meinen Kollegen an der Leibniz Universität Hannover für die schöne Zeit. Danke für euren Rat und Wissen, mit denen ihr mich unterstützt habt. Besonders bedanken möchte ich mich bei meinen Bürokolleginnen Larissa Chazette und Maike Ahrens. Danke für die vielen Gespräche und die gegenseitige Motivation und Unterstützung in den Jahren. Ihr habt mich immer daran erinnert, wie gerne ich forsche. Dafür bin ich euch sehr dankbar. Dazu gehört auch Dr. Carolin Unger-Windeler, der ich für die vielen Gespräche über hybride Softwareentwicklung danken möchte. Diese haben mir immer sehr viel Freude bereitet und mein Wissen sehr bereichert. Besonders möchte ich auch Dr. Jil Klünder danken, dafür dass du immer ein offenes Ohr hattest und ich mich immer auf deine Hilfe verlassen konnte.

Ich danke allen Studienteilnehmern, dass sie sich die Zeit genommen haben, um mich mit ihrem Wissen zu unterstützen. Besonders möchte ich Björn Balfanz danken, der mich bei den Studien mit der Vermittlung von Teilnehmern unterstützt hat. Die Gespräche mit dir waren immer eine große Motivationsquelle und haben diese Arbeit mit vorangetrieben.

Ich möchte mich auch bei meinen Geschwistern und besonders meinen Eltern Bettina und Christian Prenner bedanken. Ich bin euch unendlich dankbar für euren Zuspruch und eure Unterstützung! Danke, dass ihr immer an mich geglaubt habt.

# Inhaltsverzeichnis

# 1

# Einleitung

Softwarefirmen streben beständig nach dem besten Weg, um Software zu entwickeln, da sie, um auf dem Markt gegen andere Firmen bestehen zu können, neue Features schnell und in einer hohen Qualität entwickeln müssen [1]. Zudem müssen sie in der Lage sein, schnell auf Veränderungen im Markt reagieren zu können. Die Entwicklung von agilen Methoden entstand aus dem Bedürfnis nach mehr Flexibilität im Entwicklungsprozess. Mehr und mehr Firmen implementieren agile Methoden und Praktiken [2]. Softwarefirmen benötigen jedoch auch Struktur und Stabilität in der Entwicklung, um zum Beispiel mit großen Projektteams umzugehen [3, 4]. Struktur und Stabilität sind Eigenschaften, die in der Literatur mit plan-basierten Entwicklungsansätzen verbunden werden [5]. Um Struktur und Stabilität während eines Projektes zu haben und gleichzeitig flexibel zu sein, kombinieren Firmen, wie aktuelle Forschung zeigt, häufig agile und plan-basierte Methoden zu hybride Entwicklungsansätzen [4, 6]. Die Daten der *Helena Studie*[1] zeigen, dass 66% der berichteten Fälle agile und plan-basierte Methoden und Praktiken in ihrem Entwicklungsprozess kombinieren [6, 7]. Tell et al. [2] leiten aus den Daten ab, dass hybride Entwicklungsprozesse den Stand der Technik darstellen.

[1] Hybrid development approaches in software system development: www.helenastudy.wordpress.com

## 1.1 Motivation und Ziel

Die Bildung von hybriden Entwicklungsprozessen ist für die Firmen mit Schwierigkeiten verbunden [8]. Firmen müssen zum Beispiel ihren Kontext analysieren, um eine Kombination von agiler und plan-basierter Entwicklung passend für ihre Entwicklung zu finden. Die Analyse ihres Kontextes stellt auf Grund der hohen Anzahl an Einflussfaktoren eine Schwierigkeit für die Firmen dar [8–11]. Zusätzlich müssen Firmen, basierend auf ihren Kontext, die geeigneten Methoden wählen. Auf Grund der hohen Anzahl an unterschiedlichen Methoden und dem fehlenden Wissen über Vor- und Nachteile von Methoden, stellt dies ebenfalls eine Herausforderung für Firmen dar [8, 12].

Bei der Kombination von agiler und plan-basierter Entwicklung kommt zusätzlich hinzu, dass diese beiden Vorgehen balanciert werden müssen [3, 5]. Zum Beispiel müssen Firmen darüber entscheiden wie viel Anforderungsanalyse sie zu Beginn eines Projektes durchführen sollten [3]. Diese Balance ist schwierig für Firmen [3].

Diese Schwierigkeiten sind möglicherweise der Grund warum, viele hybride Prozesse in einem evolutionären Prozess durch Erfahrung gebildet werden [1]. Erfahrung kann jedoch nur bei einem existierenden Entwicklungsprozess helfen. Einen neuen hybriden Entwicklungsprozess zu bilden, ist hingegen schwierig [13]. Küpper et al. [12] identifizierten das Fehlen eines systematischen Vorgehens bei der Bildung von hybriden Entwicklungsprozessen. Somit fehlt Firmen eine Unterstützung bei der Kombination von agiler und plan-basierter Entwicklung.

Definierte allgemeine Entwicklungsframeworks, wie Scrum oder das V-Modell, unterstützen Firmen dabei, bestimmte Ziele, wie mehr Flexibilität oder Planbarkeit, in ihrer Entwicklung zu erreichen [14–16]. Entwicklungsframeworks ermöglicht es Firmen, jeweils für Projekte ein gleiches Vorgehen zu nutzen, das nur an einen Kontext angepasst werden muss. Ein solches gleiches Vorgehen fehlt für die Kombination von agiler und plan-basierter Entwicklung. Um Firmen bei der Kombination von agiler und plan-basierter Entwicklung zu unterstützen, wird in dieser Arbeit ein allgemeines Entwicklungsframework geschaffen, das agile und plan-basierte Entwicklung balanciert. Dieses hat den Zweck, dass Firmen dadurch eine Balance von agiler und plan-basierter Entwicklung nutzen können, ohne einen evolutionären Prozess nutzen zu müssen. Ein Entwicklungsframework muss immer an den jeweiligen Entwicklungskontext angepasst werden, da es sonst ein Projektrisiko darstellt [10, 17, 18]. Daher muss ein allgemeines Framework anpassbar an einen Kontext sein und sollte passende Empfehlungen zu Änderungen des Frameworks enthalten. Zusammengefasst wird für diese Arbeit daher das folgende Ziel formuliert:

Das Ziel dieser Arbeit ist die Schaffung eines allgemeines Entwicklungsframeworks für die Balance von agiler und plan-basierter Entwicklung, das an einen Projektkontext anpassbar ist und passende Empfehlungen für eine Anpassung des Frameworks an einen Projektkontext enthält.

Für die Bildung eines solchen Frameworks muss untersucht werden, wie Firmen agile und plan-basierte Entwicklung kombinieren und welche Anpassungen in welchen Kontexten getroffen werden sollten. Kuhrmann et al. [19] haben identifiziert, dass die Forschung zu hybriden Entwicklungsansätzen meist auf Fallstudien und Erfahrungsberichten beruht. Es fehlen jedoch fallübergreifende Forschungen, die verschiedene Erkenntnisse miteinander verbinden. Des Weiteren haben Kalus et al. [10] identifiziert, dass es eine Forschungslücke darüber gibt, wie sich bestimmte Kontextfaktoren auf einen Entwicklungsprozess auswirken. Beide Forschungslücken werden in dieser Arbeit für die Entwicklung eines allgemeinen Frameworks für die Balance von agiler und plan-basierter Entwicklung adressiert.

## 1.2 Forschungsvorgehen

Das Ziel dieser Arbeit ist es ein Framework für die Balance von agiler und plan-basierter Entwicklung zu schaffen, das zusätzlich passende Empfehlungen für eine Anpassung an einen bestimmten Kontext liefert. Design Science ist ein Forschungsvorgehen, um ein Artefakt für einen bestimmten Zweck zu schaffen [20]. Um ein passendes Framework zu entwickeln, wird daher Design Science in dieser Dissertation als Forschungsvorgehen genutzt. Design Science teilt sich in drei Zyklen auf [21]:

**Relevance Cycle** Im Relevance Cycle werden Probleme und Schwierigkeiten in der Anwendungsdomäne untersucht. Diese besteht aus Menschen, Organisationssystemen und technischen Systemen. Die Anwendundsdomäne in dieser Arbeit sind Firmen, die agile und plan-basierte Entwicklung kombinieren, um damit gewisse Ziele zu erreichen.

**Design Cylce** Im Design Cycle wird das Artefakt geschaffen, das das Ergebnis von Design Science darstellt. Dabei wird in beständigen Iterationen zwischen Design und Evaluation iteriert, um das Artefakt beständig zu verbessern.

**Rigor Cycle** Im Rigor Cycle wird sowohl bestehendes Wissen im Forschungsbereich untersucht, wie auch Neues geschaffen. Dies geschieht um die Neuheit des gebildeten Artefaktes sicherzustellen und Designentscheidungen auf der Basis von wissenschaftlichen Erkenntnis-

sen zu treffen. In dieser Arbeit wird der Rigor Cycle angewendet, um die oben beschriebenen Forschungslücken zu schließen.

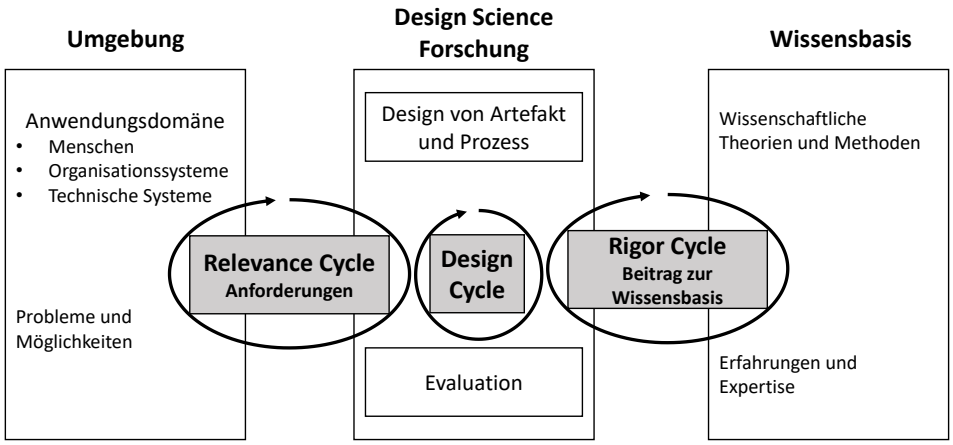


**Abbildung 1.1:** Zusammenhang des Relevance, Design und Rigor Cycle in Design Science nach Hevner [21]

## 1.3 Struktur der Arbeit

In Abbildung 1.2 ist die Struktur der Arbeit dargestellt. Die Darstellung folgt den drei Zyklen von Design Science. Auf der linken Seite ist der Relevance Cycle. Dieser wird von den Kapiteln 3 und 4 abgedeckt. Das Ergebnis dieser Kapitel ist es, dass kein strukturiertes Vorgehen für die Kombination von agiler und plan-basierter Entwicklung gibt und dass Firmen Schwierigkeiten bei der Balance von beiden Ansätzen haben.

Auf der rechten Seite der Abbildung ist der Rigor Cycle. Dieser wird von den Kapiteln 2, 4 und 5 abgedeckt. In Kapitel 2 wird eine Definition für Entwicklungsansätze gegeben, die agile und plan-basierte Entwicklung miteinander kombinieren und wichtige Begriffe erklärt. In Kapitel 4 werden Entwicklungsprozesse analysiert, die Firmen für die Kombination von agiler und plan-basierter Entwicklung nutzen. Daraus werden allgemeine Vorgehen abgeleitet. In Kapitel 5 wird im Rahmen einer Interviewstudie analysiert, wie Projektteams eine Balance von agiler und plan-basierter Entwicklung erreichen. Diese Kapitel erweitern die Wissensbasis und die gesammelten Erkenntnisse fließen in den Design Cycle ein.

Der Design Cycle wird von den Kapiteln 6, 7 und 8 abgedeckt. In Kapitel 6 stelle ich ein allgemeines Konzept für die Balance von agiler und plan-basierter Entwicklung vor, dass auf der Abwägung von Risiken basiert. In Kapitel 7 beschreibe ich die Risiken, die bei Balance von agiler und plan-basierter Entwicklung gegeneinander abgewogen werden müssen und disku-

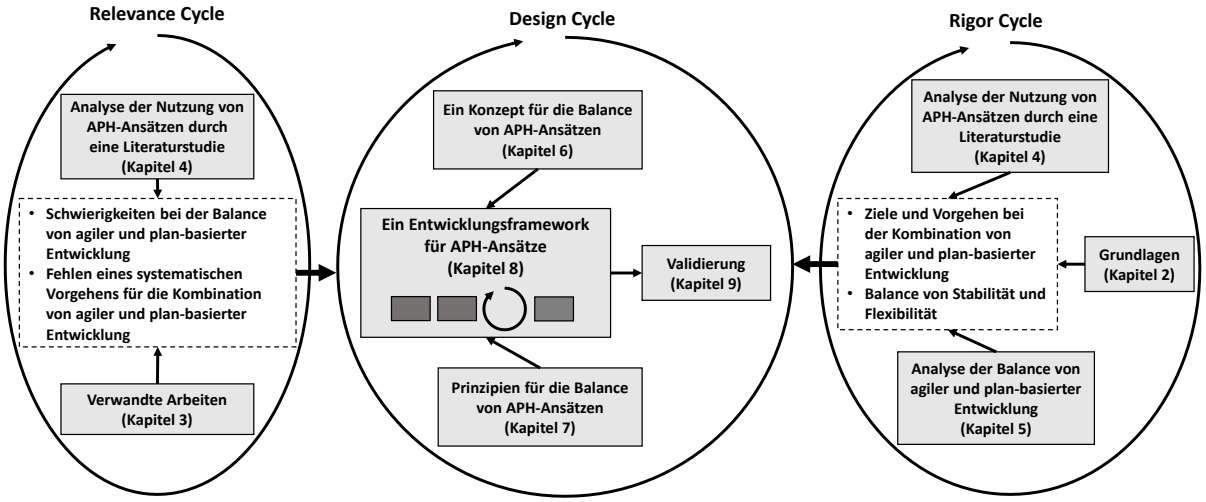


**Abbildung 1.2:** Zusammenhang der Kapitel in dieser Dissertation

tiere die resultierenden Balancen. Aus den beiden Konzepten und der erstellten Wissensbasis, stelle ich in Kapitel 8 ein Framework vor, durch das die beschriebenen Balancen aus Kapitel 7 implementiert werden können. Zusätzlich beschreibe ich, auf welche Weise das Framework an verschiedene Kontexte angepasst werden sollte.

Zu Design Science gehört die Validierung des erstellten Artefaktes. In Kapitel 9 führe ich im Rahmen von Experteninterviews eine Validierung des Frameworks und der Anpassungen an einen Kontext durch. Am Ende der Arbeit, diskutiere ich in Kapitel 10 Limitationen der Arbeit und zukünftige Forschungsrichtungen, die sich unter anderem aus der Validierung des Frameworks ergeben.

# 2

# Grundlagen

Das Ziel dieser Arbeit ist es, ein Framework für die Kombination von agiler und plan-basierter Entwicklung zu schaffen. Daher werden in diesem Kapitel die Grundlagen für plan-basierte, agile und hybride Entwicklung erklärt.

## 2.1 Plan-basierte Entwicklung

In den 1960er Jahren gewann die Entwicklung von Software eine immer größerer Bedeutung. Im Gegensatz zur Ingenieurswelt gab es jedoch kein strukturiertes Vorgehen bei der Entwicklung [5]. Stattdessen bestand die Entwicklung aus meist den zwei Schritten Analyse und Implementierung [15]. Diese beiden Schritte reichen jedoch nicht für die Entwicklung von komplexen und großen Softwaresystemen aus [15]. In dieser Zeit dauerte die Entwicklung von Software oft länger als erwartet, war über dem Budget und resultierte in Software von schlechter Qualität [5]. In der Softwareentwicklung wurde daher in dieser Zeit nach Wegen gesucht, um mehr Struktur und Disziplin in die Softwareentwicklung zu bringen und sie damit der Ingenieurswelt ähnlicher zu machen [5].

In 1970 stellte Royce [15] ein Entwicklungsmodell vor, das den Entwicklungsprozess in Aktivitäten einteilt. Die Aktivitäten dieses Entwicklungsmodells sind in Abbildung 2.1 dargestellt. Die Aktivitäten ermöglichen es, die Entwicklung einer Software kontrolliert von der Erhebung der

Anforderungen, über die Architekturdefinition und der Implementierung hin zur Durchführung der Tests zu führen. Da die Entwicklung diese Aktivitäten durchläuft und die Arbeitsprodukte aus jeder Aktivität an die nächste weitergereicht werden, hat dieses Modell den Namen Wasserfallmodell bekommen. Royce [15] gibt an, dass wenn ein Projektteam Probleme in einer Aktivität feststellt, es in eine frühere Aktivität zurückspringen kann, um diese Probleme zu beheben. Wenn zum Beispiel während der Programmieraktivität Fehler in der Architektur festgestellt werden, kann das Projektteam in die Aktivität des Programmdesigns zurückgehen. Jede dieser Aktivitäten wird durch einen Dokumentations- und Evaluationsprozess begleitet, um die Qualität in jedem Schritt sicherzustellen und Fehler frühzeitig zu entdecken [22, 23]. Dies stellt ein wichtiges Prinzip in plan-basierter Entwicklung dar und unterstützt dabei, dass Projektteams nicht in eine frühere Aktivität zurückkehren müssen [5, 15].

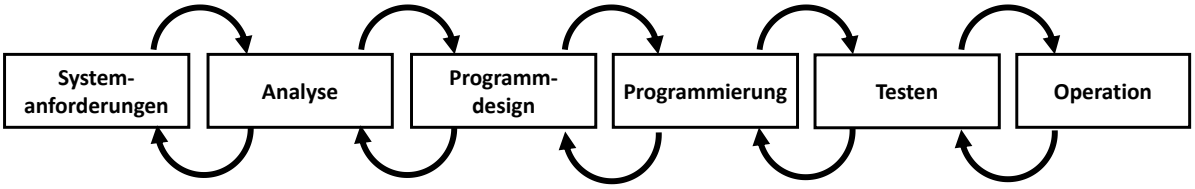


**Abbildung 2.1:** Die Aktivitäten des Wasserfallmodells nach Royce [15]

Das Ziel von plan-basierter Entwicklung ist es, eine Standardisierung und Wiederholbarkeit von Softwareentwicklung zu erreichen, um die Entwicklung von Software berechenbar und messbar zu machen [5]. Dies ist ein Kernprinzip von plan-basierter Entwicklung. Die Standardisierung und Wiederholbarkeit wird in plan-basierter Entwicklung durch die Definition von genauen Prozessen erreicht [5]. Diese Prozesse beinhalten zum Beispiel genaue Pläne, Rollen-und Verantwortungsverteilung und die Beschreibung von Arbeitsergebnissen. Die Personen, die diese Prozesse anwenden, sind in der Anwendung der Prozesse geschult. Zusätzlich überwacht eine zentrale Stelle die Durchführung der Prozesse [5].

In plan-basierter Entwicklung wird das gesamte Projekt zu Beginn geplant. Dies bedeutet, dass festgelegt wird, wann genau welche Aktivitäten durchgeführt werden und wie lange diese dauern. Um diese Pläne festlegen zu können, werden alle Anforderungen bereits zu Beginn des Projektes erhoben. Basierend darauf kann die Entwicklung des gesamten Projektes geplant werden [5]. Durch die Standardisierung und der genauen Planbarkeit ist die plan-basierte Entwicklung besonders für große Projektteams und sicherheitskritische Anwendungen geeignet. Das Wasserfallmodell ist immer noch das am meisten verbreitete Modell für plan-basierte Entwicklung, obwohl mittlerweile weitere Modelle geschaffen wurden, wie das V-Modell oder RUP [2].

Plan-basierte Entwicklung bringt, einige Vorteile mit sich. Es hat jedoch auch Nachteile.

Plan-basierte Entwicklung setzt darauf, alle Anforderungen zu Beginn eines Projektes zu erheben [24]. Das Problem dabei ist, dass Kunden zu Beginn eines Projektes oftmals nicht genau wissen, welche Anwendung sie genau benötigen. Projektteams sind daher häufig mit Anforderungsänderungen konfrontiert [16]. Durch die genaue Planung zu Beginn eines Projektes ist es jedoch schwierig, Änderungen in plan-basierten Ansätzen einzubringen [5]. Des Weiteren bekommt der Kunde in plan-basierten Ansätzen die Software erst zu Gesicht, wenn diese fertig und ausgeliefert ist. Dies bedeutet, dass plan-basierte Entwicklung spät zu Resultaten führt und kein Feedback des Kunden während der Entwicklung ermöglicht.

## 2.2 Agile Entwicklung

Um den Schwierigkeiten von plan-basierter Entwicklung zu begegnen, formte sich Mitte der 1990er Jahre die Bewegung von agiler Entwicklung. In dieser Bewegung entwickelten sich viele Methoden und Frameworks, wie Scrum, Extreme Programming oder Feature Driven Development [25]. All diese verschiedenen Methoden sind unter dem Manifest für agile Entwicklung vereinigt. In diesem haben die Pioniere der agilen Entwicklung in 2001 vier Werte und 12 Prinzipien für die agile Entwicklung festgelegt. Die vier Werte sind die Folgenden [26]:

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

Die Verfasser des Manifestes für agile Entwicklung schreiben zu diesen Werten, dass sie die Werte auf der rechten Seite wichtig finden, jedoch die Werte auf der linken Seite höher einschätzen [26]. Für die erste Aussage bedeutet dies, dass in agiler Entwicklung mehr Wert auf den Austausch von Informationen von Angesicht zu Angesicht und auf eine Selbstorganisation der Teams gelegt wird, als auf eine formelle Kommunikation und eine zentrale Koordination von Entwicklern. Für die zweite Aussage bedeutet dies, dass funktionierende Software das wichtigste Fortschrittsmaß in der Entwicklung ist. Für die dritte Aussage bedeutet dies, dass Entwicklerteams regelmäßig Software an den Kunden ausliefern und Feedback einholen. Für die letzte Aussage bedeutet dies, dass Entwicklerteams während der gesamten Laufzeit Anforderungsänderungen zulassen und diese umsetzen.

Agile Entwicklung basiert darauf, dass die Entwicklung in schnellen Zyklen erfolgt [14, 16, 27]. Auf diese Weise können Softwareprojekte empirisch durchgeführt werden. Dies bedeutet, dass die Entwickler in regelmäßigen Abständen ihr Produkt, Prozess und ihren Fortschritt analysieren und basierend darauf Entscheidungen über die Weiterführung des Projektes treffen. In

Abbildung 2.2 ist dieses Empirische Management Modell abgebildet. Zu Beginn einer Iteration findet die Eingabe von Anforderungen an ein Projektteam statt. In einer Iteration implementiert ein Entwicklerteam die Anforderungen und produziert eine Ausgabe. Gleichzeitig existiert ein Kontrollprozess, der den Verlauf einer Iteration und die Ausgabe überprüft. Zum Beispiel wird die Ausgabe dem Kunden präsentiert, der dazu Feedback gibt. Das Feedback und Entscheidungen über die weitere Entwicklung dienen wieder als Eingabe des Kreislaufs. Auf diese Weise findet eine beständige Überprüfung und Anpassung in Softwareprojekten statt [14]. Dadurch unterstützt eine iterative Entwicklung dabei, dass Entwickler näher an den Wünschen ihrer Kunden entwickeln und auf Probleme reagieren können. Die Feedbackzyklen dienen nicht nur dazu, ein Produkt nach den Wünschen eines Kunden zu entwickeln. Entwicklerteams bewerten auf diese Weise beständig ihr Vorgehen und nehmen Verbesserungen vor [26].

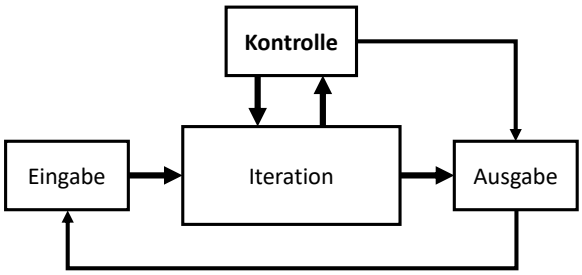


**Abbildung 2.2:** Empirisches Management Model nach Schwaber und Beedle [14]

## 2.3 Hybride Entwicklungsansätze

Firmen nutzen oftmals nicht eine Entwicklungsmethode, sondern kombinieren verschiedene Methoden zu sogenannten hybriden Entwicklungsansätzen [1,2]. Kuhrmann et al. [7, p.2] haben für hybride Entwicklungsansätze die folgende Definition geschaffen:

> *"A hybrid software development approach is any combination of agile and traditional (plan-driven or rich) approaches that an organizational unit adopts and customizes to its own context needs (e.g. application domain, culture, process, project, organizational structure, techniques, technologies, etc.)".*

Diese Definition schließt alle möglichen Methodenkombinationen mit ein, auch solche bei denen ausschließlich agile Methoden miteinander kombiniert werden. Dadurch beschreibt diese Definition den Bereich von hybriden Entwicklungsansätzen sehr weitgefasst und wenig differenziert. Diese Dissertation ist fokussiert auf die Kombination von agiler und plan-basierter Entwicklung, die somit eine Untermenge von hybriden Entwicklungsansätzen darstellt.

Heeager [28] unterscheidet bei der Kombination von agiler und plan-basierter Entwicklung zwei Fälle. Zum einen gibt es kompatible Ansätze, zum anderen gibt es kombinierte Ansätze. Heeagar [28] beschreibt, dass im Falle einer Kombinierbarkeit zwei Dinge zur gleichen Zeit genutzt werden können, ohne dabei eins von beiden zu vernachlässigen. Auf der anderen Seite beschreibt eine Kompatibilität nur, dass zwei Dinge zur gleichen Zeit genutzt werden können. Für hybride Entwicklungsansätze bedeutet dies, dass bei einer Kombination von agiler und plan-basierter Entwicklung beide Ansätze gleichermaßen verfolgt und miteinander integriert werden. Hingegen bedeutet eine Kompatibilität von agiler und plan-basierter Entwicklung, dass zum Beispiel in einem strikten Wasserfallmodell agile Praktiken während der Entwicklung eingesetzt werden können. Dadurch sind jedoch beide Ansätze nicht kombiniert, da die Werte von agiler Entwicklung nicht zur Anwendung kommen, sondern nur einzelne Praktiken.

Diese Arbeit ist fokussiert auf Entwicklungsansätze, in denen Firmen gleichermaßen die Vorteile von agiler und plan-basierter Entwicklung nutzen und einen Entwicklungsprozess schaffen wollen, in dem diese Vorteile kombiniert sind und beide zur Anwendung kommen. Diese Menge der Entwicklungsansätze stellt daher nicht nur eine Untermenge der hybriden Entwicklungsansätze dar, sondern auch eine Untermenge der Ansätze, in denen agile und plan-basierte Entwicklung zusammen gebracht wird. Für diese Untermenge gibt es keinen Begriff und auch keine Definition. Für die Untermenge wird daher in dieser Dissertation der Begriff *Agil-Plan-basierter-Hybrider Entwicklungsansatz* (kurz: APH-Ansatz) und die folgende Definition eingeführt:

**Definition:** Ein **Agil-Plan-basierter-Hybrider Entwicklungsansatz** ist eine Kombination von agiler und plan-basierter Entwicklung, bei der ein Projektteam die Prinzipien von agiler und plan-basierter Entwicklung gleichermaßen nutzt und eine Balance zwischen beiden herstellt und den Entwicklungsprozess ihren eigenen Kontextbedürfnissen anpasst.

## 2.4 Terminologie

Im Folgenden werden einige Begriffe definiert, die in dieser Arbeit verwendet werden:

**Entwicklungsansatz:** Ein Entwicklungsansatz besteht aus den Werte und Prinzipien, die eine Firma für ihre Entwicklung wählt. In dieser Arbeit wird zwischen drei unterschiedlichen Entwicklungsansätzen unterschieden: Der agile, der plan-basierte und der Agil-Plan-basierter-hybride Entwicklungsansatz.

**Entwicklungsframework:** Ein Entwicklungsframework ist ein allgemeines Vorgehensmodell für die Implementierung eines Entwicklungsansatzes durch bestimme Praktiken, Regeln

und Strukturen. Beispiele für Entwicklungsframeworks sind zum Beispiel Scrum, XP, SAFe und das Wasserfallmodell.

**Entwicklungsprozess:** Ein Entwicklungsprozess ist die individualisierte Implementierung eines Entwicklungsansatzes. Firmen passen dafür Entwicklungsframeworks an ihre Kontexbedürfnisse an, kombinieren Entwicklungsframeworks oder schaffen einen eigenen Prozess.

**Tätigkeit:** Eine Tätigkeit beschreibt das Ausführen einer Aufgabe, die ein projektbezogenes Ziel erfüllt. Zum Beispiel sind die Analyse von Stakeholdern, das Sammeln von Anwendungsfällen, die Analyse von nicht-funktionalen Anforderungen alles Tätigkeiten, die das Ziel haben, die Anforderungen in einem Projekt zu verstehen.

**Aktivität:** Eine Aktivität beschreibt eine Sammlung von Tätigkeiten, die dasselbe Ziel haben. Die oben genannten Tätigkeiten haben alle das Ziel, die Anforderungen zu analysieren. Daher sind sie zu der Aktivität Anforderungsanalyse zusammengeschlossen.

# 3

# Verwandte Arbeiten

In der Wissenschaft gibt es bereits einige Arbeiten, die sich mit der Untersuchung und Anpassung von hybriden Entwicklungsansätzen beschäftigt haben. Dieses Kapitel stellt einen Überblick über diese Arbeiten dar und stellt sie in einen Bezug zu dieser Dissertation.

## 3.1 Untersuchung von hybriden Entwicklungsprozessen

West [29] hat in 2011 die These aufgestellt, dass die meisten Firmen bei ihrer Softwareentwicklung eine Mischung aus dem Wasserfall und dem Scrum Framework benutzen, um die Vorteile aus der plan-basierten und der agilen Entwicklung miteinander zu verbinden. Es fehlte jedoch konkrete Forschung darüber, wie weit hybride Ansätze in der Industrie verbreitet sind und welche Methoden miteinander kombiniert werden. Um diese Forschungslücke zu schließen, haben Theocharis et al. [30] eine systematische Literatursuche durchgeführt und untersucht, welche Entwicklungsansätze von Firmen genutzt und kombiniert werden. Ihre Resultate zeigen, dass das Wasserfallmodell und das Scrum Framework zu den meist genutzten Methoden gehören und diese häufig kombiniert werden.

Um die Nutzung von hybriden Entwicklungsmethoden genauer zu untersuchen, wurde die *Helena Studie*[1] geschaffen [7]. Die Studie war an Firmen gerichtet und erfragte deren Nut-

[1] Hybrid development approaches in software system development: www.helenastudy.wordpress.com

zung von Methoden und Praktiken. Die Studie wurde in 2017 veröffentlicht und resultierte in 1,467 Datenpunkten mit 691 kompletten Antworten [7]. Noll and Beecham [6] haben die Daten untersucht und herausgefunden, dass in 66 % der Fälle eine Kombination von agiler und plan-basierter Entwicklung genutzt wird.

Klünder et al. [1] haben herausgefunden, dass Scrum, Iterative Development, Kanban, Waterfall und DevOps am häufigsten in den Fällen genutzt werden, um Methodenkombinationen zu formen. Sie haben ebenfalls die Gründe untersucht, warum Firmen unterschiedliche Methodenkombinationen nutzen. Die häufigsten Gründe sind eine Steigerung der Produktivität, eine Steigerung der Produktqualität und eine Verbesserung der Planung und Schätzung. Die Helena Studie untersucht jedoch nicht, welche Gründe die Firmen für die Nutzung von einzelnen Methoden oder Praktiken haben. Dies bedarf einer genaueren Untersuchung, die in dieser Dissertation durchgeführt wird.

Tell et al. [2] haben ebenfalls die Daten der Helena Studie analysiert. Sie fanden keine Abhängigkeiten zwischen der Nutzung von verschiedenen Methodenkombinationen und der Firmengröße und deren Geschäftsbereich. Stattdessen haben sie eine Reihe von Kernpraktiken gefunden, die konsistent in den Fällen genutzt werden. Diese Kernpraktiken enthalten automatisierte Unit Tests, Codereviews, Programmierstandards, Prototyping und Release Planung. Des Weiteren argumentieren sie, dass hybride Entwicklungsansätze den Stand der Praxis für Firmen darstellen. Sie bestätigen die Annahme von West [29], dass eine große Anzahl an Firmen eine ”Wasser-Scrum-Fallähnlichen Prozess folgen. Sie bestätigen ebenfalls die Aussage von West [29], dass die Zukunft von Softwareentwicklung in der Wahl der richtigen Entwicklungsmethoden liegt und nicht in der Wahl einer konkreten Methode.

In einer explorativen Faktoranalyse haben Klünder et al. [13] den Einfluss von verschiedenen Faktoren, wie Firmengröße oder Geschäftsbereich, auf die Wahl von Praktiken und Methoden untersucht. Sie haben wenige Kontextfaktoren identifiziert und folgern, dass es andere Einflüsse bei der Wahl von Methoden geben muss.

Klünder et al. [31] haben untersucht, welche Methoden und Praktiken für welche Projektaktivität eingesetzt werden. Ihre Resultate zeigen, dass die Firmen häufig Methoden und Praktiken nicht gemäß deren Bestimmung einsetzen und von der Methodenbeschreibung abweichen. Sie diskutieren, dass die Firmen sich bewusst darüber sein müssen, wenn sie von Methodenbeschreibungen abweichen. Zusätzlich sollten sie sich bewusst sein, warum und wie sie abweichen. Bevor eine Firma eine Methode anwendet, muss sie die Gründe für die Nutzung dieser Methode analysieren. Ansonsten nutzen Firmen möglicherweise Methoden in einem Kontext, für den sie nicht geschaffen wurden.

Die Helena Studie untersucht nicht welche Ziele die Firmen mit der Nutzung von einzelnen

Methoden und Praktiken verfolgen. Zudem untersucht die Helena Studie nur ob bestimmte Methoden kombiniert werden, jedoch nicht, auf welche Weise sie kombiniert werden und wie sie von den Firmen genutzt werden. Nach Tell et al. [2] ist die Untersuchung dieser beiden Aspekte jedoch wichtig für die Erforschung von hybriden Ansätzen.

Die Erforschung der Art und Weise, wie agile und plan-basierte Entwicklung miteinander kombiniert wird und wie Methoden eingesetzt werden, besteht überwiegend aus Fallstudien und Erfahrungsberichten [19]. An dieser Stelle werden einige Publikationen beispielhaft genannt.

Bick et al. [32] haben im Rahmen einer Fallstudie untersucht warum bei einer plan-basierten Planung auf Interteamebene und einer agilen Planung auf Entwicklerteamebene Koordinationsschwierigkeiten auftreten. Sie kommen zu dem Schluss, dass beide Ebenen einen unterschiedlichen Fokus haben. So ist die Interteamebene zum Beispiel darauf fokussiert, wann Features fertig werden. Die Entwicklerteamebene ist dagegen darauf fokussiert Sprints zu planen. Die fehlende Abstimmung dieser beiden Fokusse sorgt für Koordinationsprobleme.

Cao et al. [33] haben die Vorteile einer Architekturphase zu Beginn des Projektes mit Extreme Programming in einer Fallstudie untersucht. Sie beschreiben, dass eine Architektur ein stabiles Grundgerüst bietet, auf dem das Projektteam aufbauen kann. Dies gibt Entwicklern ein klares Verständnis für das gesamte System und Abhängigkeiten können vom Projektteam leichter gemanagt werden.

Heeagar and Nielsen [34] beschreiben die Schwierigkeiten auf Grund der Inflexibilität von Anforderungsdokumenten und dem Aufwand, der benötigt wird, um diese zu ändern. Die identifizierte Lösung ist, dass detaillierte Anforderungsdokumentation iterativ angefertigt wird.

Diese Publikationen sind Beispiele für Fallstudien und Erfahrungsberichte im Bereich von hybrider Entwicklung. Auf weitere ähnliche Publikationen wird an dieser Stelle nicht näher eingegangen, da eine genauere Untersuchung von Ergebnissen aus Fallstudien und Erfahrungsberichten im nächsten Kapitel erfolgt. Neben der Vielzahl an solchen Fallstudien gibt es jedoch wenig allgemeine Forschung zu der Nutzung von APH-Ansätzen, die Daten aus mehreren Fällen kombinieren und daraus allgemeine Erkenntnisse ableiten [19].

Bick et al. [35] haben die verschiedenen Möglichkeiten der Interteamkoordination in großen Entwicklerteams im Rahmen einer Multifallstudie bei SAP SE untersucht. Sie entdeckten fünf unterschiedliche Koordinationsmöglichkeiten, die sich in der Menge der zentralen und horizontalen Koordination unterscheiden. Zentrale Koordination eignet sich besonders in Fällen, in denen Abhängigkeiten vor der Entwicklung bereits bekannt sind. Eine horizontale Koordination eignet sich dann, wenn Entwicklerteams Abhängigkeiten eigenständig lösen können.

Boehm und Turner [36] analysieren die Managementschwierigkeiten, wenn agile und plan-

basierte Entwicklerteams am selben System arbeiten. Sie identifizieren Schwierigkeiten im Bereich des Entwicklungsprozesses, Businessprozesses und im Bereich des Personals. Die Schwierigkeiten treten zum Beispiel auf, da agile und plan-basierte Entwicklungsprozesse unterschiedlich sind und aufeinander abgestimmt werden müssen. Agile und plan-basierte Teams schaffen zudem möglicherweise unterschiedliche Artefakte, die nicht einfach zu integrieren sind.

Theobald und Diebhold [37] haben die Probleme an der Schnittstelle zwischen agilen Teams und ihrer nicht-agilen Umgebung untersucht. Interne Schnittstellen existieren zwischen agilen und nicht-agilen Teams, die am selben Projekt arbeiten, aber auch zur umgebenden Organisation. Andere Organisationsabteilungen, wie Personalabteilungen, verlangen häufig eine andere Arbeitsweise als agile Teams. Externe Schnittstellen existieren zum Kunden, Zulieferern und Regulationsbehörden. Es ist schwierig für agile Teams mit traditionellen Kunden oder Zulieferern zu arbeiten. Regulationsbehörden bringen häufig Einschränkungen für agile Entwicklung, die häufig durch ein pures agiles Vorgehen nicht berücksichtigt werden können.

Die Arbeiten von Boehm und Turner [36] und von Theobald und Diebhold [37] beschäftigen sich damit, dass agile Teams in einem traditionellen Umfeld mit anderen Teams am gleichen System arbeiten. Dies berücksichtigt jedoch nicht, dass Teams sowohl mit einem plan-basierten als auch einem agilen Ansatz arbeiten.

Hoda et al. [38] haben in einer Interviewstudie untersucht, auf welche Probleme agile Teams in ihrem jeweiligen Kontext treffen und welche Maßnahmen sie gegen diese Probleme ergreifen. Zum Beispiel sind agile Projektteams mit dem Problem konfrontiert, dass Kunden Verträge über die Dauer, Preis und Umfang eines Projektes möchten. Die Projektteams gehen mit diesem Problem um, indem sie zum Beispiel in den Verträgen erlauben, dass Features ausgetauscht werden können oder halten einen zusätzlichen Puffer in den Verträgen fest. Ein weiteres Problem ist der Umgang mit Systemen, bei denen die Architektur komplex ist. Um solche Systeme zu implementieren, entwickeln die Teams zu Beginn des Projektes eine robuste Architektur und nutzen einen parallelen Architekturprozess zur Entwicklung. Dieser erstellt die Architektur für Features, bevor deren Entwicklung beginnt.

Waterman et al. [39] haben in einer Interviewstudie untersucht, wie viel Architekturaufwand ein agiles Projektteam zu Beginn eines Projektes betreiben sollte. Einflussfaktoren für weniger Architekturdefinition sind zum Beispiel instabile Anforderungen und die Schaffung von frühem Wert. Einflussfaktoren für mehr Architekturdefinition sind hingegen technische Risiken oder ein nicht-agiler Kunde. Im zweiten Schritt haben sie unterschiedliche Architekturstrategien identifiziert. Diese sind zum Beispiel, dass das Projektteam zu Beginn des Projektes nur die tech-

nischen Risiken adressiert oder dass sie Anwendungsframeworks, wie zum Beispiel Django[2], einsetzen.

Die allgemeinen Untersuchungen von hybriden Entwicklungsansätzen sind gering. Die meiste Forschung besteht aus Fallstudien, bei denen nicht unterschiedliche Fälle miteinander verglichen werden. Einen großen Beitrag zu der Erforschung von der Nutzung von hybriden Entwicklungsansätzen hat die Helena Studie erbracht. Tell et al. [2] kommen jedoch zu der Erkenntnis, dass für die Erforschung von hybriden Entwicklungsmethoden die reine Analyse der verwendeten Methoden nicht ausreicht. Stattdessen muss nicht nur untersucht werden, ob sie kombiniert werden, sondern auf welche Weise und wie sie genutzt werden. In dieser Dissertation wird diese Forschungslücke adressiert. Zum Beispiel wird im nächsten Kapitel untersucht, welche Vorgehensmodell Firmen in APH-Ansätze nutzen.

## 3.2 Anpassung von hybriden Entwicklungsansätzen

Es gibt eine Vielzahl von Publikationen, die hybride und insbesondere APH-Entwicklungsprozesse vorstellen und die Firmen dabei unterstützen sollen mit einem bestimmten Projektkontext umzugehen. Beispiele hierfür sind eine Entwicklung in großen Projektteams [33, 40–42] oder eine sicherheitskritischen Entwicklung [43, 44]. An dieser Stelle wird nicht genauer auf diese Publikationen eingegangen, da es im nächsten Kapitel eine ausführliche Analyse der beschriebenen Entwicklungsprozesse aus diesen Publikationen gibt.

Ansätze, die Firmen bei der Kombination von Methoden und dem Zuschneiden von Methoden unterstützten, gibt es. Sie sind zum einen jedoch häufig auf agile Entwicklung fokussiert [12], zum anderen fehlt ein systematischer Ansatz [12]. Boehm und Turner [5] haben einen ersten Ansatz für die Kombination von agiler und plan-basierter Entwicklung geschaffen. Sie haben die Unterschiede von agiler und plan-basierter Entwicklung untersucht und diskutieren, welches Vorgehen in welchem Kontext geeignet ist. Aus dieser Untersuchung leiten sie fünf Faktoren ab, die den jeweiligen *Home Ground* eines Vorgehens beschreiben. Die fünf Faktoren sind in Abbildung 3.1 dargestellt.

Je mehr sich ein Projekt sich mit den Projektfaktoren am Rand des Radarcharts in Abbildung 3.1 bewegt, desto mehr liegt es im Home Ground von plan-basierter Entwicklung. Projekte, die sich in der Mitte befinden liegen im agilen Home Ground. Wenn das Projekt weder komplett im agilen noch im plan-basierten Home Ground liegt, sollte das Projektteam einen APH-Ansatz wählen. Dafür beschreiben sie ein risikobasiertes Vorgehen, in dem die Risiken von agiler und plan-basierter Entwicklung gegeneinander abgewogen werden.

[2] www.djangoproject.com

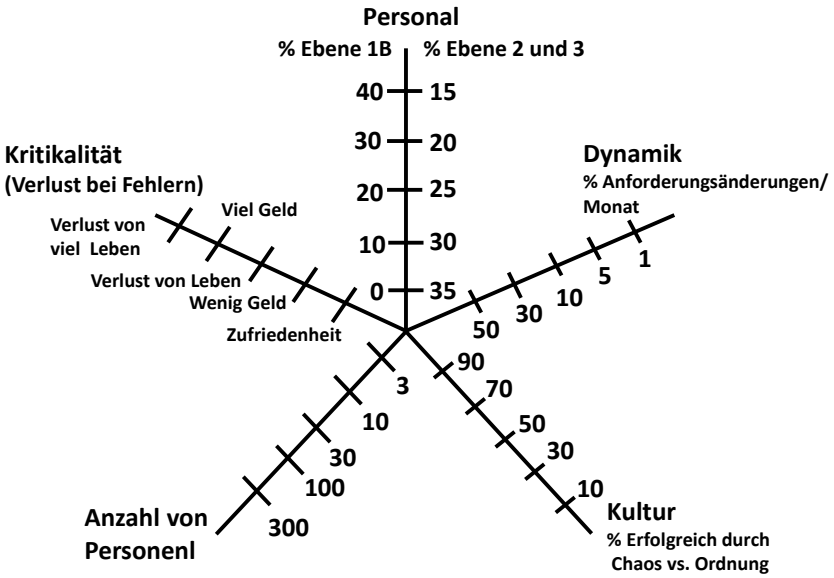


**Abbildung 3.1:** Einflussfaktoren für die Wahl des Entwicklungsansatzes nach Boehm und Turner [5]
Ebene 1B: Personen, die mit Training Methodenschritte ausführen können (Programmieren einer einfach Funktion)
Ebene 2 und 3: Personen, die Methoden auf Projektkontexte anpassen können

Geras et al. [45] haben die Aspekte für die Bestimmung des Home Ground erweitert und unterscheiden zwischen menschlichen Aspekten und produktspezifischen Eigenschaften. Bei den menschlichen Eigenschaften betrachten sie die Entwickler und Kunden in einem Softwareprojekt. Die abgeleiteten Aspekte sind Kenntnisse von agilen Methoden, Erfahrung und Einstellung zu agilen Methoden, die Verfügbarkeit des Kunden, das Domainwissen und die Kultur. Die produktspezifischen Eigenschaften beschäftigen sich mit dem Produkt selbst und der Umgebung, in der das Produkt operieren muss. Die abgeleiteten Aspekte sind Häufigkeit der Anforderungsänderungen, Unsicherheiten, Kritikalität des Verlustes bei Fehlern in der Software, die Komplexität des Produkts und der Umgebung, die Größe des Produkts und der verlangten Qualität.

Thesing et al. [24] haben eine Interviewstudie durchgeführt, um die Vor- und Nachteile von agiler und plan-basierter Entwicklung zu untersuchen. Basierend auf ihren Resultaten, haben sie ein Verfahren entwickelt, mit dem ein Projektteam entscheiden kann, ob es einem agilen oder plan-basierten Ansatz folgen sollte. Zuerst haben sie Ausschlusskriterien für agile Entwicklung definiert. Projekte, die nicht in einzelne Inkremente eingeteilt werden können oder in denen Änderungen nicht möglich sind, sind ausgeschlossen von agiler Entwicklung. In einem zweiten Schritt definieren sie weitere Faktoren basierend auf Projekteinschränkungen, Menschen und Kultur. Projekteinschränkungen berücksichtigen die Klarheit des Umfanges, die Länge des Projektes und das Bedürfnis nach frühen Ergebnissen und fixierten Ressourcen.

Die Menschen- und Kulturaspekte berücksichtigen den Organisationstyp und die Größe des Projektteams.

Silvestre et al. [46] haben eine auf BPMN basierende Methode entwickelt, um hybride Entwicklungsprozesse in Abhängigkeit von Zielen und Kontextfaktoren anzupassen. Das Projektteam kann in dieser Methode verschiedene Regeln definieren, die eine Veränderung am BPMN Modell beschreiben. Diese Regeln werden bei bestimmen Kontextfaktoren ausgelöst. Damit diese Methode genutzt werden kann, muss jedoch der Entwicklungsprozess als BPMN Modell vorliegen.

Xu und Ramesh [47] haben einen Fragenkatalog entwickelt, mit dessen Hilfe Firmen Herausforderungen in ihrem Entwicklungsprozess identifizieren können. Diese können zum Beispiel mangelnde Ressourcen (Zeit oder Personal) oder Kommunikationsherausforderungen sein. Zusätzlich geben sie Strategien an, wie diese adressiert werden können.

Kalus und Kuhrmann [10] haben eine systematische Literatursuche durchgeführt und dabei die Faktoren untersucht, die wichtig für die Anpassung von Softwareprozessen an einen bestimmten Kontext sind. Sie haben dabei 49 Einflussfaktoren identifiziert, die sie in vier Kategorien eingeteilt haben. Diese Kategorien sind das Team, die interne Umgebung, die externe Umgebung und Ziele. Sie haben zudem 20 Maßnahmen identifiziert, die jeweils beim Eintreten eines Kontextfaktors angewendet werden können. Sie diskutieren, dass zwar die Einflussfaktoren bekannt sind, die bei der Anpassung von Softwareprozessen beachtet werden müssen, die daraus resultierenden Maßnahmen für ein Projektteam sind jedoch abstrakt und müssen für jedes Projekt individuell gefunden werden.

Diese Erkenntnis und das Fehlen eines strukturierten Ansatzes für hybride Entwicklungsprozesse erklären möglicherweise, warum hybride Entwicklungsansätze häufig durch einen evolutionären Prozess gebildet werden, der auf Erfahrung aus vergangenen Projekten beruht [1]. Klünder et al. [1] haben die Daten der Helena Studie untersucht und herausgefunden, dass in 83,9% der Fälle ein evolutionäres Vorgehen verwendet wird.

Die Forschung im Bereich von APH-Ansätzen ist fokussiert auf die Entscheidung, ob ein APH-Ansatz genutzt werden sollte oder nicht. Es wird jedoch nicht diskutiert, wie sich verschiedene Kontextfaktoren auf einen Entwicklungsprozess auswirken und welche Anpassungen von einem Projektteam durchgeführt werden müssen, um bestimmte Ziele zu erreichen. An dieser Stelle existiert eine weitere Forschungslücke im Bereich von APH-Ansätzen. Diese Forschungslücke wird in dieser Dissertation adressiert. Dafür werden zum Beispiel im folgenden Kapitel untersucht, welche Ziele Firmen mit der Nutzung von agilen und plan-basierter Entwicklung in APH-Ansätzen verfolgen und welche Maßnahmen sie dafür nutzen.

# 4

# Analyse der Nutzung von APH-Ansätzen durch eine Literaturstudie

Das Ziel dieser Arbeit ist es, ein allgemeines Framework zu schaffen, das an einen individuellen Kontext angepasst werden kann, um die Entwicklung von APH-Ansätzen (Agil Plan-basiert Hybrid) zu systematisieren. Für die Schaffung dieses Frameworks wird in diesem Kapitel untersucht, welche Ziele Firmen mit einem APH-Ansatz verfolgen, wie Firmen die Prozesse in APH-Ansätzen organisieren und welche Schwierigkeiten sie bei der Bildung von APH-Ansätzen haben. Es ist wichtig diese Punkte zu erforschen, um ein Framework zu entwickeln, das einen echten Nutzen für die Firmen darstellt. In diesem Kapitel stelle ich die von mir und zwei weiteren Wissenschaftlern[1] durchgeführte Erforschung dieser Aspekte vor. Teile der Ergebnisse dieses Kapitels wurden in [48] veröffentlicht und auf der *International Conference on Software and System Processes* präsentiert. Weitere Teile wurden im *Journal of Software: Evolution and Process* veröffentlicht [3].

[1] Carolin Unger-Windeler und Kurt Schneider

## 4.1 Forschungsvorgehen

Wie bereits erläutert, ist es wichtig, für die Entwicklung eines allgemeinen Frameworks für APH-Ansätze die Bedürfnisse der Firmen zu verstehen. Dafür wurden die folgenden Forschungsfragen formuliert:

**RQ1: Welches Vorgehen nutzen Firmen für den Entwicklungsprozess in APH-Ansätzen?**

Für die Entwicklung ist es wichtig zu verstehen, wie die Entwicklungsprozesse von Firmen in APH-Ansätzen aufgebaut sind. Daher untersuchen wir mit dieser Forschungsfrage welche Aktivitäten die Firmen in ihren APH-Ansätzen nutzen und wie sie diese anordnen und verwenden.

**RQ2: Welche Ziele verfolgen Firmen mit der Anwendung von agiler und plan-basierter Entwicklung in APH-Ansätzen und wie adressieren sie diese jeweils?**

Nach Kalus und Kuhrmann [10] existiert eine Forschungslücke hinsichtlich der Auswirkungen eines Kontextes auf einen hybriden Entwicklungsprozess. Mit dieser Forschungsfrage erfolgt die Untersuchung der Ziele, die Firmen mit APH-Ansätzen verfolgen und wie sie diese adressieren. Auf diese Weise wird die Forschungslücke für APH-Ansätze adressiert und es kann ein Framework geschaffen werden, das einen tatsächlichen Nutzen für die Firmen darstellt.

**RQ3: Welche Schwierigkeiten haben die Firmen bei der Nutzung von APH-Ansätzen?**

Das Framework in dieser Arbeit soll eine Unterstützung für Firmen bei der Nutzung von APH-Ansätzen bieten. Das Framework stellt einen Nutzen für Firmen dar, wenn es bestehende Schwierigkeiten von Firmen beim Einsatz von APH-Ansätzen adressiert.

### 4.1.1 Forschungsmethode

In Erfahrungsberichten oder auch Fallstudien wird oft darüber berichtet, welche Probleme und Schwierigkeiten eine Firma hat, gewisse Ziele zu verfolgen und wie sie diese gelöst haben. Daher eignen sich diese Publikationen für die Beantwortung der Forschungsfragen. APH-Ansätze werden bereits über einen langen Zeitraum genutzt, daher gibt es bereits viel Literatur in Form von Fallstudien und Erfahrungsberichten zu APH-Ansätzen [8, 12]. Es gibt daher im Bereich der Literatur viele Quellen, die für die Untersuchung der Forschungsfragen herangezogen werden können.

Wegen dieser Vorteile haben wir uns für eine Systematische Mappingstudie entschieden, um die Forschungsfragen zu beantworten. Im Gegensatz zu einer systematischen Literatursuche, bei der Beweise für etwas gesucht werden, ist das Ziel einer systematischen Mappingstudie ein Überblick über ein Forschungsfeld zu bekommen und es zu strukturieren [49]. Die Planung und Durchführung der Mappingstudien folgt dem entwickelten Prozess von Petersen et al. [50]. Das Vorgehen von Petersen et al. [50] wurde durch einen Schneeballprozess erweitert, um weitere relevante Publikationen zu finden [51]. Im Folgenden wird das Vorgehen der Mappingstudie im Detail beschrieben.

#### 4.1.1.1 Definition des Suchstrings

Bei einer Mappingstudie muss das zu untersuchende Forschungsgebiet durch Schlüsselwörter definiert werden. Bei dieser Mappingstudie werden Publikationen gesucht, die sich mit der Kombination von agilen und plan-basierten Ansätzen beschäftigen. Die Kombination von agilen und plan-basierten Ansätzen werden in der Literatur *hybrid development approaches* genannt [6, 7]. Da sich die Mappingstudie auf Softwareentwicklung konzentriert und um das Schlüsselwort so generisch wie möglich zu formulieren, haben wir *hybrid software development* als ein Schlüsselwort gewählt. Viele Autoren beschreiben jedoch nur die Kombination von agilen und plan-basierten Ansätzen, ohne den Begriff *hybrid* zu verwenden. Um auch diese Publikiation zu finden, haben wir die Begriffe *agile*, *plan-based* und *combine* zum Suchstring hinzugefügt. In vielen Publikationen wird der Begriff *traditionell* als Synonym für *plan-basiert* verwendet [7, 52, 53]. Aus diesem Grund haben wir auch diesen Begriff hinzugefügt. Um agile Methoden einzuführen und APH-Ansätze zu entwickeln, *integrieren* Firmen häufig agile Methoden in einen bestehenden plan-basierten Ansatz [54, 55]. Daher haben wir auch diesen Begriff zum Suchstring hinzugefügt. Boehm und Turner [23] legen in ihrer Arbeit dar, dass Firmen agile und plan-basierte Ansätze *balancieren* müssen, um erfolgreich zu sein und prägten diesen Begriff in diesem Zusammenhang. Auch diesen Begriff haben wir daher als Synonym zum Suchstring hinzugefügt.

Boehm and Turner [23] diskutieren weiter, dass die Größe des Projektteams und die Kritikalität (Verlust aufgrund von Fehlern) wichtige Einflussfaktoren für die Entwicklung von APH-Ansätzen sind. Ein großes Projektteam und eine hohe Kritikalität führen häufig zu einem APH-Ansatz. Um weitere Publikationen zu finden, die die Gründe von APH-Ansätzen beschreiben, haben wir die Begriffe *large-scale* and *safety critical* kombiniert mit dem Begriff *agile* zum Suchstring hinzugefügt. Der vollständige Suchstring ist in Abbildung 4.1 abgebildet. Schlüsselwörter, die parallel angeordnet sind, sind durch ein *ODER* verknüpft und solche die sequentiell angeordnet sind, sind durch ein *UND* verknüpft.

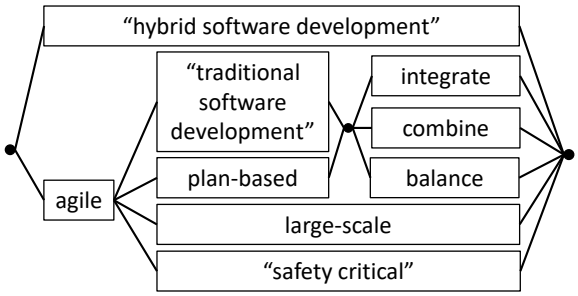


**Abbildung 4.1:** Abbildung des ersten Suchstrings

Für die Beantwortung von Forschungsfrage RQ1 werden Fallstudien und Erfahrungsberichte benötigt, die den Prozess des Entwicklungsansatzes beschreiben. Um eine erweiterte fokussierte Suche nach solchen Fallstudien durchzuführen und weitere Publikationen zu finden, haben wir einen zweiten Suchdurchlauf durchgeführt, bei dem wir das Wort *process* zum Suchstring hinzugefügt haben. Ebenfalls haben wir die Wörter *approach* und *framework* als Synonyme mit hinzugefügt. Der Suchstring für den zweiten Suchdurchlauf findet sich in Abbildung 4.2. Der Klartext für beide Suchstrings findet sich in Appendix A.

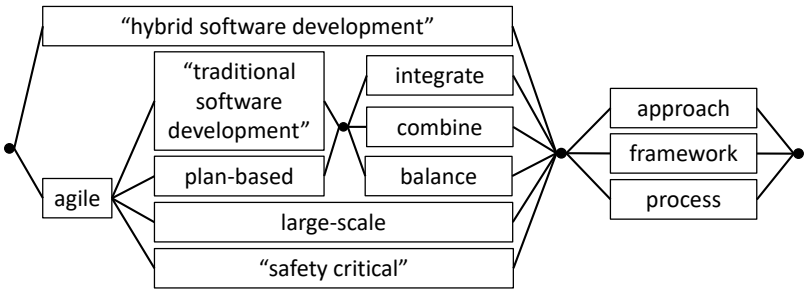


**Abbildung 4.2:** Abbildung des zweiten Suchstrings

#### 4.1.1.2 Definition der Ein- und Ausschlusskriterien

Um Objektivität während des Filterungs- und Auswahlprozesses zu gewährleisten, haben wir Ein- und Ausschlusskriterien definiert. Die Kriterien sind in Tabelle 4.1 zusammengefasst. Wir haben die Publikationen aufgenommen, die mindestens ein Einschlusskriterium erfüllen und keines der Ausschlusskriterien.

Das Ziel der Mappingstudie ist es unter anderem, die Ziele und Gründe von APH-Ansätze zu verstehen. Es werden daher nur solche Publikationen berücksichtigt, die einen absichtlich eingeführten APH-Ansatz beschreiben. Oftmals nutzen Firmen APH-Ansätze während einer

**Tabelle 4.1:** Liste der Ein- und Ausschlusskriterien

| | Kriterium | Beschreibung |
|---|---|---|
| **Einschlusskriterien** | $EK_1$ | Die Publikation beschreibt Ziele, die von Firmen mit einem APH-Ansatz verfolgt werden und deren Maßnahmen, um diese Ziele zu erreichen. |
| | $EK_2$ | Die Publikation beschreibt Ziele, die Firmen mit einem APH-Ansatz verfolgen. |
| | $EK_3$ | Die Publikation beschreibt Schwierigkeiten oder Probleme, die die Firmen bei der Nutzung von APH-Ansätzen haben. |
| | $EK_4$ | Die Publikation beschreibt Erfolgsfaktoren oder Praktiken für die Lösung von Schwierigkeiten und Problemen. |
| | $EK_5$ | Die Publikation beschreibt den Prozess, Framework oder Struktur eines APH-Ansatzes. |
| **Ausschlusskriterien** | $AK_1$ | Die Publikation beschreibt eine Transformation zu einem agilen Entwicklungsansatz. |
| | $AK_2$ | Die Publikation beschreibt die Anwendung agiler Praktiken in einem sonst strikten Wasserfallprozess. |
| | $AK_3$ | Die Publikation nutzt keine Daten, die in der Industrie, einer Firma oder in einem Projekt erhoben wurden. |
| | $AK_4$ | Die Publikation ist weder auf Deutsch noch auf Englisch geschrieben. |
| | $AK_5$ | Die Publikation ist kein peer-reviewter Beitrag auf einer Konferenz oder in einem Journal. |

Transformation von einem plan-basierten Ansatz hin zu einen agilen Ansatz, um die Einführung agiler Methoden zu managen und weiterhin produzieren zu können [56]. Diese Fälle beschreiben jedoch keine absichtliche Nutzung von APH-Ansätzen. Zudem sind die APH-Ansätze in diesen Fällen nur eine Übergangslösung und sollen nicht auf lange Sicht genutzt werden. Daher werden in diesen Fällen keine Ziele beschrieben, die zu einer absichtlichen Nutzung von APH-Ansätzen führen. Es werden daher Publikationen ausgeschlossen, die eine agile Transformation beschreiben ($AK_1$)

Basierend auf der Definition von APH-Ansätzen werden Fälle ausgeschlossen, in denen agile Methoden in einem sonst strikten Wasserfallprozess genutzt werden ($AK_2$). In diesen Fällen folgen die Firmen gänzlich den Prinzipien von plan-basierter Entwicklung, ohne dass sie einen Kompromiss mit den agilen Prinzipien schließen. Auf diese Weise profitieren die Firmen nicht von beiden Ansätzen.

Da im Rahmen dieser Mappingstudie die tatsächlichen Ziele und Schwierigkeiten von Firmen untersucht werden, haben wir bei der Auswahl der Publikationen darauf geachtet, dass in diesen Daten verwendet werden, die direkt in der Wirtschaft und damit in tatsächlichen Firmen erhoben wurden. Zu diesen Publikationen gehören beispielsweise Fallstudien, Erfahrungsberichte oder Interviewstudien, da hier die Datensammlung in und mit Firmen stattgefunden hat. Publikationen, die APH-Ansätze nur theoretisch betrachten, wurden ausgeschlossen ($AK_3$).

Wir haben auch Publikationen eingeschlossen, die Literatursuchen durchgeführt haben und dabei diese Punkte beachtet haben.

Um RQ1 zu beantworten wurden Publikationen eingeschlossen, die die Prozesse in einem APH-Ansatz beschreiben ($EK_5$). Hierbei wurden auch Publikationen eingeschlossen, die einen APH-Ansatz vorschlagen, um die Industrie zu unterstützen. Für die Beantwortung von RQ2 haben wir Publikationen eingeschlossen, die Ziele von Firmen in APH-Ansätzen beschreiben und die Maßnahmen mit denen sie diese adressieren ($EK_1$). Es wurden auch solche Publikationen hinzugenommen, die nur Ziele, aber keine Maßnahmen beschreiben ($EK_2$). Für die Beantwortung von RQ3 haben wir Publikationen eingeschlossen, die Probleme und Schwierigkeiten nennen, die Firmen bei der Nutzung von APH-Ansätzen haben ($EK_3$ und $EK_4$).

#### 4.1.1.3 Auswahl der Datenbanken

Für die Mappingstudie haben wir fünf Datenbanken ausgewählt: Google Scholar, IEEE, ACM, Science Direct und SpringerLink. In jeder der Datenbanken haben wir eine gründliche Suche durchgeführt. Die Datenbanken bieten teilweise unterschiedliche Suchoptionen, daher wurde der Suchstring jeweils angepasst.

#### 4.1.1.4 Durchführung

Mit Hilfe des jeweiligen Suchstrings wurde jeweils ein Startset gebildet, gefolgt von zwei Iterationen Snowballing. Nach den zwei Iterationen haben wir die Suche jeweils beendet, da keine neuen Publikationen dazu gekommen sind. Im Folgenden wird der Filterungsprozess im Detail beschrieben.

Die jeweiligen Startsets entstehen durch die automatische Suche mit den Suchstrings in den ausgewählten Datenbanken. Die Auswahl der Publikationen folgt in beiden Durchläufen dem selben Ablauf. Die erste Filterung erfolgt durch die Anwendung der Ein- und Ausschlusskriterien auf die Titel der Publikationen. Die weitere Filterung erfolgt durch die Anwendung der Kriterien auf die Zusammenfassungen und der Schlüsselwörter der Publikation und im letzten Schritt auf deren Inhalt.

Nach der Bildung des Startsets durch die automatisierte Suche erfolgt der Snowballingprozess. Dafür erfolgte für jede Publikation eine Vorwärts- und Rückwärtssuche. Für die Rückwärtssuche werden die Referenzen der Publikationen verwendet und für die Vorwärtssuche alle Publikationen, in denen die betreffende Publikation mindestens einmal zitiert wird [51].

Wie viele Publikationen die beiden Suchdurchläufe für die Startsets und der jeweilige anschließende Snowballingprozess erbracht haben, ist für den ersten Suchstring in Abbildung

A.1 in Appendix A und für den zweiten Suchstring in Abbildung A.2 in Appendix A dargestellt. Der Suchdurchlauf mit dem ersten Suchstring erbrachte insgesamt 36 Publikationen und der Durchlauf mit dem zweiten Suchstring erbrachte 24 Publikationen. Anschließend wurden Publikationen aus beiden Suchdurchläufen auf Duplikate überprüft. Die anschließende Überprüfung auf Duplikate in den beiden Suchdurchläufen führte zum Ausschluss von 6 Publikationen. Eine Übersicht über alle identifizierten Publikationen und des Filterungsprozesses findet sich in Appendix A.

#### 4.1.1.5 Datenextraktion und Analyse

Die Extraktion der relevanten Daten ist unterstützt durch die Erstellung eines Datenextraktionsformulars, das in Tabelle 4.2 abgebildet ist. Jedes Feld in dem Formular hat einen Namen und einen Wert. Soweit wie möglich sind die Felder den jeweiligen Forschungsfragen zugewiesen. Nach der Extraktion der Daten haben wir eine thematische Analyse durchgeführt [57]. Dafür werden initiale Codes den einzelnen Textpassagen zugeordnet, die die Forschungsfragen beantworten. In einem zweiten Schritt wird durch konstantes Vergleichen der Codes Themen gebildet.

Für die Beantwortung von RQ1 haben wir die Beschreibungen des Entwicklungsansatzes aus den Publikationen extrahiert. Die Beschreibungen haben wir genutzt, um eine graphische Darstellung des jeweiligen beschriebenen Entwicklungsprozesses zu erstellen. Dabei haben wir die genannten Tätigkeiten und deren Abläufe festgehalten und zusätzlich die genannten Rollen, Artefakte, Meetings und Praktiken dokumentiert. Darüber hinaus haben wir zu diesen Elementen zusätzlich präsentierte Informationen, wie der Inhalt von Artefakten, Aufgaben und Verantwortlichkeiten von Rollen oder die Länge von Tätigkeiten, festgehalten.

Um die Entwicklungsprozesse miteinander vergleichen zu können, erfolgte eine Untersuchung der Tätigkeiten hinsichtlich der Ziele, die die Firmen damit verfolgen. Diese Ziele sind zum Beispiel das Sammeln von Anforderungen oder Testen. Dabei wurde ersichtlich, dass die Tätigkeiten Aktivitäten bilden, die mit den Standardaktivitäten aus dem Software Lifecycle übereinstimmen. Diese Standardaktivitäten sind Requirements Engineering, Design, Planung, Implementierung, Testen und Veröffentlichung [15]. Für die Beantwortung der ersten Forschungsfrage haben wir die Darstellungen der Entwicklungsprozesse manuell nach der Anordnung der Aktivitäten geclustert.

Zusätzlich erfolgte eine Untersuchung der Aktivitäten der geclusterten Entwicklungsprozesse. Dafür haben wir die genannten Rollen, Artefakte, Meetings und Praktiken in den Aktivitäten betrachtet und basierend darauf auf Gemeinsamkeiten und Unterschiede untersucht.

**Tabelle 4.2:** Datenextraktionsformular

| | Name | Wert | RQ |
|---|---|---|---|
| Allgemein | Study ID | Integer | |
| | Titel | Name der Publikation | |
| | Autoren | Liste von Namen | |
| | Publikationsjahr | Kalenderjahr | |
| | Forschungsmethoden | Angewendete Forschungsmethode | |
| Inhalt | Agile Ziele | Liste von agilen Zielen, die verfolgt werden | RQ2 |
| | Agile Maßnahmen | Liste von Paaren aus agilen Zielen und implementierten agilen Maßnahmen | RQ2 |
| | Plan-basierte Ziele | Liste von plan-basierten Zielen, die verfolgt werden | RQ2 |
| | Plan-basierte Maßnahmen | Liste von Paaren aus plan-basierten Zielen und implementierten plan-basierten Maßnahmen | RQ2 |
| | Schwierigkeiten | Liste von genannten Schwierigkeiten | RQ3 |
| | Erfolgsfaktoren | Liste an genannten Erfolgsfaktoren | RQ3 |
| | Entwicklungsprozess | Darstellung des entnommenen Entwicklungsansatzes | RQ1 |

#### 4.1.1.6 Bedrohungen der Validität

Hinsichtlich der Bedrohung der Validität werden nach Maxwell [58] und Peterseon et al. [49] deskriptive, theoretische und interpretative Validität und die Generalisierbarkeit betrachtet.

**Deskriptive Validität** Deskriptive Validität betrachtet die Korrektheit der gemachten Beobachtungen. Um dieser Bedrohung zu begegnen, wurde ein konkretes Datenextraktionsformular erstellt, um die Aufnahme der Daten zu erleichtern. Das Formular ermöglicht es, Daten zu ihrer Quelle zurückzuverfolgen. Um möglichen Bias bei der Extraktion der Daten entgegen zu wirken, wurde dieser Schritt vom Schreiber dieser Arbeit und von Carolin Unger-Windeler zusammen durchgeführt. Zudem wurde die Extraktion der Daten zwischen beiden Forschern diskutiert, um eine Übereinstimmung zu erreichen.

**Theoretische Validität** Die theoretische Validität betrachtet, in wieweit die relevanten Aspekte gesammelt werden. Das Ziel der Mappingstudie war es, einen Überblick hinsichtlich des wissenschaftlichen Wissens über die Ziele, Balancepunkte und Organisation von APH-Ansätzen zu erlangen. Wissenschaftliche Publikationen betrachten APH-Ansätze jedoch häufig von einem theoretischen Standpunkt aus und betrachten nicht die Ziele der Industrie. Um diese Bedrohung zu minimieren wurden Publikationen, die keine Verbindung zur Industrie, Firmen oder Projekten haben, ausgeschlossen.

Um das Risiko zu verringern, Publikationen zu übersehen, wurde ein Suchstring formuliert, der das Forschungsfeld für APH-Ansätze sehr weit aufspannt. Dafür wurden dem Suchstring

Synonyme für verschiedene Begriffe des Suchstrings, wie *traditional* oder *balance*, die in der Forschung von APH-Ansätzen genutzt werden, hinzugefügt. Zudem wurde der Suchstring mit anderen Experten im Bereich von APH-Ansätzen diskutiert. Zusätzlich wurden zwei Kontexte hinzugefügt, nämlich *large-scale* und *safety-critical*, die bekannt für die Nutzung von APH-Ansätzen in der Softwareentwicklung sind, um weitere Publikationen zu erreichen. Es wurden mehrere Datenbanken durchsucht und es wurde ein Snowballprozess durchgeführt, bis keine neuen Publikationen mehr gefunden wurden.

Die Mappingstudie wurde von mir und Carolin Unger-Windeler durchgeführt, um den Bias, der durch einen einzelnen Forscher entsteht, zu verringern. Der Ein- und Ausschluss von Publikationen ist jedoch subjektiv und es besteht das Risiko eine falsche Entscheidung zu treffen. Um die Objektivität des Auswahlprozesses zu erhöhen und damit dieses Risiko zu minimieren, wurden konkrete Ein- und Ausschlusskriterien aufgestellt. Unsicherheiten bezüglich des Ein- und Ausschlusses bestimmter Publikationen wurde unter den involvierten Forschern besprochen. Die Datenextraktion wurde hauptsächlich vom Schreiber dieser Arbeit durchgeführt, die zweite Forscherin hat jedoch den gesamten Prozess überprüft.

**Interpretative Validität** Die Interpretative Validität betrachtet die Schlüsse, die aus den Daten gezogen werden. Die meisten der Resultate wurden mehrfach in verschiedenen Publikationen genannt. Daraus wird geschlossen, dass sich die unterschiedlichen Publikationen gegenseitig unterstützen und dass die Ergebnisse valide sind. Die Interpretative Validität wird daher als geringere Bedrohung eingestuft.

**Generalisierbarkeit** Es kann nicht garantiert werden, dass die entdeckten Ziele alle Ziele repräsentieren, die von Firmen mit APH-Ansätzen verfolgt werden. Möglicherweise existieren Firmen, die andere beziehungsweise unentdeckte Ziele verfolgen. Dies trifft auch auf die genannten Maßnahmen, Schwierigkeiten und entdeckten Organisationsstrukturen zu.

## 4.2 Resultate

Insgesamt haben wir 54 Publikationen für die Beantwortung unserer Forschungsfragen identifiziert. Abbildung 4.3 zeigt eine Übersicht der Forschungsmethoden, die in den Publikationen genutzt werden. Mit 37 Publikationen überwiegt deutlich die Fallstudie als Forschungsmethode. Zusätzlich gibt es fünf Erfahrungsberichte. Es gibt fünf Publikationen, in denen Design Science genutzt wird, um einen APH-Ansatz für einen bestimmten Fall zu entwickeln. In

sechs Publikationen wird die Anwendung von Design Science zusätzlich durch eine Fallstudie ergänzt, bei der der beschriebene APH-Ansatz in einem Projekt Anwendung findet.

Die Übersicht zeigt, dass allgemeine Untersuchungen von APH-Ansätzen, die nicht nur die verwendeten Methoden oder Praktiken betrachten, in der Forschung fehlen. Dies bestätigt die Notwendigkeit der durchgeführten Mappingstudie, um die Erkenntnisse aus den verschiedenen Fallstudien zu vereinen und miteinander zu vergleichen. Im Folgenden werden die Ergebnisse der Mappingstudie präsentiert.

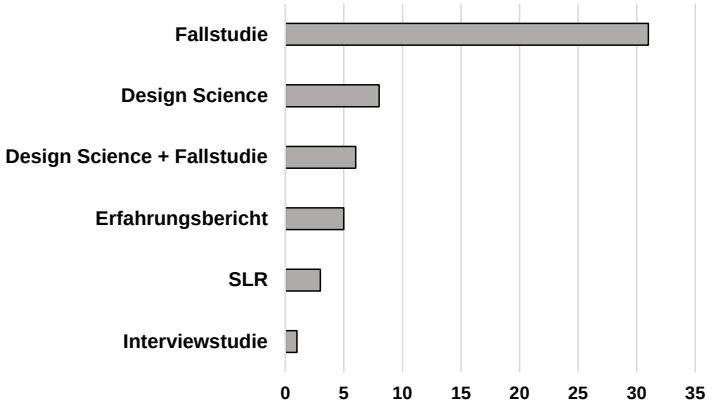


**Abbildung 4.3:** Übersicht der verwendeten Forschungsmethoden

### 4.2.1 Forschungsfrage RQ1

Unsere Analyse der Tätigkeiten in den beschriebenen Entwicklungsansätzen ergab, dass diese Aktivitäten bilden, die den Standard Aktivitäten des Software Lifecycle entsprechen. Bei allen identifizierten Entwicklungsprozessen findet sich die Nutzung dieser Aktivitäten. Der Unterschied zwischen den gefundenen Entwicklungsprozessen ist die Anordnung dieser Aktivitäten. Die Clusterbildung während der Datenanalyse erbrachte drei generelle Modelle, die Firmen zur Anordnung der Aktivitäten in APH-Ansätzen nutzen: Das Wasserfall-Agil Framework, das Wasserfall-Iteration Framework und das Pipeline Framework. Im Folgenden werden diese drei Frameworks im Detail beschrieben.

#### 4.2.1.1 Das Wasserfall-Agil Framework

Eine generelle Darstellung des Wasserfall-Agil Frameworks ist in Abbildung 4.4 abgebildet. In diesem Framework ordnen Firmen die Standardaktivitäten des Software Lifecycles sequentiell an. Damit ist diese Framework dem klassischen Wasserfallmodell von Royce [15] am

ähnlichsten. Projekte, die das Wasserfall-Agil Framework implementieren, nutzen das Wasserfallmodell als Basismodell für den Entwicklungsprozess. Es wurden sechs sequentielle Aktivitäten identifiziert, die den Aktivitäten des Wasserfallmodells entsprechen [15].

Das Wasserfall-Agil Framework startet mit der Anforderungsanalyse, in der die Anforderungen für das Projekt gesammelt werden [42, 59, 60]. Darauf folgt das Architekturdesign, in der die Architektur für die Software entwickelt wird [42, 59, 60]. Die Entwicklung der Software wird in dem Framework iterativ durchgeführt. Dafür werden vorher die Anforderungen in ein Backlog überführt [61, 62]. Während der Entwicklung der Software nutzt das Projektteam agile Methoden. Nach der Entwicklung wird die Software getestet und anschließend ausgeliefert (Operation) [44, 59, 63].

Obwohl einige der Fälle das V-Modell als Erweiterung des Wasserfallmodells nutzen, werden diese Fälle nicht gesondert beschrieben, da sich diese Fälle in ihrer Beschreibung nicht unterscheiden. Zudem wird die Nutzung des Wasserfallmodells häufiger genannt.

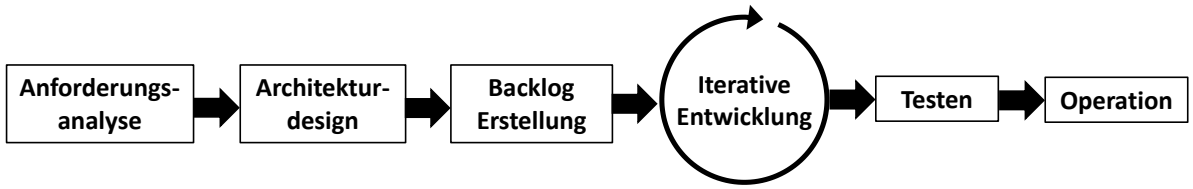


**Abbildung 4.4:** Darstellung des Wasserfall-Agil Frameworks

#### 4.2.1.2 Das Wasserfall-Iteration Framework

Das Wasserfall-Iteration Framework ist in Abbildung 4.5 dargestellt. Die Basis für dieses Framework ist ein iterativer Ansatz und die Firmen durchlaufen das Wasserfallmodell in jeder Iteration [64, 65]. Hier haben wir dieselben Aktivitäten identifiziert, die auch im Wasserfall-Agil Framework verwendet werden, ohne jedoch die Backlogerstellung. Das liegt wahrscheinlich daran, dass das Backlog bereits existiert und während der Iterationen nur erweitert wird. Die Tätigkeiten in diesen Aktivitäten sind dabei auf die aktuelle Iteration fokussiert. Das bedeutet, dass die Anforderungen und Architektur jeweils für die aktuelle Iteration entwickelt beziehungsweise erweitert werden und dass nach jeder Iteration ein neues Inkrement entwickelt und getestet ausgeliefert werden kann [64, 65].

Bose und Thakur [65] und Heeagar und Nielsen [66] beschreiben zudem, dass die iterative Entwicklung innerhalb einer großen Iteration selbst noch mal in kleinere Iterationen aufgeteilt werden kann.

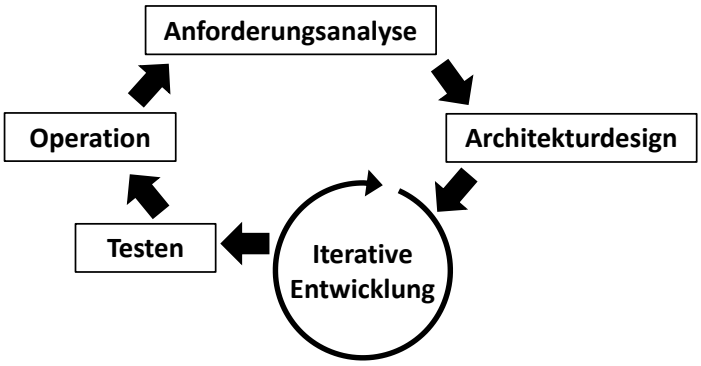


**Abbildung 4.5:** Darstellung des Wasserfall-Iteration Frameworks

#### 4.2.1.3 Das Pipeline Framework

Das Pipeline Framework folgt auch dem klassischen Wasserfallmodell. Der Unterschied hier ist jedoch, dass die Aktivitäten nicht sequenziell durchgeführt werden, sondern parallel für unterschiedliche Inkremente [67]. In Abbildung 4.6 ist eine generelle Darstellung des Frameworks dargestellt. N beschreibt das aktuelle Inkrement, das in der Entwicklung in Iteration I ist. Parallel zu dieser Aktivität wird das Architekturdesign für Inkrement N+1 entwickelt und alle notwendigen Tests für Inkrement N-1 durchgeführt. Folglich wandern alle Inkremente in der nächsten Iteration (I+1) zur nächsten Aktivität. Auch hier fehlt die Backlogerstellung, da das Backlog bereits existiert und nur erweitert wird, ähnlich zum Wasserfall-Iteration Framework.

Ähnlich zum Wasserfall-Iteration Framework beschreiben Read und Briggs [68] und Heikkilä et al. [69], dass die Entwicklungsaktivität in mehrere kurze Iterationen aufgeteilt werden kann. Die Identifizierung des Pipeline Frameworks wird unterstützt durch die Ergebnisse von Hoda et al. [38]. Sie haben Fälle identifiziert, in denen das Projektteam das Pipeline Framework für die Definition der Architektur verwendet haben.

**Beobachtung 1:** Es gibt drei mögliche Frameworks für APH-Ansätze, die alle auf dem Wasserfallmodell basieren. Der Unterschied zwischen den Frameworks ist, dass das Wasserfallmodell entweder sequentiell, iterativ oder parallel durchlaufen wird.

#### 4.2.1.4 Kombinationen und Varianten der Frameworks

In Tabelle A.1 in Appendix A sind alle Publikationen aufgelistet, die einen APH-Ansatz beschreiben. In dieser Tabelle ist dargestellt, welche der drei vorgestellten Frameworks in dem jeweiligen Fall von den Firmen genutzt wird. Ebenfalls ist dort verzeichnet, welche Aktivitäten im Frameworks die Firmen verwenden. Die nachfolgenden Erläuterungen basieren auf der Analyse dieser Tabelle und die beschriebenen Anzahlen können daraus entnommen werden.

| | | | | | |
|---|---|---|---|---|---|
| **Anforderungsanalyse** | N | N+1 | N+2 | N+3 | N+4 |
| **Architekturdesign** | N-1 | N | N+1 | N+2 | N+3 |
| **Iterative Entwicklung** ↻ | N-2 | N-1 | N | N+1 | N+2 |
| **Testen** | N-3 | N-2 | N-1 | N | N+1 |
| **Operations** | N-4 | N-3 | N-2 | N-1 | N |
| **Iteration** | **I-2** | **I-1** | **I** | **I+1** | **I+2** |

**Abbildung 4.6:** Darstellung des Pipeline Frameworks nach einer Darstellung von Paige et al. [67]

In 19 Fällen wird das Wasserfall-Agil Framework genutzt. Davon nutzen 11 Fälle das Wasserfall-Agil Framework alleinstehend und kombinieren es nicht mit einem anderen Framework. In sieben Fällen wird das Wasserfall-Iteration Framework während der Entwicklung im Wasserfall-Agil Frameworks genutzt. In einem Fall geschieht dies mit dem Pipeline Framework. Dies bedeutet, dass das Wasserfall-Agil Framework als Grundgerüst für die beiden anderen Frameworks dienen kann. Durch die Kombination entfallen Aktivitäten des Wasserfall-Agil Frameworks, wie zum Beispiel in vier Fällen das Architekturdesign zu Beginn des Projektes oder in vier Fällen das Testen am Ende des Projektes. Wohingegen die Anforderungsanalyse des Wasserfall-Agil Frameworks in allen Fällen, die das Framework verwenden, auch genutzt wird. Dies deutet darauf hin, dass die Anforderungsanalyse im Wasserfall-Agil Framework eine wichtige Aktivitäten in Projekten ist.

**Beobachtung 2:** Das Wasserfall-Agil Framework kann als Grundgerüst für das Wasserfall-Iteration oder Pipeline Framework genutzt werden. Besonders wichtig ist dabei die initiale Anforderungsanalyse zu Beginn des Projektes.

Von den 11 Fällen, die das Wasserfall-Agil Framework alleinstehend nutzen, nutzen neun die Aktivität für das Architekturdesign. Von den acht Fällen, in denen das Wasserfall-Agil Framework mit dem Wasserfall-Iteration oder Pipeline Framework kombiniert wird, sind es vier. Das deutet darauf hin, dass in diesen Fällen die Architekturtätigkeiten mehr während der iterativen Entwicklung im Wasserfall-Agil Framework durchgeführt werden. Das heißt jedoch nicht, dass in diesen Fällen immer die Aktivität für das Architekturdesign im Wasserfall-Iteration oder Pipeline Framework genutzt wird. Es gibt auch Fälle, in denen diese Aktivität in den beiden Frameworks auslassen wird und die Architektur direkt von den Entwicklern während der Implementierung erstellt wird. Dies geschieht auch in den Fällen, in denen das Wasserfall-Agil Framework alleinstehend genutzt wird und das Architekturdesign nicht genutzt wird.

**Beobachtung 3:** Das Architekturdesign des Wasserfall-Agil Frameworks wird nicht immer angewendet, stattdessen kann sich das Erstellen der Architektur mehr in die iterative Entwicklung verschieben beziehungsweise von den Entwicklern bei der Implementierung durchgeführt werden.

Es wurden nur zwei Fälle gefunden, in denen die Aktivität für die Backlogerstellung genutzt wurde. Daher wurde sie zwar zur Vollständigkeit aufgenommen, wird aber nicht als essentielle Aktivität betrachtet. In den meisten Fällen wird das Backlog direkt mit in der Anforderungsanalyse erstellt und während der Entwicklung erweitert.

In den 11 Fällen, in denen das Wasserfall-Agil Framework alleinstehend genutzt wird, wird in acht Fällen die Testaktivität genutzt (eine Publikation macht keine Angaben zum Testen). In den verbleibenden Fällen wird direkt während der iterativem Entwicklung getestet. Dafür nennen nur zwei Fälle, die das Wasserfall-Agil und Wasserfall-Iteration Framework kombinieren, dass sie die Testaktivität des Wasserfall-Agil Frameworks nutzen. Die Daten deuten darauf hin, dass bei einer Kombination des Wasserfall-Agil mit dem Wasserfall-Iteration oder Pipeline Frameworks die Test- und Operationsaktivitäten des Wasserfall-Agil Frameworks nicht genutzt werden. Stattdessen werden diese Tätigkeiten während der iterativen Entwicklung im Wasserfall-Agil Frameworks durchgeführt.

Dabei werden die Unit Testfälle direkt von den Entwicklern durchgeführt. Die Durchführung der Integrations- und Systemtests ist dagegen unterschiedlich. Teilweise werden diese Tests direkt bei der Implementierung in den Entwicklerteams mit durchgeführt, teilweise werden diese aber in einer extra Testaktivität durchgeführt. Dies geschieht durch den Einsatz der Testaktivität des Wasserfall-Agil, Wasserfall-Iteration oder Pipeline Frameworks. Auch hinsichtlich der Teamstrukturen gibt es hier Unterschiede. Zaki und Moawad [59] beschreiben zum Beispiel, dass die Qualitätssicherung in das Entwicklungsteam mit eingebunden ist. Wohingegen Portela und Borrego [70] beschreiben, dass die Implementierung und das Testen in unterschiedlichen Teams stattfindet. Da in den angegebenen Fällen nach der Durchführung der Systemtests auch die Auslieferung der Software folgt, wird davon ausgegangen, dass auch in den Fällen in denen keine Angaben über das Ausliefern der Software gemacht wird, dies ebenfalls direkt danach geschieht.

**Beobachtung 4:** Bei einer Kombination des Wasserfall-Agil mit dem Wasserfall-Iteration oder Pipeline Frameworks verschieben sich die Tätigkeiten fürs Testen und der Operation häufig in die iterative Entwicklung des Wasserfall-Agil Frameworks. Software wird so iterativ getestet und ausgeliefert.

Obwohl sich die Organisation von APH-Ansätzen in drei grundlegende Framworks aufteilen lässt, zeigen die Daten aus Tabelle A.1, dass diese Frameworks trotzdem individualisiert wer-

den. Zudem zeigt sich, dass Firmen nicht immer ein Framework nutzen, sondern diese miteinander kombinieren. Vom Autor dieser Arbeit wurde bereits in [8] identifiziert, dass Firmen durch die vielen Anpassungsmöglichkeiten Schwierigkeiten bei der Entwicklung von APH-Ansätzen haben. Dies zeigt, dass aus diesen Frameworks ein allgemeines Framework gebildet werden muss, das genau definierte Anpassungsmöglichkeiten bietet. Daher ist die Identifizierung und Analyse dieser Frameworks ein wichtiger Schritt zur systematischen Entwicklung von APH-Ansätzen. Die Daten zeigen, dass eine Anforderungsanalyse zu Beginn des Projektes und eine strukturierte iterative Entwicklung während des Projektes wichtig für ein allgemeines Framework sind, da diese konsistent über die Fälle genutzt werden. Das Wasserfall-Agil Framework bildet ein Grundgerüst für APH-Ansätze. Strukturierte Iterationen können Projektteams durch die Anwendung des Wasserfall-Iteration oder des Pipeline-Framework erreichen.

Um die Anwendung der Frameworks weiter zu untersuchen, haben wir untersucht, ob es einen Zusammenhang zwischen dem beschriebenen Kontext und dem verwendeten Framework gibt. Die Daten aus Tabelle A.1 zeigen einen leichten Trend zur Nutzung des Wasserfall-Agil Frameworks als alleinstehendes Framework bei der Entwicklung von kritischen Systemen. Dies deutet darauf hin, das bei der Entwicklung von kritischen Systemen mehr auf das klassische Wasserfallmodell gesetzt wird, als auf Agilität. Darüber hinaus, konnten jedoch keine Zusammenhänge identifiziert werden.

#### 4.2.1.5 Durchführung der Aktivitäten

In diesem Abschnitt wird untersucht wie Firmen die Anforderungsanalyse und die Architekturdefinition zwischen dem Beginn eines Projektes und dessen Laufzeit aufteilen.

**Anforderungsanalyse und Planung** In sechs Fällen wird explizit genannt, dass die Anforderungen am Anfang eines Projektes auf einem hohen Abstraktionslevel erhoben werden und während der Entwicklung die Details der Anforderungen gesammelt werden [63, 64, 70–73]. Dabei machen die Autoren jedoch keine detaillierten Angaben darüber, wie dieses Niveau der Anforderungsaufnahme aussieht. Daher haben wir die Tätigkeiten untersucht, die die Firmen während der initialen Anforderungsanalyse durchführen. Die identifizierten Tätigkeiten sind in Tabelle A.2 in Appendix A aufgelistet. Die Daten zeigen, dass das Projektteam am Anfang vorrangig damit beschäftigt ist, den Projektgrund und das Kundenbedürfnis, die Stakeholder und Restriktionen für das Projekt zu analysieren und eine Vision für das Projekt zu erstellen [59, 60, 74]. In keiner der Publikationen wird die Aussage getroffen, dass das Projektteam die Anforderungen am Anfang vollständig aufnimmt.

Die Daten in Tabelle A.2 zeigen, dass in APH-Ansätzen zu Beginn mehr Planung als in

einem reinen agilen Projekt durchgeführt wird. Dies zeigt sich daran, dass zum Beispiel der Umfang des Projektes, das Budget und die benötigte Zeit am Anfang des Projektes festgelegt wird und Zeit- bzw. Meilensteinpläne erstellt werden. Dies findet in rein agilen Projekten nicht statt [16]. Im Gegensatz zu einer reinen plan-basierten Entwicklung, sind diese Pläne meistens nicht fest, sondern werden im Laufe des Projektes verfeinert und überprüft [61, 64]. Dadurch nutzen Firmen in APH-Ansätzen mehr Planung, lassen aber trotzdem Raum für Flexibilität.

Dadurch dass die Anforderungen am Anfang nur auf einem hohen Abstraktionslevel aufgenommen werden, gibt es während der Entwicklung ein kontinuierliches Requirements Engineering, um die Details der Anforderungen zu sammeln. Dies geschieht, indem Firmen da Wasserfall-Iteration oder das Pipeline Framework mit dem Wasserfall-Agil Framework kombinieren. Bei der Nutzung des Wasserfall-Iteration Frameworks werden am Anfang einer Iteration die Anforderungen für die aktuelle Iteration festgelegt [65, 73, 75]. Bei der Nutzung des Pipeline Frameworks werden die Anforderungen kontinuierlich und parallel zur Entwicklung erhoben. Dafür halten die Product Owner zum Beispiel regelmäßige Produktmanagementmeetings beschrieben, in denen neue Features eingebracht und diskutiert werden [68, 69].

**Architekturdefinition** Ähnlich zum Requirements Engineering beschreiben sechs Publikationen, dass die Architektur zu Beginn eines Projektes auf einem hohen Niveau definiert wird [59, 61–63, 67, 71]. Auch hier machen die Publikationen keine Angaben darüber, worin dieses hohe Niveau genau besteht. Keine der Publikationen trifft die Aussage, dass die Architektur vollständig für das gesamte Projekt bereits zu Beginn des Projektes definiert wird. Dies deutet darauf hin, dass Firmen die Definition der Architektur zwischen dem Beginn des Projektes und der Entwicklung aufteilen. Da die Publikationen keine Angaben darüber machen, wie detailliert die Firmen die Architektur zu Beginn definieren, haben wir auch hier die genannten Tätigkeiten untersucht, die zu Beginn eines Projektes von den Firmen durchgeführt werden. Die Liste der identifizierten Aktivitäten ist in Tabelle A.3 in Appendix A dargestellt.

Die Daten zeigen, dass während des Architekturdesigns zu Beginn eines Projektes der Fokus auf der Analyse der Schnittstellen und der verwendeten Technologie und Tools liegt. Zudem werden die benötigten Standards analysiert und die Komponenten festgelegt. Damit betreiben die Firmen in APH-Ansätzen mehr Architekturdefinition als in einem reinen agilen Projekt [39]. Die detaillierten Architekturentscheidungen werden, ähnlich zu agiler Entwicklung, während der Entwicklung getroffen. Heeager et al. [66] und Millard et al. [60] beschreiben die Rolle eines Architekten, der in einem Projekt die Aufsicht über die Weiterentwicklung der Architektur während der Implementierung hat. Die Aufgabe dieser Rolle ist, die Anforderungen hinsichtlich der Bedeutung für die Architektur zu analysieren und basierend darauf Entscheidungen über

die Weiterentwicklung der Entwicklung der Architektur auf Projektebene zu steuern.

**Beobachtung 5:** Firmen führen mehr Langzeitplanung und Architekturdefinition zu Beginn eines Projektes durch als in reinen agilen Ansätzen. Trotzdem führen sie die Anforderungsanalyse, Planung und Architekturdefinition zu Beginn auf einem hohen Niveau durch. Die Aufnahme von detaillierten Anforderungen oder detaillierten Architekturentscheidungen finden während der Laufzeit des Projektes statt.

Diese Beobachtung zeigt, dass die Firmen bei der Anforderungsanalyse, der Planung und der Architekturdefinition einen Mittelweg zwischen agiler und plan-basierter Entwicklung wählen. Sie betreiben mehr Planungsaufwand und Architekturdefinition als in agilen Ansätzen. Gleichzeit folgen sie nicht vollständig einem plan-basierten Ansatz. Würden die Firmen zum Beispiel einem plan-basierten Ansatz vollkommen folgen, würden sie die Architektur vollständig zu Beginn des Projektes vor der Implementierung definieren [76].

### 4.2.2 Forschungsfrage RQ2

In dieser Forschungsfrage geht es um die Untersuchung der Ziele, die Firmen in APH-Ansätzen haben und wie sie diese adressieren. Eine vollständige Liste der identifizierten Ziele und Maßnahmen und den Publikationen in denen diese genannt werden, findet sich in den Tabellen A.4 und A.5 in Appendix A. Auf der einen Seite nutzen Firmen agile Methoden vorrangig, um flexibel bei allen Arten von Änderungen zu sein [33, 77–81], mit unsicheren Anforderungen umzugehen [34, 76, 81–86] und um früher ausliefern zu können [43, 77, 81, 87–90]. Um dies zu erreichen, nutzen Firmen häufig ein iteratives Vorgehen [34, 79, 90]. Auf der anderen Seite nutzen Firmen plan-basierte Methoden vorrangig, um mit kritischen nicht-funtkionalen Anforderungen umzugehen und große Entwicklerteams zu managen [43, 80, 83–85, 91]. Sie nutzen dafür vorrangig Anforderungsanalyse und Architekturdesign zu Beginn des Projektes und Langzeitplanung [78–80, 82, 83, 92].

Diese Ziele und Maßnahmen sind typisch für die Anwendung von agilen und plan-basierten Methoden [5]. Dies bedeutet, dass Firmen APH-Ansätze absichtlich nutzen, um die Vorteile von agiler und plan-basierter Entwicklung zu kombinieren. Das heißt, Firmen sehen einen Nutzen in APH-Ansätzen und nutzen sie nicht nur, um zum Beispiel agile Entwicklung in ihre Firmenstruktur einzubetten.

#### 4.2.2.1 Kombination von Zielen

Um zu zeigen, dass die Kombination der Vorteile von agiler und plan-basierter Entwicklung tatsächlich existiert, haben wir die genannten Ziele in den Tabellen A.5 und A.4 hinsichtlich

häufiger Itemsets mit Hilfe des Apriori Algorithmus untersucht. Die Resultate dieser Untersuchung sind in Abbildung 4.7 dargestellt. Für die Generation der Itemsets wurde als Kriterium festgelegt, dass ein Tupel mindestens dreimal vorkommen muss. Hierbei wird eine möglichst hohe Zahl gewählt, um aussagekräftige Ergebnisse zu generieren. Durch die beschränkte Anzahl an Datenpunkten besteht jedoch bei der Mindestanzahl von vier bereits die Gefahr Tupel zu übersehen. Daher wurde sich für ein Mindestvorkommen von drei entschieden.

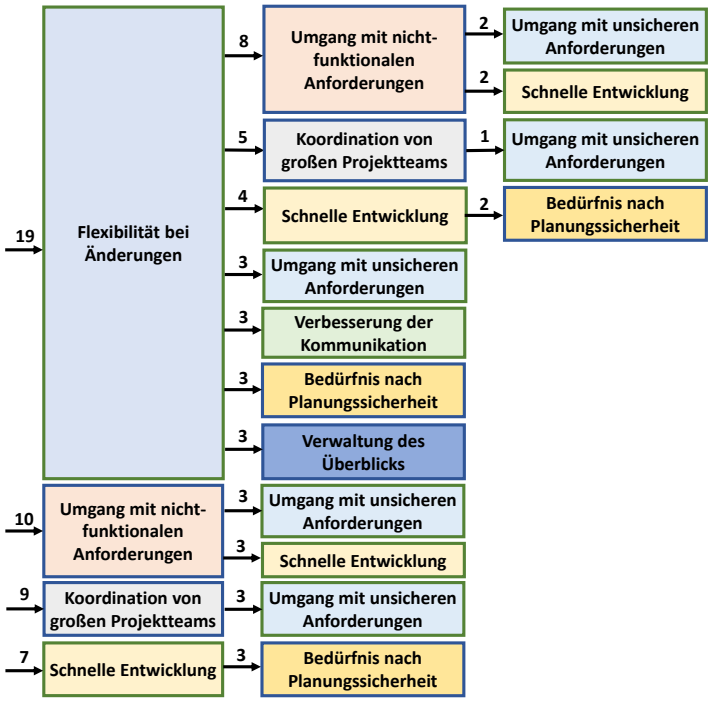


**Abbildung 4.7:** Resultate der Analyse von frequenten Itemsets unter den genannten Zielen (minimaler Support = 3). Die grünen Rechtecke repräsentieren agile Ziele und die blauen Rechtecke plan-basierte Ziele.

Die Abbildung 4.7 wird von rechts nach links gelesen und repräsentiert die gefundenen Tupel. Die Zahlen zeigen, wie häufig jedes Tupel vorkam. Zum Beispiel wurde die Kombination aus den folgenden Zielen zweimal gefunden: Flexibilität bei Änderungen, Umgang mit nicht-funktionalen Anforderungen und Umgang mit unsicheren Anforderungen. Wohingegen die Kombination der folgenden Ziele achtmal gefunden wurde: Flexibilität bei Änderungen und der Umgang mit nicht-funktionalen Anforderungen. Die angezeigten Kombinationen von drei Zielen erfüllen nicht das Kriterium für das Mindestvorkommen von drei. Jedoch alle enthaltenen Untertupel erfüllen das Kriterium, daher wurde sich hier entschieden die Kombinationen von drei Zielen mit aufzunehmen.

Bezüglich der Ergebnisse zeigt sich, dass agile und plan-basierte Ziele tatsächlich von den Firmen kombiniert werden. Agile Ziele werden von den Publikationen häufig in Kombinatio-

nen genannt. Dies deutet darauf hin, dass Firmen viele der Eigenschaften von Agilität nutzen wollen. Es wurde kein häufiges Itemset entdeckt, bei dem plan-basierte Ziele und Gründe in Kombination genannt werden. Dies bedeutet möglicherweise, dass Firmen häufig ein zentrales plan-basiertes Ziel haben, weswegen sie ein plan-basiertes Vorgehen benötigen und trotzdem möglichst viele der Eigenschaften von Agilität umsetzen wollen. Firmen, die zum Beispiel sicherheitskritische Anwendungen entwickeln, müssen trotzdem anpassungsfähig sein und mit unsicheren Anforderungen umgehen können.

**Beobachtung 6:** Firmen nutzen APH-Ansätze, um absichtlich die Vorteile von agiler und plan-basierter Entwicklung zu nutzen. Sie wollen dabei möglichst viele Eigenschaften von agiler Entwicklung nutzen, während sie häufig ein zentrales plan-basiertes Ziel verfolgen.

### 4.2.3 Widersprüche in APH-Ansätzen

Zusätzlich zu den Kombinationen der Ziele, haben wir die Auswirkungen der Kombination von Zielen untersucht. Dafür haben wir die Maßnahmen miteinander verglichen. Dabei wird ersichtlich, dass bei der Kombination von agilen und plan-basierten Maßnahmen Widersprüche auftreten. Diese entstehen dadurch, dass die agilen und plan-basierten Methoden ein unterschiedliches Vorgehen haben. Ein Beispiel dafür ist, wenn ein Ziel durch eine dezentrale Koordination und ein anderes Ziel durch eine zentrale Koordination adressiert wird. Wir haben daher die Maßnahmen auf solche Widersprüche hin untersucht. Dabei haben wir jegliche Maßnahmenkombinationen untersucht und nicht nur solche, die durch die identifizierten Zielkombinationen auftreten. Insgesamt haben wir fünf Aktivitäten identifiziert, in denen es durch gegensätzliche Maßnahmen zu Widersprüchen kommt.

#### Widerspruch im Requirements Engineering

Der erste identifizierte Widerspruch existiert im Requirements Engineering. Der Widerspruch existiert zwischen der Aufnahme und Analyse der Anforderungen zu Beginn des Projektes und der Aufnahme der Anforderungen iterativ während des Projektes. Der Widerspruch liegt demnach darin, wann die Anforderungen aufgenommen werden.

Für den Umgang mit Sicherheits, Security und Regulationsanforderungen nutzen Firmen mehr Vorab-Analyse der Anforderungen [78, 80, 81]. Vorab bedeutet hier, dass Firmen dies zu Beginn eines Projektes durchführen. Des Weiteren wird Vorab-Anforderungsanalyse zur Koordination genutzt, da diese die Koordination von großen Entwicklungsprojekten und verteilter Entwicklung unterstützt [70, 85, 89]. Zudem unterstützt es bei der Verwaltung des Überblicks und der Abhängigkeiten [90]. Firmen haben ein Bedürfnis nach Planungssicherheit [32, 69, 93].

Dafür ist es von Vorteil, viele der Anforderungen bereits am Anfang eines Projektes zu kennen [90].

Der Vorab-Analyse der Anforderungen steht jedoch das agile Vorgehen gegenüber, indem auf eine enge Zusammenarbeit mit den Kunden während des Projektes gesetzt wird und die Anforderungen während des Projektes gesammelt werden. Firmen wollen auf diese Weise ihre Businessausrichtung und die Zufriedenheit ihrer Kunden verbessern [78]. Zudem gehen sie auf diese Weise mit unsicheren Anforderungen und Änderungen um [34, 67]. Eine lange Anforderungsanalyse am Anfang verzögert zudem den Start der Entwicklung und steht dem Bedürfnis der Firmen nach einer kurzen Time-to-market und der frühen Schaffung von Wert gegenüber [86, 87].

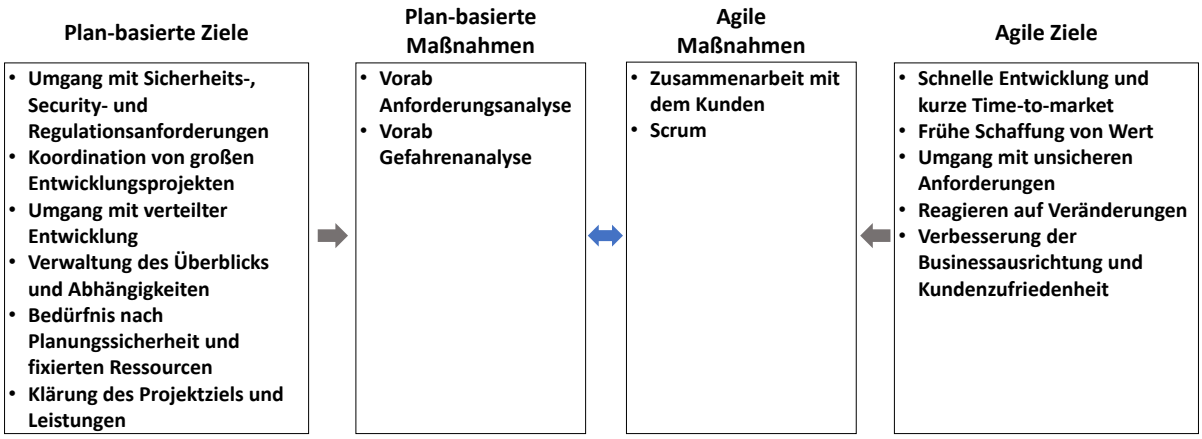


**Abbildung 4.8:** Widerspruch der Maßnahmen beim Requirements Engineering

### Widerspruch bei der Architekturdefinition

Der Widerspruch bei der Architektur existiert zwischen der Vorab-Architekturdefinition zu Beginn eines Projektes und einer agilen Architekturdefinition während des Projektes. Bei der Nutzung einer agilen Architektur werden die Designentscheidungen für die Architektur erst während der Entwicklung getroffen und es wird für Optionen geplant, um flexibel zu bleiben [76]. Der Widerspruch liegt demnach darin wann die Architektur definiert wird.

Die Gründe für eine Vorab-Definition der Architekt sind zum einen der Umgang mit Sicherheits, Security und Regulationsanforderungen [79, 83]. Es unterstützt bei der Verwaltung des Überblicks der Software und deren Abhängigkeiten [83, 94, 95]. Besonders in großen oder verteilten Projekten ist eine Vorab-Definition der Architektur hilfreich und unterstützt bei der Koordination [84, 89, 91].

Dagegen stehen die Gründe für eine agile Architektur. Nach Waterman et al. [39] ist eine agile Architektur notwendig, um auf Änderungen der Anforderungen reagieren und mit unsi-

cheren Anforderungen umgehen zu können. Eine Vorab-Definition der Architektur verzögert die Entwicklung, das dem Bedürfnis nach einer kurzen Time-to-market und frühem Wert gegenübersteht.

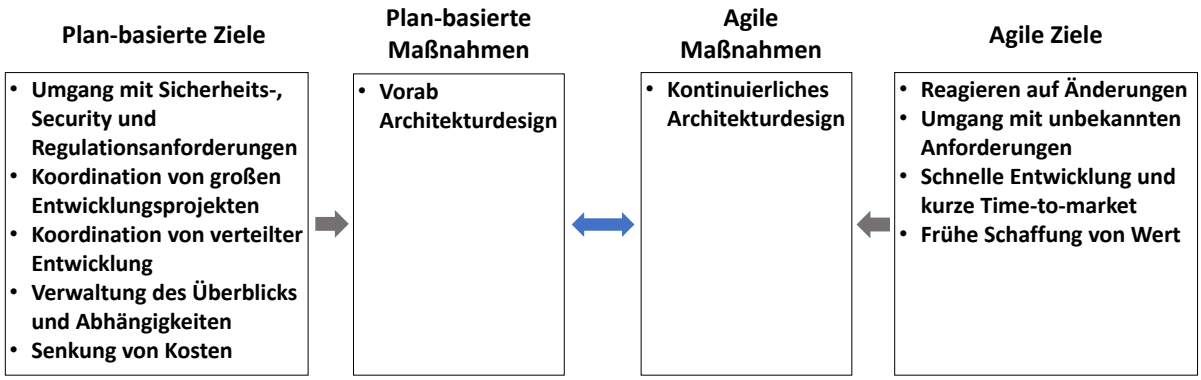


**Abbildung 4.9:** Widerspruch der Maßnahmen beim Architekturdesign

### Widerspruch in der Planung

Der nächste Widerspruch liegt in der Planung von Projekten. Der Konflikt existiert zwischen Langzeitplanung in plan-basierten Ansätzen und Kurzzeitplanung in agilen Ansätzen. Praktiken der Langzeitplanung sind die Vorab-Definition des Umfanges und der Projektrahmenbedingungen (Budget und Zeit) [77, 85], ein Fokus auf Planung und Kontrolle [32] und die Nutzung von Verträgen [85]. Dies bedeutet, dass durch die Langzeitplanung der Verlauf einen Projektes bereits festgelegt ist. Dagegen steht die Kurzzeitplanung durch Scrum, die Zusammenarbeit mit dem Kunden und einer kontinuierlichen Planung [78, 96]. Hierbei wird immer nur für einen kurzen Zeitraum geplant. Der Widerspruch besteht demnach darin, wie weit in die Zukunft geplant wird.

Die Gründe für die Nutzung von Langzeitplanung sind die Koordination von großen Projekten [82, 85], das Bedürfnis nach Planungssicherheit und fixierten Ressourcen [32, 77], die Kontrolle von Änderungen [80, 85] und die Klärung von Projektzielen und Leistungen [79, 82].

Dagegen stehen die Gründe für Kurzzeitplanung. Firmen wollen auf Änderungen reagieren können und damit die Businessausrichtung und die Kundenzufriedenheit erhöhen [33, 77, 78, 80]. Dafür wollen sie insgesamt flexibel sein und Pläne anpassen können [82]. Auch ist die Verteilung von Ressourcen häufig unsicher, wodurch eine kontinuierliche Planung nötig ist [96].

### Widerspruch in der Koordination

Der Widerspruch bei der Koordination entsteht, wenn zentrale Koordination aus plan-basierten und dezentrale Koordination aus agilen Ansätzen aufeinander treffen. Praktiken zentraler

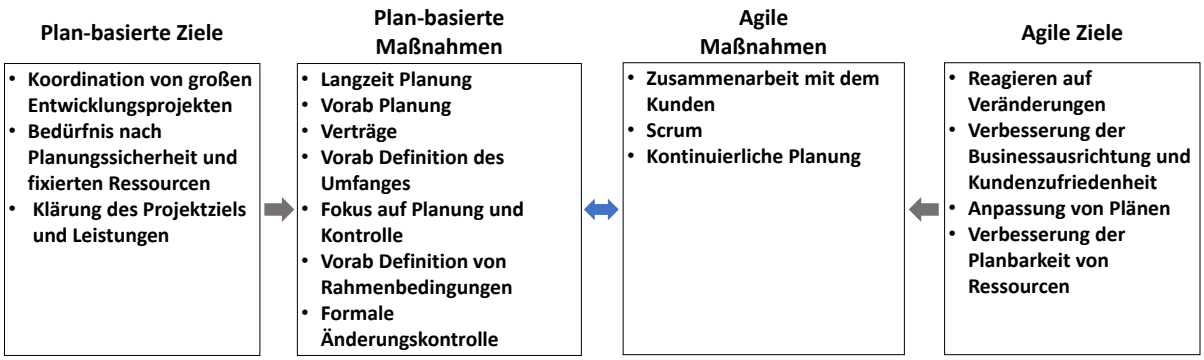


**Abbildung 4.10:** Widerspruch der Maßnahmen in der Planung

Koordination sind ein zentrales Team für Planung und Kontrolle und formale Kommunikation [32]. Dagegen stehen in agilen Ansätzen die Praktiken von dezentralen Entscheidungen und selbst-organisierten Teams. Der Widerspruch besteht also darin, wie Entscheidungen getroffen werden und die Teams organisiert sind.

Eine zentrale Steuerung unterstützt bei der Durchführung von großen und verteilten Projekten [82,92,97]. Somit wird zentrale Koordination für die Koordination von Projekten benötigt. Auf der anderen Seite sind selbst-organisierte Teams und ein dezentraler Entscheidungsprozess wichtige Faktoren, um auf Veränderungen reagieren zu können und eine schnelle Entwicklung zu haben.

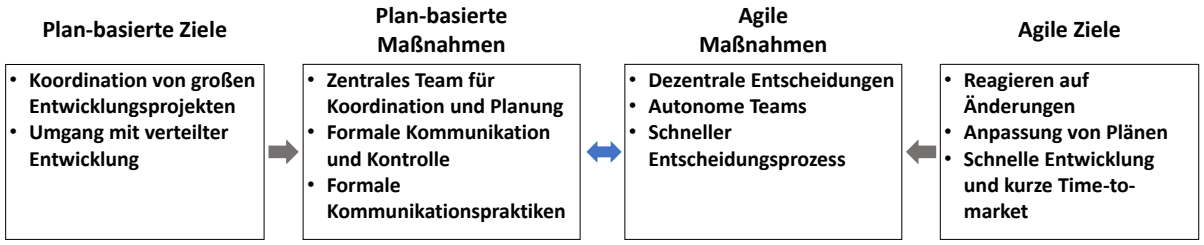


**Abbildung 4.11:** Widerspruch der Maßnahmen bei der Koordination

### Widerspruch im Testen

Der letzte identifizierte Widerspruch liegt im Bereich des Testens. Der Widerspruch existiert zwischen der Trennung der Testdurchführung und Entwicklerteams in plan-basierten Ansätzen und ganzheitlichen Teams aus agilen Ansätzen. In plan-basierten Ansätzen werden die Tests von den Entwicklerteams getrennt, in dem erst am Ende des Projektes die Software getestet wird oder durch den Einsatz des Pipeline Frameworks [40, 63]. Die Gründe hierfür sind zum einen besonders der Umgang mit Sicherheits-, Security- und Regulationsanforderungen, einer komplexen Entwicklung oder dem Testen von eingebetteter Software. Das Pipeline Framework

unterstützt auch bei der Durchführung der Tests in großen Entwicklungsprojekten.

Dagegen steht der Einsatz von ganzheitlichen Teams, in denen die Tests in den Entwicklerteams durchgeführt werden. Dies unterstützt dabei Probleme, früher zu erkennen und die zu Kommunikation unterstützen. Es führt ebenfalls dazu, dass Teams leichter auf Änderungen reagieren und sie schneller entwickeln können.

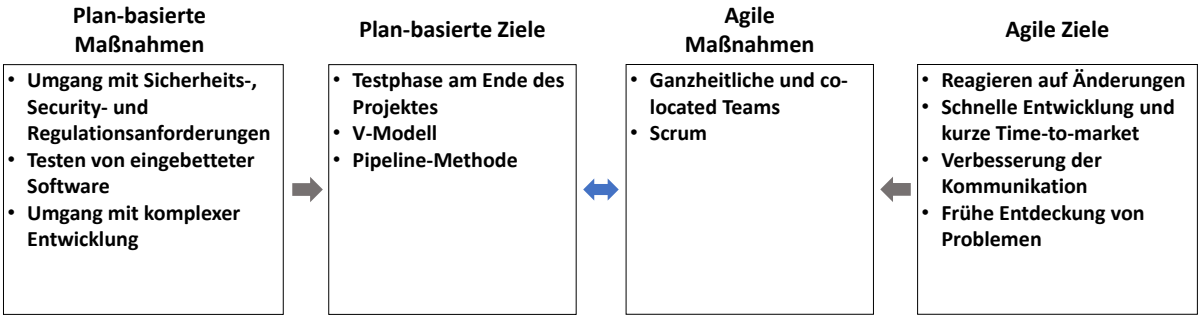


**Abbildung 4.12:** Widerspruch der Maßnahmen beim Testen

**Beobachtung 7:** Durch die Kombination der Vorteile (Ziele) von agiler und plan-basierte Entwicklung entstehen in APH Ansätzen Widersprüche zwischen dem Vorgehen von agilen und plan-basierten Methoden. Diese Widersprüche werden ausgelöst, wenn agile und plan-basierte Ziele aufeinandertreffen. Diese Widersprüche treten beim Requirements Engineering, bei der Architekturdefinition, in der Planung, beim Testen und bei der Koordination auf.

### 4.2.4 Forschungsfrage RQ3

In diesem Abschnitt werden die Schwierigkeiten und Probleme untersucht, die Firmen bei der Nutzung von APH Ansätzen haben. In Tabelle 4.3 sind alle Schwierigkeiten aufgelistet und nach der Häufigkeit der Nennung sortiert.

**Tabelle 4.3:** Schwierigkeiten in APH Ansätzen

| Schwierigkeit | # Nennungen | Publikationen |
|---|---|---|
| Management von Anforderungsänderungen | 6 | [34, 78, 80, 86, 90, 98] |
| Balance zwischen up-front und kontinuierlichem Requirements Engineering | 5 | [34, 69, 81, 86, 89] |
| Balance zwischen up-front und kontinuierlichem Architekturdesign | 5 | [43, 76, 79, 91, 95] |
| Balance zwischen zentraler Koordination und selbstorganisierten Teams | 5 | [32, 86, 88, 89, 99] |
| Schwierigkeiten durch eine separierte Durchführung der Tests | 5 | [34, 87, 90, 97, 100] |
| Spannungen zwischen dem Anforderungsteam und den Entwicklern | 3 | [81, 86, 88] |
| Balance zwischen Lang- und Kurzzeitplanung | 3 | [34, 88, 90] |
| Balance zwischen expliziter Dokumentation und Mitarbeiterwissen | 3 | [80, 86, 90] |

**Management von Anforderungsänderungen**

Diese Schwierigkeit ist die am häufigsten genannte. Im Laufe eines Projektes können Änderungen in den Anforderungen geschehen. Auf diese Änderungen muss das Projektteam reagieren. Die Schwierigkeit ist, dass häufig ein hoher Aufwand benötigt wird, um Anforderungen, die am Anfang beschlossen wurden, zu ändern, wie zum Beispiel bei Spezifikationen [34, 80, 98]. Auch die Untersuchung der Auswirkungen von Änderungen benötigen Aufwand und Zeit [90, 98]. In APH-Ansätzen existieren häufig verschiedene Anforderungsdokumente nebeneinander für die verschiedenen Abstraktionslevel von Anforderungen. In Fällen von Anforderungsänderungen müssen diese Dokumente über alle Abstraktionslevel geändert werden, damit sie synchronisiert sind [98]. Eine zusätzliche Schwierigkeit ist, dass es für die Firmen wichtig ist, schnell auf Anforderungsänderungen reagieren zu können, um die Entwicklung nicht zu verlangsamen [98].

**Balance zwischen vorab und kontinuierlichem Requirements Engineering**

Die Daten zeigen, dass Projekte mit einem APH-Ansatz sowohl ein vorab als auch ein kontinuierliches Requirements Engineering, bei dem die Anforderungen erst während der Entwicklung aufgenommen werden, benötigen. Die Schwierigkeit ist zu entscheiden, wie viel Aufwand in eine Vorab-Anforderungsanalyse investiert wird und in welchem Detailgrad die Anforderungen aufgenommen werden [34]. Zu wenig Aufwand am Anfang führt zu Anforderungen, die ein zu hohes Abstraktionslevel haben, nicht ausreichend verstanden und unzureichend für die Entwicklung sind [34]. Auf der anderen Seite sind Anforderungen häufig noch unsicher am Anfang des Projektes und werden erst sicherer während der Laufzeit des Projektes [81]. Daher ist es wichtig, nicht zu viele Anforderungen vorab festzulegen, da diese sich nochmal ändern könnten.

**Balance zwischen vorab und kontinuierlicher Architekturdefinition**

Die Daten zeigen ebenfalls, dass Firmen die Architektur sowohl vorab, als auch kontinuierlich definieren. Bei einer kontinuierlichen Architekturdefinition entwickelt sich die Architektur zusammen mit der Entwicklung. Das Risiko dieses Vorgehens ist, dass die Architektur auf eine eher zufällige Weise entsteht und nicht die beste Lösung präsentieren kann [76]. Wenn die Architektur jedoch in einen zu hohen Detailgrad zu Beginn definiert wird, erhöht dies das Risiko von kostenintensiven Überarbeitungen der Architektur auf Grund von unsicheren Anforderungen [76, 84, 85]. Die Schwierigkeit ist zu entscheiden, in welchem Detailgrad die Architektur am Anfang entwickelt wird und welche Entscheidungen zurückgehalten werden.

**Balance zwischen zentraler Koordination und selbst-organisierten Teams**

Die Daten zeigen, dass die Firmen Ziele verfolgen, die sowohl eine zentrale Koordination als auch eine dezentrale Koordination mit selbst-organisierten Teams verlangen. Selbst-organisierte Teams treffen selbstständig Absprachen und Entscheidungen untereinander [92]. Dies ermöglicht eine schnellere Entwicklung [88]. Die Gefahr ist jedoch, dass bei einer großen Anzahl von Teams Selbstorganisation chaotisch wird, da zu viele Personen involviert sind und den einzelnen Entwicklerteams der Überblick fehlt [89]. Die Firmen wollen ihren Teams genug Freiraum für Selbst-Organisation geben, gleichzeitig jedoch auch eine ausreichende Koordination sicherstellen [88]. Die Schwierigkeit ist, die Konzepte von Selbst-Organisation und zentraler Koordination gegeneinander abzuwägen und eine Balance zu finden.

**Schwierigkeiten durch eine separierte Durchführung der Tests**

Die Daten zeigen, dass die Firmen in einigen Fällen die Durchführung der Tests von den Entwicklerteams separieren, zum Beispiel durch die Nutzung des Pipeline Frameworks. Die Schwierigkeit ist es, mit den Fehlern umzugehen, die während der separierten Durchführung der Tests gefunden werden [34, 87, 97]. Die Behebung dieser Fehler unterbricht für die Entwicklerteams die Arbeit am aktuellen Inkrement [97]. Durch die Separierung des Tests von den Entwicklerteams, haben die Tester weniger Einblicke in die Entwicklung der Software und es fällt ihnen schwerer, Tests zu formulieren [90].

**Spannungen zwischen dem Anforderungsteam und den Entwicklerteams**

Die Daten zeigen, dass in einigen Fällen parallel zu den Entwicklerteams ein Anforderungsteam existiert, das die Verantwortung über die inhaltliche Steuerung des gesamten Projektes hat und die die Anforderungen und Stakeholder verwaltet. Dies ist zum Beispiel bei der Anwendung des Pipeline Frameworks der Fall. Die Schwierigkeiten entstehen durch die Trennung des Anforderungsteams von den Entwicklerteams. Das Anforderungsteam möchte gerne fixierte Zeiten von den Entwicklern genannt bekommen, zu denen bestimmte Features fertiggestellt sind, um die Entwicklung zu planen und Releases mit den Kunden abzusprechen [88]. In agilen Methoden wird jedoch kurzfristig geplant und es werden keine Deadlines für Features vereinbart. Das Anforderungsteam hat daher Sorge, die Kontrolle über die Entwicklerteams zu verlieren [88]. Dies führt zu Spannungen zwischen beiden Seiten.

Die Entwicklerteams auf der anderen Seite kritisieren häufig eine fehlende Priorisierung der Anforderungen und Feedbacks zu fertiggestellten Features von Seiten des Anforderungsteams. Das Anforderungsteam betrachtet häufig seine Arbeit als getan, wenn es die Anforderungen an die Entwicklung übergeben hat [86]. Die Entwicklerteams verlangen jedoch, dass das Anforde-

rungsteam kontinuierlich für Rückfragen zur Verfügung steht und eine kontinuierliche Zusammenarbeit besteht. In einigen Fällen fehlt jedoch ein gemeinsames Verantwortungsgefühl des Anforderungsteams und des Entwicklerteams für ein Projekt [88]. Die Nutzung eines Anforderungsteams führt zudem zu weniger direktem Kontakt zwischen den Entwicklern und den Kunden oder Nutzern [81]. Dies ist jedoch ein wichtiger Aspekt von agiler Entwicklung.

#### Balance zwischen Lang-und Kurzzeitplanung

Die Firmen nutzen Langzeitplanung, um Planungssicherheit für ihre Projekte herzustellen und um Projekte zu koordinieren [32, 85]. Projekte möchten aber gleichzeitig in der Lage sein, auf Änderungen reagieren zu können und mit Unsicherheiten umzugehen. Dafür benötigen sie Kurzzeitplanung [88]. Die Schwierigkeit für die Firmen ist es, zu entscheiden, wie viel Langzeitplanung und Kurzzeitplanung in Projekten nötig ist.

#### Balance zwischen expliziter Dokumentation und Mitarbeiterwissen

Agile Methoden legen den Fokus darauf, dass das Wissen in den Köpfen der Mitarbeiter gehalten wird, um den Entwicklungsprozess zu beschleunigen. Eine Vielzahl an Standards und Regularien verlangen auf der anderen Seite eine ausführliche Dokumentation von Anforderungen, Architektur und ähnlichem [80]. Heeager und Nielsen [80] beschreiben, dass es schwierig ist, einen dokumentations-getriebenen Ansatz mit agiler Entwicklung zu kombinieren. Die Schwierigkeit ist festzulegen, welche Informationen dokumentiert werden müssen und welche in den Köpfen der Mitarbeiter gehalten werden können [80, 90].

**Beobachtung 8:** Die meisten Schwierigkeiten in APH-Ansätzen betreffen die Balance von agiler und plan-basierter Entwicklung. Die Schwierigkeit bei der Balance besteht nicht in der Frage, ob bestimmte Praktiken angewendet werden, sondern in der Frage, wie diese angewendet werden müssen.

Die Analyse der Schwierigkeiten zeigt, dass es bei der Balance von agiler und plan-basierter Entwicklung zum Beispiel nicht um die Frage geht, ob das Wasserfallmodell angewendet werden sollte. Die Daten zeigen, dass die meisten Firmen dieses anwenden. Die Frage bei der Balance besteht stattdessen darin, wie es am besten angewendet werden kann, sodass die Firmen ihre Ziele erreichen. Die Schwierigkeiten für die Firmen bestehen in der Frage, wie stark ihr Fokus auf den Aktivitäten zu Beginn eines Projektes liegen sollte und wann sie in eine kontinuierliche Umsetzung der Aktivitäten wechseln sollten. Dies zeigt, dass Firmen bei der Kombination von agiler und plan-basierter Entwicklung entscheiden müssen, wie sie gewisse Methoden und Praktiken anwenden und nicht nur, ob sie diese anwenden.

## 4.3 Zusammenfassung

In diesem Kapitel wurde gezeigt, dass die Firmen die Vorteile von agiler und plan-basierter Entwicklung nutzen wollen und somit APH-Ansätze absichtlich nutzen. Des Weiteren haben wir gezeigt, dass die Kombination von agilen und plan-basierten Zielen Widersprüche im Vorgehen verursachen. Die Aktivitäten, in denen diese Widersprüche auftreten, sind Requirements Engineering, Architekturdefinition, Planung, Testen und Koordination. Die agile und plan-basierte Entwicklung in diesen Aktivitäten muss balanciert werden, um die Widersprüche aufzulösen und erfolgreich entwickeln zu können. Die Untersuchungen in RQ1 haben gezeigt, dass die Firmen in den Aktivitäten sowohl das agile als auch das plan-basierte Vorgehen anwenden und somit eine echte Kombination von beiden Ansätzen bilden. Hierfür nutzen sie das Wasserfallmodell als Grundgerüst und nutzen, während der Entwicklung, iterative Ansätze. Es zeigt auch, dass die reine Untersuchung von angewendeten Methoden in der Erforschung von APH-Ansätzen nicht ausreicht, da zum Beispiel das Wasserfallmodell unterschiedlich angewendet werden kann. Zum Beispiel können die Anforderungen vollständig zu Beginn erhoben werden oder nur auf einem abstrakten Level.

Die Analyse der Schwierigkeiten zeigt, dass die Firmen Schwierigkeiten haben eine Balance zwischen dem Vorgehen zu erreichen, die für eine Kombination nötig ist. Auch bei der Analyse der Schwierigkeiten zeigt sich, dass für eine Balance nicht die Frage entscheidend ist, ob gewisse Praktiken angewendet werden, sondern wie sie angewendet werden. Bestehende Ansätze für eine Unterstützung bei der Kombination von agiler und plan-basierter Entwicklung sind auf die Auswahl von Methoden und Praktiken fokussiert. Wir haben in der Literaturstudie wenige Informationen darüber gefunden, wie zum Beispiel das Wasserfallmodell genutzt werden kann, damit Firmen ein agiles und plan-basiertes Vorgehen balancieren können. In dem folgenden Kapitel wird daher untersucht, wie Firmen eine Balance zwischen agiler und plan-basierter Entwicklung erreichen.

# 5

# Analyse der Balance von agiler und plan-basierter Entwicklung

Das vorherige Kapitel beschreibt Widersprüche, die durch die gegensätzlichen Ziele von Firmen in APH Ansätzen (Agil Plan-basierter Hybrider Entwicklungsansatz) auftreten. Die Analyse der Quellen aus der Literaturrecherche zeigt auch, dass Firmen Schwierigkeiten dabei haben, eine Balance zwischen agiler und plan-basierter Entwicklung herzustellen und die Widersprüche aufzulösen. Die Literaturrecherche erbrachte jedoch keine Informationen darüber, wie eine Balance von den Firmen erreicht werden kann. Um diese Forschungslücke zu schließen, habe ich in Zusammenarbeit mit zwei weiteren Forschern[1] eine Interviewstudie durchgeführt. Die Ergebnisse in diesem Kapitel wurden in [4] veröffentlicht und auf der *International Conference on Software and System Processes* präsentiert.

## 5.1 Forschungsvorgehen

Die verwendete Forschungsmethode für diese Studie ist Grounded Theory. Die Anwendung von Grounded Theory verlangt nicht nach einer konkreten Forschungsfrage, sondern fordert,

[1] Jil Klünder und Kurt Schneider

dass Konzepte aus den Daten hervortreten [101, 102]. Diese repräsentieren die wichtigsten Themen der Teilnehmer.

### 5.1.1 Studiendesign und Datenerhebung

Für die Datensammlung in dieser Studie haben wir Interviews mit Praktikern aus Firmen durchgeführt, die einen APH-Ansatz nutzen. Im Vergleich zu einer Umfrage ermöglicht eine Interviewstudie auf interessante Punkte im Detail einzugehen. Die Suche nach Teilnehmern erstreckte sich über Personen, die mit einem APH-Ansatz arbeiten oder mit einem solchen gearbeitet haben. Da die Balance von APH-Ansätzen verschiedene Aktivitäten in einem Entwicklungsansatz betrifft, bezog die Suche verschiedene Rollen wie Projektmanager, Product Owner, Entwickler und Tester mit ein. Um Teilnehmer für die Studie zu sammeln, haben wir die Industriekontakte des Fachgebiets Software Engineering an der Leibniz Universität Hannover angeschrieben und Anfragen auf LinkedIn verteilt. Insgesamt umfasst die Interviewstudie 15 Personen in Deutschland, die mit einem APH-Ansatz arbeiten. Tabelle 5.1 zeigt eine Übersicht der Interviewteilnehmer mit ihrer Erfahrung, Rolle, dem Businessbereich ihres Projektes und der Projektgröße.

**Tabelle 5.1:** Demographie der Interviewteilnehmer

| ID | Erfahrung (in Jahren) | Rolle | Projekt | Projektgröße (# an Mitarbeitern) |
|---|---|---|---|---|
| 01 | 25 | Projektmanager / Softwarearchitekt | IT Dienste für Banken | 5 - 20 |
| 02 | $>30$ | Projektmanager | IT Dienste für Banken | 30 - 50 |
| 03 | $>25$ | Produktmanager | IT Dienste für Banken | 15-20 |
| 04 | 0.5 | Softwarearchitekt | IT Dienste für Versicherungen | 20 -30 |
| 05 | $>10$ | Tester | IT Dienste für Banken | 20 |
| 06 | 10 | Entwickler | Automobilzulieferung | 4 |
| 07 | 25 | Product Owner / Softwarearchitekt | Software für Satelliten | 10 |
| 08 | $>4$ | Entwickler | Nutzerverwaltungssystem | 3 |
| 09 | 35 | Agile Coach | IT Dienste für Versicherungen | 30 |
| 10 | $>5$ | Ressourcenmanager | IT Dienste für Geschäftsprozesse | 50 |
| 11 | 8 | Agile Coach | IT Dienste für Geschäftsprozesse | 5 -10 |
| 12 | 13 | Entwickler / Softwarearchitekt | IT Dienste für eine Bahnunternehmen | 90 |
| | | | IT Dienste für Versicherungen | 5 |
| 13 | 7 | Agile Coach | IT Dienste für Logistik | 7 - 11 |
| 14 | $>10$ | Agile Coach | IT Dienste für Versicherung | 10 - 30 |
| 15 | $>30$ | Product Owner | Automobilzulieferer | 60 - 80 |

Die Datenerhebung erfolgt durch semi-strukturierte Interviews. Diese bieten eine Struk-

tur und einen roten Faden für die Interviews. Gleichzeitig ermöglichen sie es, interessante Punkte weiter zu vertiefen und hindern die Interviewteilnehmer nicht daran, wichtige Punkte zu nennen [103]. Die Konstruktion der Interviews basiert auf den Guidelines von Runeson und Höst [104] sowie Hove und Benta [103]. Die Durchführung der Interviewstudie lag zwischen April und Dezember 2021 und die Interviews dauerten durchschnittlich 60 Minuten. Das längste Interview betrug 82 Minuten und das kürzeste 48 Minuten. Die Interviews wurden von uns aufgenommen und anschließend transkribiert. Ein Interview konnte auf Grund von technischen Problemen nicht aufgenommen werden. In diesem Fall hat der Interviewer jedoch alle wichtigen Informationen vom Interviewten handschriftlich festgehalten. Die Sammlung und Analyse der Daten erfolgte iterativ, um auftretende Themen berücksichtigen zu können. Für die Interviews haben wir einen Leitfaden definiert. Dieser findet sich in Appendix B.

Der Anfang der Interviews bezieht sich auf den Hintergrund, Erfahrung und das Projekt in dem die Interviewteilnehmer involviert sind. Dies dient dazu, ein Verständnis für den Kontext des Interviewteilnehmers zu erlangen. Die anschließenden Fragen beschäftigen sich mit der Rolle des Interviewteilnehmer im Projekt und den Aufgaben und Verantwortlichkeiten, die diese Person hat.

Der zweite Fragenblock beschäftigt sich mit dem Vorgehen des Projektteams im Projekt. Dieser beinhaltet Fragen über das allgemeine Vorgehensmodell (agil, plan-basiert oder hybrid) des Projektes. Dazu erfolgten Fragen darüber, welche Aktivitäten zu Beginn, während und am Ende eines Projektes von dem Projektteam durchgeführt werden. Zusätzlich haben wir Fragen darüber gestellt, welche Praktiken, Methoden und Artefakte in den Projekten eingesetzt werden und wie diese vom Projektteam angewendet und verbunden werden. Zusätzlich beinhaltete dieser Block Fragen zu den Gründen und Vorteilen bei der Nutzung der genannten Methoden, Praktiken und Artefakte.

Der dritte Fragenblock befasst sich mit der Balance in APH-Ansätzen. Dabei haben wir gefragt, in wie weit plan-basierte und agile Methoden und Praktiken für die Aktivitäten von Requirements Engineering, Architektur, Planung, Testen, Dokumentation und Koordination genutzt werden. Nach den Untersuchungen in der Literaturstudie, sind dies die Aktivitäten, die balanciert werden müssen. Wir haben zusätzlich gefragt, welche Vorteile der Einsatz von plan-basierten und agilen Praktiken bei der Durchführung der Aktivitäten bringen und wann diese schädlich für eine Aktivität sind.

Der letzte Fragenblock beschäftigt sich damit, die Erkenntnisse aus der Literaturrecherche zu erweitern. Daher beziehen sich die Fragen in diesem Block auf Probleme und Schwierigkeiten, die die Interviewteilnehmer haben und noch nicht durch das Interview abgedeckt wurden.

### 5.1.2 Datenanalyse

Der Kodierungsprozess von Grounded Theory besteht aus Open, Axial, Theoretical und Selective Coding und wurde vom Verfasser dieser Arbeit durchgeführt. Im Folgenden ist der Kodierprozess an Hand eines Beispiels beschrieben. Dafür wird die *Struktur in der Koordination* als eine der Hauptkategorien verwendet. Das Beispiel ist in Abbildung 5.1 dargestellt.

Der erste Schritt des Kodierprozesses besteht darin, die Transkripte in unabhängige Abschnitte einzuteilen, die jeweils ein eigenes Thema beschreiben. Das Thema des Abschnittes wird mit *Stichworten* zusammengefasst. Anschließend werden den Stichworten *Codes*, die die Stichworte mit zwei oder drei Worten beschreiben, zugewiesen. Das folgende Beispiel zeigt einen solchen Abschnitt:

**Rohdaten:** *Interviewteilnehmer 12: "Die Selbstorganisation endet meist da, wo wir eine fundamentale Frage haben, die nicht nur unser Team betrifft, sondern auch andere Teams. Meistens sind es fundamentale technische Sachen, weil die fachlichen Anforderungen meistens ein spezielles Team betreffen. Fundamentale Fragen gehen meistens nach oben (Projektleitung), das würden wir ungern selber organisieren."*
**Stichworte:** *Selbst-Organisation endet an fundamentalen technischen Themen, die mehrere Teams betreffen.*
**Code:** *Teameinschränkung*

In diesem Beispiel beschreibt der Interviewteilnehmer, dass bei fundamentalen Fragen, die das gesamte Projekt und nicht nur ein einzelnes Entwicklerteam betreffen, die Projektleitung in das Klären der Frage mit einbezogen wird. Daher wurde der Code *Teameinschränkung* diesem Abschnitt zugewiesen. Nach dem Open Coding Prozess wurde Axial Coding auf die Codes angewendet, um Konzepte durch ständiges Vergleichen der Codes zu bilden. Aus diesen und ähnlichen Codes wurde das Konzept *Einschränkungen der Selbstorganisation* gebildet.

Der nächste Kodierschritt besteht aus dem Vergleich der entstandenen Konzepte. In diesem Prozess werden für das nächst höhere Abstraktionslevel Kategorien gebildet. Während der Analyse der Konzepte zeigt sich, dass Selbst-Organisation durch Einschränkungen definiert ist. Zum Beispiel ist oben die Einschränkung beschrieben, dass Entwicklerteams bei Fragen, die das gesamte Projekt betreffen, in ihrer Selbst-Organisation eingeschränkt werden.

Nach dem Axial Coding folgt das Selective Coding. Dieses dient dazu, die Kernkategorie zu identifizieren. Bei der Analyse zeigt sich, dass die Firmen plan-basierte Elemente in ihrem Prozess nutzen, um die agile Entwicklung zu strukturieren. Mit plan-basierten Methoden limitieren die Firmen den Raum, in dem sich ein Projekt bewegen kann. Die Kernkategorie trägt daher den Namen *Struktur für die agile Entwicklung*.


**Abbildung 5.1:** Beispiel für Codes, Konzepte und Kategorie

Der letzte Schritt des Codingprozesses besteht aus Theoretical Coding, bei dem die Beziehungen zwischen den verschiedenen Konzepten und Kategorien analysiert werden. Dieser Schritt wird weiter unterstützt durch den Einsatz von *Memoing*. Memoing ist eine Technik, bei der wichtige Gedanken über den Zusammenhang der Daten während des Codingprozesses festgehalten werden [101, 102]. Im präsentierten Beispiel zeigt sich, dass die Architektur der Software einen Einfluss auf die Selbst-Organisation der Teams hat. Je mehr Abhängigkeiten die Entwicklerteams zueinander oder zu anderen Projekten haben, desto weniger selbstorganisiert sind sie.

### 5.1.3 Bedrohung der Validität

Grounded Theory gehört zur empirischen Forschung. Daher wird hinsichtlich der Bedrohung der Validität, nach Maxwell [58] deskriptive, theoretische und interpretative Validität und die Generalisierbarbeit betrachtet.

**Deskriptive Validität** Deskriptive Validität betrachtet die Korrektheit der gemachten Beobachtungen. Um diese Bedrohung zu minimieren, wurden die Interviews aufgenommen und für die nachfolgende Analyse transkribiert. Ein Interview konnte aus technische Gründen nicht aufgenommen werden. Um trotzdem alle relevanten Daten aufzunehmen, wurden während des Interviews Notizen gemacht.

**Theoretische Validität** Theoretische Validität betrachtet, inwieweit die relevanten Aspekte gesammelt wurden. Eine Interviewstudie, die sich über mehrere Fälle erstreckt, ermöglicht es, Einblicke in unterschiedlichen Firmen zu erlangen. Die Anwendung von APH-Ansätzen in unterschiedlichen Firmen kann so miteinander verglichen werden. Dies ermöglicht einen tieferen Einblick in die Nutzung von APH-Ansätzen. Durch semi-strukturierte Interviews ist es möglich, während des Verlaufs des Interviews auf interessante Punkte einzugehen. Dies minimiert die Gefahr relevante Punkte zu übersehen. Die Interviewfragen sind aus der vorangegangenen Literaturrecherche abgeleitet. Dies birgt die Gefahr von Bias, da sich die Interviews auf die identifizierten Aspekte fokussieren. Um diesen Bias zu minimieren, haben wir am Ende des Interviews offene Fragen nach Schwierigkeiten und anderen wichtigen Punkten gestellt, die nicht bereits durch das Interview abgedeckt wurden. Zusätzlich haben wir die Interviewfragen zwei Forschern im Bereich von agiler und hybrider Entwicklung zur Prüfung gegeben und ihre Anmerkungen eingearbeitet. Des Weiteren erfolgte die Datensammlung iterativ, um auf neue Themen im Verlauf der Interviewstudie reagieren zu können. Um unverfälschte Antworten zu bekommen, folgen die Interviews den Richtlinien von Runeson und Höst [104] und Hove und Benta [103]. Insbesondere haben wir in allen Fällen die Anonymität der Teilnehmer versichert.

**Interpretative Validität** Die interpretative Validität betrachtet die gezogenen Schlüsse aus den Daten. Um sicherzustellen, dass die Datenauswertung nach einem strukturierten Prozess erfolgt, wurde Grounded Theory als Forschungsmethode gewählt. Grounded Theory bietet einen genauen Codierprozess, der während der Datenauswertung verwendet wurde. Der Kodierungsprozess wurde vom Autor dieser Arbeit durchgeführt. Dies kann jedoch zu Bias führen, da nur ein Forscher verantwortlich für die Kodierung ist. Um diesen Bias zu minimieren, hat die zweite involvierte Forscherin den gesamten Kodierungsprozess überprüft. Dazu hat sie die Codes, Konzepte und Kategorien untersucht, die der Autor dieser Arbeit aus den Aussagen extrahiert hat.

**Generalisierbarbeit** Bei der Auswahl der Interviewteilnehmer haben wir darauf geachtet, verschiedene Geschäftsbereiche, Projektgrößen und Rollen in die Interviewstudie einzubeziehen. Trotzdem deckt diese Interviewstudie nicht alle möglichen Kontexte und Rollen ab. Zudem besteht die Interviewstudie aus 15 Teilnehmern, die alle aus Deutschland kommen. Dies verringert die Generalisierbarbeit der Resultate.

## 5.2 Resultate

Aus der Analyse der Kernkategorie ergeben sich die beiden zentralen Ziele, die Firmen mit einem APH-Ansatz verfolgen. Firmen wollen auf der einen Seite flexibel sein, um auf Änderungen und Schwierigkeiten reagieren zu können. Auf der anderen Seite benötigen sie zum Beispiel für Koordination und Planungssicherheit in ihrem Projekt Stabilität und Struktur in der Entwicklung. Für eine Balance nutzen sie Planung, Anforderungsanalyse und Architekturdefinition zu Beginn eines Projektes, um ein grundlegendes Maß an Struktur und Stabilität zu erreichen. Die Firmen achten jedoch darauf, in diesen Strukturen genug Freiräume zu lassen, damit das Projektteam trotzdem flexibel sein kann. Im Folgenden wird für die Aktivitäten Planung, Anforderungsanalyse, Architektur, Testen und Koordination beschrieben, wie sie diese Balancen im Detail erreichen.

**Beobachtung 1:** Die zwei Hauptantriebe für die Nutzung von APH-Ansätzen sind, dass Firmen Struktur und Stabilität in der Entwicklung benötigen und gleichzeitig flexibel sein wollen. Sie nutzen vorwiegend up-front Tätigkeiten aus den plan-basierten Methoden, um eine Struktur zu erreichen. Die Firmen lassen in den Strukturen genug Freiräume um flexibel zu bleiben.

### 5.2.1 Balance in der Planung

Die Kunden in einem Softwareprojekt benötigen für ihre eigene interne Planung Angaben über das Budget und die Dauer eines Softwareprojektes. Auch die Softwarefirmen selbst benötigen diese Angaben, um Ressourcen und insbesondere ihr Personal innerhalb eines Unternehmens verteilen zu können. Dazu führen die Firmen, basierend auf dem ermittelten Umfang für das Projekt, eine initiale grobe Schätzung der benötigten Zeit, des Budget und der Skills für das Projekt durch. Durch die Erstellung von Langzeitplänen stellen sie sicher, dass sie diese Schätzung einhalten.

*Interviewteilnehmer 06: ”Der Zeitraum [für das Projekt] kam tatsächlich vom Kunden. Sie wollen bis dann und dann ein Ergebnis haben[...]. Und dann kann man halt zurückrechnen: Ok ein Jahr vorher sollen die Feldtests starten und dann weiß man bis dahin muss das auf jeden Fall fertig sein. ”*

Zudem unterstützt Langzeitplanung dabei, Abhängigkeiten in Projekten und zu anderen Projekten zu berücksichtigen und zu planen. Dafür werden von den Firmen Roadmaps eingesetzt, da sie einen Überblick über das Projekt geben und so für Stabilität sorgen. Wenn mehrere Teams an einem Projekt beteiligt sind, kann eine Roadmap dabei helfen zu zeigen, wie die verschiedenen Teile des Projektes zusammenhängen und wo Abhängigkeiten zu anderen Projekten existieren.

*Interviewteilnehmer 11: ”Besonders, wenn du Dinge aus Projektmanagementsicht hast, die ineinandergreifen. Da den Überblick, besonders in großen Projekten zu behalten, dafür sind diese Roadmaps super hilfreich. ”*

Auf der anderen Seite wollen die Firmen flexibel sein, da es immer Unsicherheitsfaktoren während der Entwicklung gibt. Bei zu detaillierten Langzeitplänen müssen die Projektteams die Pläne daher immer wieder überarbeiten.

*Interviewteilnehmer 12: ”Je enger man das Raster für die Planung legt, desto eher wird man auch danebenliegen und sie häufig revidieren müssen. ”*

Für die Balance zwischen genug Planung und Flexibilität konnten wir verschiedene Mechanismen identifizieren, die die Firmen nutzen. Wir haben acht Fälle entdeckt, in denen das Projektteam zusammen mit dem Kunden einen Toleranzbereich, in dem die tatsächliche Dauer und Kosten des Projektes liegen können, für die Schätzung festlegen. Daher kann in diesen Fällen von einer groben Schätzung zu Beginn des Projektes gesprochen werden. Dadurch erzeugt das Projektteam eine Struktur für das Projekt, da grob feststeht, was entwickelt werden soll. In Fällen von Änderungen muss jedoch nicht immer die gesamte Schätzung neu berechnet werden. Das Projektteam ist zudem flexibel, solange es sich in diesem Toleranzbereich bewegt.

*Interviewteilnehmer 13: ”Bei der (Anforderungsanalyse) kommt dann auch raus: Ok, wir haben verstanden, was ihr so braucht. Wenn wir mal diese Epics hochrechnen, dann werden wir wahrscheinlich Kosten von einer Million haben. Und so wie die Diskussion verlaufen ist, haben wir eine Unschärfe von plus minus 30 %.”*

Um zu überprüfen, ob das Projekt sich insgesamt innerhalb dieses Toleranzbereichs bewegt oder droht darüber hinaus zu laufen, überwachen die Projektteams kontinuierlich den Projektstatus. Dieses Überwachen des Projektstatus wird von fünf Interviewteilnehmern explizit genannt.

*Interviewteilnehmer 13: ”Du musst immer gucken: Passt der Projektstatus noch zu dem Budgetrahmen, den wir uns gesetzt haben. ”*

Um bei der Nutzung von Langzeitplanung genug Flexibilität zu behalten, berichten 6 Interviewteilnehmer, dass sie diese nur nutzen, um Abhängigkeiten und die Entwicklungsreihenfolge von Epics zentral zu planen. Die Arbeit der Entwickler in den einzelnen Entwicklerteams wird nicht zentral durch eine Roadmap geplant. Die Entwicklerteams planen ihre Arbeit selbstständig, indem sie ihre Sprintbacklogs füllen.

*Interviewteilnehmer 11: "Eine Roadmap verändert sich ja und die verändert sich extremst schnell. Deshalb sagt man ja auch, dass das Top-Backlog ausspezifiziert ist. Darunter, je weiter ich komme, desto mehr leere Hüllen habe ich. Je feiner du eine Roadmap machst, desto mehr totdokumentierst du dich."*

**Beobachtung 2:** Die Firmen schaffen durch eine initiale Schätzung und die Definition einer Roadmap Planungssicherheit und Stabilität für ihre Projekte. Für genug Flexibilität versehen sie die Schätzung mit einem Toleranzbereich und berücksichtigen in der Roadmap nur den Ablauf für die Hauptfeatures und Abhängigkeiten zu anderen Projekten.

Die initiale Analyse des Umfanges und die daraus resultierende Schätzung gibt dem Projektteam und dem Kunden Stabilität. Wie flexibel ein Projekt dabei ist, hängt davon ab, wie flexibel der Umfang vom Projektteam gehandhabt werden kann. Zum Beispiel ermöglicht ein flexibler Umfang durch das Fallenlassen von einzelnen Features, den Aufwand in einem Projekt zu verringern und so auf Probleme zu reagieren. Zudem ermöglicht er es auf Anforderungsänderungen zu reagieren, in dem Features ausgetauscht werden. Ein starrer Umfang, der zu Beginn eines Projektes festgelegt wird, ermöglicht einem Projektteam wenig Flexibilität, da das Projektteam keine Features fallen lassen kann.

Die Analyse der Daten hat vier unterschiedliche *Modi* offenbart, die eine unterschiedlich starke Flexibilität beim Umgang mit dem Umfang darstellen. In der letzten Spalte von Tabelle 5.2 ist der jeweilige Modus angegeben, der von den Projekten verwendet wird. Die Modi formen ein Spektrum von einer agilen Arbeitsweise hin zu einer plan-basierten Arbeitsweise. Bei einer agilen Arbeitsweise wird zu Beginn eines Projektes auf die Schätzung des Gesamtprojektes verzichtet [16].

Auf der einen Seite des Spektrums und damit am nächsten an einem puren agilen Vorgehen ist der *Flexibilitätsmodus*. In diesen Fällen ermitteln die Projektteams den Anforderungsumfang, der vom Kunden für ein Projekt geplant ist. Basierend auf diesem geplanten Umfang führen die Projektteams eine Schätzung des Projektes durch. Zusätzlich geben sie einen Toleranzbereich für die Schätzung an. Für den Verlauf des Projektes wird dieser Umfang trotzdem flexibel gehalten. Dies ermöglicht es dem Kunden, einfach Features auszutauschen und Änderungen einzubringen, ohne dass Verträge extra geändert werden müssen. Im Verlaufe des Projektes wird in diesem Modus die Priorität von Features kontinuierlich analysiert und angepasst. Dieser Modi ist daher für Fälle geeignet, in denen die Anforderungen eher unsicher sind.

**Tabelle 5.2:** Verwendete Frameworks und Planungsmodi in den Fällen

| ID | Frameworks | Modus |
|---|---|---|
| 01 | Wasserfall-Agil Framework + Pipeline Framework | Flexibilitätsmodus |
| 02 | Wasserfall-Agil Framework + Pipeline Framework + Wasserfall-Iteration Framework | Meilensteinmodus |
| 03 | Pipeline Framework | Iterationsmodus |
| 04 | Wasserfall-Agil Framework + Pipeline Framework | Meilensteinmodus |
| 05 | Wasserfall-Agil Framework + Pipeline Framework | Managementmodus |
| 06 | Wasserfall-Agil Framework | Managementmodus |
| 07 | Pipeline Framework | Meilensteinmodus |
| 08 | Wasserfall-Agil Framework + Pipeline Framework | Meilensteinmodus |
| 09 | Wasserfall-Agil Framework + Pipeline Framework | Flexibilitätsmodus |
| 10 | Wasserfall-Agil Framework | Managementmodus |
| 11 | Wasserfall-Agil Framework + Pipeline Framework | Meilensteinmodus |
| 12 | Wasserfall-Agil Framework + Pipeline Framework + Wasserfall-Iteration Framework | Meilensteinmodus |
| | Wasserfall-Iteration Framework | Managementmodus |
| 13 | Wasserfall-Agil Framework + Pipeline Framework | Flexibilitätsmodus / Interationsmodus |
| 14 | Wasserfall-Agil Framework + Pipeline Framework | Flexibilitätsmodus |
| 15 | Pipeline Framework | Meilensteinmodus |

*Interviewteilnehmer 01: "Der wesentliche Unterschied zu einem klassischen Projekt ist, dass Sie [...] eben nicht planen wir wollen bis dann und dann die Inhalte zu hundert Prozent umgesetzt haben. Sondern Sie sagen am Anfang: Ok das dauert Pi mal Auge sagen wir mal anderthalb Jahre und das sind irgendwie so die Skills, die wir benötigen [...]."*

Der zweite Modus ist der *Meilensteinmodus*. In diesen Fällen sind Firmen mit einigen Anforderungen konfrontiert, die einen konkreten Umfang zu einer konkreten Deadline haben. Um auf Probleme während der Entwicklung dieser festen Anforderungen reagieren zu können, werden immer einige flexible Anforderungen mit aufgenommen. Diese sollten, müssen aber nicht, umgesetzt werden und können im Notfall fallen gelassen werden. Diese flexiblen Anforderungen werden um die festen Anforderungen herum geplant. Dafür analysieren die Firmen kontinuierlich, welche Anforderungen vorgezogen oder zurückgestellt werden können. Dies hilft ihnen auch dabei, die ganze Projektlaufzeit ihr volles Entwicklungspotenzial auszuschöpfen.

*Interviewteilnehmer 08: "Wir haben es mit Verhandlungen hinbekommen, dass es erstmal nur um das Identifizieren geht, und alles Weitere im Nachhinein gemacht wird. Das Gesetz schreibt nicht vor, dass man über verschiedene Rechte irgendwelche Leute verwalten muss. Es schreibt aber vor, dass sich die Leute identifizieren müssen. Deshalb haben wir diese Sache vorgezogen."*

Der dritte Modus ist der *Iterationsmodus*. In diesen Fällen benötigen Firmen häufig ein hohes

Maß an Planungssicherheit, z.B. weil der Kunde einen genauen Preis zu bestimmten Features benötigt oder das Projekt viele Absprachen mit anderen Projekten treffen muss. Gleichzeitig können die Anforderungen jedoch nicht zu weit in die Zukunft geplant werden, da sie dafür zu unsicher sind. Die Lösung in diesen Fällen besteht darin, dass die Firmen den Umfang für eine Iteration genau festlegen und planen. Manchmal nutzen die Firmen sogar Verträge für diese Iterationen, indem ein genaues Budget, zusammen mit einer zu erbringenden Menge an Features festgelegt wird. Die Länge einer Iteration liegt dabei zwischen drei und sechs Monaten. Mit dieser Lösung erreichen die Firmen das hohe Bedürfnis nach Planungssicherheit, sind aber trotzdem noch auf eine Sicht von drei bis sechs Monaten flexibel. Wenn Änderungen in einer laufenden Iteration auftreten, dann wird abgewogen, ob sich diese Änderungen innerhalb des festgelegten Budgets bewegt. Wenn nicht, dann wird diese Änderung durch ein Change Request behandelt.

*Interviewteilnehmer 13: "Der Prozess sorgt dafür, dass das, was die Teams die nächsten drei Monate machen sehr sehr klar ist. [...] Da gibt es Gremien, wo du deine Sachen vorstellen musst und die müssen genaue Kriterien erfüllen. [...] Dann gibt es Rückfragen und dann geht es über in ein sogenanntes Big Room Planning. Du planst relativ klar."*

Der letzte Modus ist der *Managementmodus*. Dieser befindet sich am nächsten an einem reinen plan-basierten Vorgehen. In diesen Fällen werden zwar agile Methoden während der Entwicklung angewendet, die Firmen legen jedoch die Anforderungen, Umfang und Zeit für ein gesamtes Projekt fest. Die Anwendung von agilen Methoden wird als Vorteil für das Managen eines Projektes und als Managementtool betrachtet. Mit Hilfe von agilen Methoden lassen sich die Arbeitsabläufe strukturieren und Review und Planning Meetings ermöglichen eine kontinuierliche Überwachung des Projektstatus. Zusätzlich helfen Dailys und Kanbanboards dabei, die Arbeitsauslastung der Mitarbeiter zu überwachen. Die reine Anwendung von agilen Methoden führt in diesen Fällen jedoch nicht zu Flexibilität. Die untersuchten Fälle in der Interviewstudie, die diesen Modus benutzen, operieren trotzdem in einem komplexen Umfeld. Das heißt, sie müssen auf Probleme während der Entwicklung reagieren können und bräuchten einen flexiblen Umfang.

An diesen Modi ist zu sehen, dass Firmen, die einen flexiblen Umfang für ihre Projekte haben auch in der Durchführung der Projekte flexibel sind. Es lässt sich daher die folgende Beobachtung ableiten:

**Beobachtung 3:** Die Flexibilität eines Projektes hängt davon ab, wie flexibel ein Projektteam einzelne Features austauschen und fallen lassen kann während eines Projektes. Wir haben vier verschiedene Modi identifiziert, die eine unterschiedliche Flexibilität im Umgang mit dem Umfang darstellen. Der Modus, mit der meisten Flexibilität, ist der Flexibilitätsmodus und, der mit der geringsten Flexibilität, ist der Managementmodus.

### 5.2.2 Balance der Anforderungsanalyse

Die Firmen wollen zu Beginn eines Projektes dem Projekt eine inhaltliche Struktur und Richtung geben, nach der das gesamte Projektteam ausgerichtet werden kann.

*Interviewteilnehmer 01: "Sie müssen erstmal so die groben Pfeiler einrammen und sagen, in diese Richtung soll es gehen, weil sonst bewegen Sie sich ja im chaotischen Bereich und da wollen wir ja nicht hin."*

Auf der anderen Seite wollen sie nicht zu Beginn zu viel Aufwand in die Anforderungsanalyse investieren, um sich Freiheitsgrade bei der Entwicklung zu lassen.

*Interviewteilnehmer 01: "Sie müssen einen Grad der Anforderungsanalyse haben, wo sie sich noch gewisse Freiheiten erlauben können. Sonst sind sie ja ganz schnell wieder im agilen Wasserfall."*

In der Spalte *Frameworks* von Tabelle 5.2 ist dargestellt, welche der drei vorgestellten Frameworks die Firmen zur Organisation ihres APH-Ansatzes nutzen. In 12 der 16 untersuchten Fälle nutzen die Projekte das Wasserfall-Agil Framework. Bei den drei Projekten, die nur das Pipeline Framework verwenden, handelt es sich um Bestandsprojekte, die bereits über Jahre hinweg laufen. Diese nutzen das Wasserfall-Agil Framework daher nicht. Von den 12 Fällen nutzen drei Projekte den Managementmodus bei der Verwaltung des Umfanges. Diese Projekte haben die meisten Anforderungen zu Beginn des Projektes gesammelt und stellen daher keine Balance beim Requirements Engineering her. In den restlichen Fällen haben wir eine Balance bei der Anforderungsanalyse beobachtet. Die folgenden beschriebenen Beobachtungen beziehen sich daher auf die verbliebenen neun Fälle. Hierbei haben wir festgestellt, dass in allen Fällen, die Balance ähnlich ist.

Für die Balance konzentrieren sich die Firmen in der Anforderungsanalyse zu Beginn des Projektes darauf, ein ausreichendes und gemeinsames Verständnis zwischen dem Projektteam und den Stakeholdern für das Projekt zu erreichen. Bei dem gemeinsamen Verständnis mit den Stakeholdern geht es darum, dass sich alle involvierten Personen darüber einig sind, was in dem Projekt umgesetzt werden soll. Für das gemeinsame Verständnis innerhalb des Projektteams wird sichergestellt, dass alle die gleiche Vision haben und Einigkeit darüber herrscht, wie das Projekt umgesetzt werden kann.

*Interviewteilnehmer 01: "Bei den Dingern haben wir den Anspruch die zu verstehen und zwar so tief, dass das Team am Ende sagen kann, ok, wir haben jetzt eine Vorstellung was da gebraucht wird. Und wir sind auch in der Lage so einen Daumenwert abzugeben, wie viel Entwicklungsaufwand darin steckt"*

Um das gemeinsame Verständnis zu erreichen, identifiziert das Projektteam die Stakeholder und deren Anwendungsfälle für die Software. Des Weiteren identifizieren sie die kritischen nicht-funktionalen Anforderungen, die gleich zu Beginn eines Projektes berücksichtigt werden müssen. Zu diesen Anforderungen gehören zum Beispiel Performance- und Securityanforderungen. Für das gemeinsame Verständnis betrachtet das Projektteam zudem Absprachen, die mit anderen Projekten oder innerhalb des Projektes zu Beginn des Projektes getroffen werden müssen. Dies betrifft zum Beispiel wichtige Integrationspunkte oder Deadlines. Damit erfährt das Projektteam, welchen Inhalt das Projekt hat, und kann sich so an einer gemeinsamen Richtung ausrichten. Um ein Verständnis über den Inhalt des Projektes zu erreichen, priorisiert das Projektteam zusammen mit den Kunden die Anforderungen. Dies erzeugt für die Stakeholder Transparenz darüber, welche Anforderungen umgesetzt werden und welche möglicherweise nicht.

Wenn das Projektteam bei der Anforderungsanalyse keine Informationen mehr erhält, die das Verständnis verbessern, dann brechen sie diese ab. Ein Indikator, der verwendet wird, um zu bestimmen, ob die Anforderungen ausreichend zu Beginn erhoben wurden, ist, wenn das Projektteam in der Lage ist eine grobe Schätzung des Projektes abzugeben. Dies ermöglicht ihnen mit ausreichenden Informationen in das Projekt zu starten, ohne dass sie zu viele Freiheitsgrade verlieren und sich im Detail verlieren.

Interviewer: "Wann wird up-front Anforderungsanalyse schädlich?"
*Interviewteilnehmer 13: "Du hast diesen Moment in dem gefühlt alle Fragen beantwortet sind. Da kommt dann auch so eine Aussage, wie groß der Aufwand ist. Da kommt dann auch raus, ok, wir haben verstanden was ihr so braucht. Wenn wir mal diese Epics hochrechnen, dann werden wir wahrscheinlich Kosten von einer Million.[...] Du hast ja am Anfang die Sammlung von Informationen. Wenn da nichts mehr dazu kommt, dann solltest du anfangen."*

Um die Anforderungen während der Anforderungsanalyse zu Beginn des Projektes festzuhalten, geben die Interviewteilnehmer an, dass Epics ein geeignetes Artefakt ist. Epics ermöglichen ihnen, die Anwendungsfälle festzuhalten, ohne sich dabei in Details zu verlieren. Unterstützt wird die Dokumentation des gemeinsamen Verständnisses durch Roadmaps. Die Roadmap bildet dabei die grobe Reihenfolge ab, in der die einzelnen Features entwickelt werden sollen,

und zeigt somit die Hauptpfeiler des Projektes. Die Interviewteilnehmer nutzen Roadmaps, um die Vision und Richtung des Projektes zu definieren, darzustellen und an die Projektteilnehmer zu transportieren.

**Beobachtung 4:** Projektteams bemühen sich am Anfang eines Projektes, ein gemeinsames Verständnis über den Inhalt des Projektes zu erreichen. Dieses gibt den Projekten eine inhaltliche Struktur und Richtung, anhand derer das gesamte Projekt ausgerichtet werden kann. Das gemeinsame Verständnis besteht hauptsächlich aus der Identifizierung der Anwendungsfälle der Stakeholder. Um zu Beginn nicht den Überblick zu verlieren, werden detaillierte Anforderungen offen gelassen.

Detaillierte Anforderungen werden während der Anforderungsanalyse zu Beginn des Projektes nicht betrachtet. Detaillierter Anforderungen betreffen zum Beispiel die konkreten Nutzer und Systemschritte in Anwendungsfällen. Diese müssen vom Projektteam während der Entwicklung gesammelt und analysiert werden. Zudem kann es vorkommen, dass Anforderungen dazukommen oder ausgetauscht werden. Dafür nutzen die Firmen während des Projektes ein kontinuierliches Requirements Engineering. Alle untersuchten Fälle nutzen für ein kontinuierliches Requirements Engineering das Pipeline Framework. Der Vorteil des Pipeline Frameworks ist, dass es einen kontinuierlichen parallelen Anforderungsprozess zur Entwicklung ermöglicht. Für ein kontinuierliches Requirements Engineering wurden die folgenden wichtigen Tätigkeiten identifiziert:

**1. Aufnahme von neuen oder geänderten Anforderungen:** Neue Anforderungen oder Änderungen von Kunden und Stakeholdern müssen aufgenommen und hinsichtlich ihres Geschäftswertes analysiert werden. Basierend darauf muss auch hier eine Priorität dieser Anforderungen festgelegt werden, um das Projekt im weiteren Verlauf zu steuern.

**2. Schätzung:** Diese Tätigkeit ist wichtig in APH-Ansätzen, damit die Projektteilnehmer analysieren können, wie die neuen oder geänderten Anforderungen die Projektparameter (initiale Schätzung) beeinflussen. Dafür werden die Anforderungen an die Entwickler gegeben. Diese analysieren die Anforderungen und bewerten ihre Umsetzbarkeit. Basierend auf der Analyse geben die Entwickler eine Schätzung der Anforderungen ab. Mit Hilfe der Schätzung und der Priorität der Anforderungen kann der Product Owner die weitere Entwicklung planen. Wichtig ist, dass die Schätzung schnell erfolgt, um unnötige Verzögerungen zu verhindern. Daher wird auch hier nur eine grobe Analyse durchgeführt, die ausreicht, dass die Entwickler ein Verständnis für die Anforderungen gewinnen.

**3. Backlogpflege:** Zu Beginn eines Projektes werden die Anforderungen auf der Ebene von Epics gesammelt. Während der Entwicklung müssen die detaillierten Anforderungen zu diesen Epics formuliert und mit User Stories festgehalten werden.

**4. Refinements:** Der Product Owner präsentiert die User Stories den Entwicklern. Wenn diese die User Stories als ausreichend für die Entwicklung betrachten, können die User Stories für die Entwicklung geplant werden.

**5. Planung der Entwicklung:** Wenn die Entwickler ihre Zustimmung zu den geschriebenen User Stories geben, können diese für die Entwicklung geplant werden.

### 5.2.3 Balance bei der Architektur

Mit der Architekturdefinition zu Beginn eines Projektes wollen die Projektteilnehmer eine technische Struktur erreichen. Diese ermöglicht ihnen einen Überblick über die Software und eine Orientierung für die weitere Entwicklung der Software.

> *Interviewteilnehmer 11: "Also am Anfang legst du so das Zielbild [der Architektur] fest. Das wird sich verändern, aber jeder weiß, da möchte ich eigentlich hin. Da auf diesen Leuchtturm, diesen Nordstern, da bewege ich mich hin."*

Auf der anderen Seite wollen sie jedoch auch nicht zu viele Freiheitsgrade einschränken und sich in detaillierten Entscheidungen verlieren.

> *Interviewteilnehmer 01: "Sie müssen natürlich aufpassen wie eng Sie den Architekturrahmen stecken. Wenn Sie das zu detailliert runterbrechen und sich keine Freiheitsgrade erlauben, dann haben die Entwickler keine Möglichkeiten gewisse Dinge vernünftig zu entwickeln."*

Im Gegensatz zur Anforderungsanalyse nutzen alle Projekte in der Interviewstudie, die das Wasserfall-Agil Framework nutzen, eine Balance bei der Definition der Architektur. Die folgenden beschriebenen Beobachtungen stammen daher aus diesen Fällen.

Bei der Definition der Architektur zu Beginn des Projektes fokussiert sich das Projektteam darauf, das grundlegende Schema für die Architektur festzulegen. Dafür wird zum Beispiel untersucht, welche Patterns für die Architektur eingesetzt werden und ob zum Beispiel die Architektur dem Schichtmodell folgt. Ebenfalls werden die Komponenten der Software definiert, die die einzelnen Funktionen, die aus den Anwendungsfällen abgeleitet sind, darstellen. Für die Kommunikation dieser Komponenten legen die Projekteams die Schnittstellen zwischen diesen Komponenten fest. Des Weiteren legen die Firmen am Anfang fest, welche Technologien für das Projekt verwendet werden sollen. Dies betrifft zum Beispiel die verwendete Programmiersprache oder Anwendungsframeworks, die im Projekt eingesetzt werden sollen. Zu der Analyse der Architektur gehört auch die Analyse der Infrastruktur der Software. Diese betrifft zum Beispiel das Endgerät auf dem die Software laufen soll oder existierende Datenbanken, die

einbezogen werden müssen. Zusätzlich liegt der Fokus bei der Architektur auf der Umsetzung von wichtigen nicht-funktionalen Anforderungen, wie Performance oder Security. Diese müssen zu Beginn eines Projektes betrachtet werden, da es schwierig ist, diese später zu integrieren.

Bei der Analyse der Antworten zeigt sich, dass die *Definition der Schnittstellen* eine entscheidende Rolle bei der Balance von der Architekturaktivität spielen. Die Definition der Schnittstellen ermöglicht dem Projektteam einen Überblick über die Software, da sie angeben wie einzelne Komponenten miteinander zusammenhängen. Die Schnittstellen unterstützen ebenfalls bei der Koordination von Projekten. Durch die Schnittstellen wissen die Entwicklerteams mit welchen anderen Teams sie sich abstimmen müssen. Die Definition der Schnittstellen gibt dem Projekt daher eine Struktur und Stabilität. Auf diese Weise erreichen die Firmen eins ihrer zentralen Ziele.

> *Interviewteilnehmer 11: "Wo läuft welches System und mit wem kommuniziert es. Das ist glaube ich so das Minimum, was man am Anfang festhalten muss, weil das auch zur Aussteuerung der einzelnen Parteien, die an diesen Elementen arbeiten, essentiell ist. Weil sie dann auch wissen, wer ist mein Counterpart, mit wem kann ich über XYZ sprechen. "*

Auf der anderen Seite sind die Entwicklerteams durch die Definition der Schnittstellen frei darin, wie sie die einzelnen Komponenten entwickeln. Das bedeutet, dass sie freie Entscheidungen treffen können, solange sie die definierten Schnittstellen nicht verändern. Damit erreichen die Firmen, dass sie zu Beginn eine technische Struktur aufbauen, jedoch während der Entwicklung genug Freiheitsgrade offen lassen.

> *Interviewteilnehmer 09: "Wenn die Schnittstellen klar sind, dann ist es eigentlich egal wie die Implementierung dahinter aussieht. "*

Da während des Architekturdesigns zu Beginn des Projektes der Fokus auf dem grundlegenden Aufbau der Architektur liegt, wird diese während der Entwicklung weiterentwickelt. Die Weiterentwicklung der Architektur findet vorrangig in den Entwicklerteams zusammen mit der Implementierung statt. Fünf Interviewteilnehmer berichten von einer zentralen Rolle oder Team, das bei der Entwicklung einen Überblick über die Architektur behält, die Entwicklung lenkt und Architekturentscheidungen auf Projektebene trifft.

**Beobachtung 5:** Zur Balance der Definition der Architektur konzentrieren sich die Projektteams darauf, die Komponenten der Software und deren Schnittstellen zu definieren. Auf der einen Seite ermöglicht die Definition der Schnittstellen einen Überblick über die Software und unterstützt bei der Koordination. Auf der anderen Seite gibt es den Entwicklerteams Freiheiten darüber, wie sie die Komponenten implementieren.

Die Architektur ist eng mit der Dokumentation des Projektes verbunden. Die Daten der Interviewstudie zeigen, dass die Firmen im Zweifelsfall eher zu viel als zu wenig dokumentieren. Jedoch muss Dokumentation auch geschrieben und gepflegt werden. Dies macht Dokumentation zu einer aufwendigen Tätigkeit. Die Resultate zeigen, dass Firmen nur das dokumentieren, was zum Verständnis der Software und der Architektur nötig ist. Es ist wichtig die allgemeine Struktur der Software zu dokumentieren, um einen Überblick zu haben.

*Interviewteilnehmer 01: "Alles was Struktur des Produktes oder der Anwendung angeht, muss ich dokumentieren. Fachliches Wissen kann ich nur begrenzt dokumentieren. "*

Wenn Dokumentation nur eine andere Darstellung von etwas ist, das bereits existiert, wird diese nicht erstellt. Des Weiteren werden Modelle, die einfach aus dem Code abgeleitet werden können, ebenfalls nicht dokumentiert. Kommentare im Code werden immer noch als einer der wichtigsten Formen der Dokumentation betrachtet. Um Dokumentation zu bewerten, überprüfen Senior Entwickler, ob eine erstellte Dokumentation für das Verständnis der Software ausreicht. Wissen, das aus Büchern gewonnen werden kann oder online zu finden ist, wird ebenfalls nicht dokumentiert. Darüber hinaus wird Dokumentation von den Firmen als zusätzliche Tätigkeit zu der Entwicklungsarbeit betrachtet. Diese wird ähnlich zur Entwicklungsarbeit geplant und von den Entwicklern und Architekten ausgeführt.

**Beobachtung 6:** Informationen, die benötigt werden, um die Software zu verstehen, werden dokumentiert. Fachwissen und Informationen, die leicht zu erlangen sind, werden nicht dokumentiert.

### 5.2.4 Balance beim Testen

Agile Ansätze empfehlen, das Testen von den Entwicklern durchführen zu lassen, um schnell Feedback zu erlangen [16]. Acht Interviewteilnehmer berichten jedoch, dass es in einigen Fällen nötig ist, ein extra Testteam mit einem von der Entwicklung getrennten Testprozess zu nutzen, um eine strukturierte Durchführung von Integration- und Systemtests sicherzustellen.

*Interviewteilnehmer 12: "Ein Vorteil ist, du weißt ja nicht, wenn du es ausrollst, ob es noch mit den anderen funktioniert. Wenn du da nicht ordentlich getestet hast oder mit den anderen kommunizierst, dann ist natürlich so ein Testteam Gold wert, das das Gesamte so ein bisschen kanalisiert. "*

In allen untersuchten Fällen werden die Unit Testfälle direkt von den Entwicklern durchgeführt. Für die Durchführung der Integration- und Systemtests werden von den verschiedenen Fällen zwei unterschiedliche Optionen genutzt. In einigen Fällen werden diese Tests von einem ex-

tra Testteam durchgeführt, in anderen Fällen sind direkt die Entwickler verantwortlich für diese Tests.

Der Vorteil der ersten Option ist, dass es eine klare Testverantwortlichkeit und ein Team gibt, das den Testprozess steuert. Des Weiteren können sich die Entwickler auf diese Weise auf die Entwicklung konzentrieren und müssen keine Zeit im Entwicklungssprint für die Integration- und Systemtests aufwenden.

> *Interviewteilnehmer 01: "Stellen Sie sich mal vor, Sie haben einen N-to-N Prozess der sieben oder acht Softwareprodukte beinhaltet. Wem wollen Sie da die Verantwortung (fürs Testen) geben? Das können sie gar nicht. Also brauchen Sie irgendwo ein Team, das sich dann darum kümmert, dass dieser Testprozess von vorne nach hinten durchläuft."*

Die Vorteile der zweiten Option sind, dass in diesen Fällen die Entwicklerteams neue Features direkt ausliefern können. Wenn ein extra Testteam für die System- und Integrationstests verwendet wird, müssen gefundene Fehler erst an die Entwicklerteams berichtet werden, bevor diese sie beheben können. Zudem unterbricht die Behebung dieser Fehler den laufenden Sprint der Entwickler. Daher verzögert die Verwendung eines extra Testteams die Entwicklung. Wenn die Tests von den Entwicklern durchgeführt werden, fördert dies die Existenz einer lauffähigen Version der Software nach jedem Sprint und das Qualitätsbewusstsein erhöht sich innerhalb der Entwicklungsteams.

> *Interviewteilnehmer 12: "Die Entwickler wollen bei Bedarf sofort deployen. Klarer Vorteil, je näher das Deployen an der Entwicklung liegt, desto schneller kriegen wir Feedback, desto schnell können wir es auch wieder fixen. "*

Die Analyse der Interviewdaten zeigt, dass in nur einem Fall die Testaktivität der Wasserfall-Agil Framework verwendet wurde. In diesem Projekt wurde eine Software-Hardware-Anwendung entwickelt und die Testaktivität am Ende des Projektes war notwendig, um das Gesamtsystem zu testen. In allen anderen Fällen wurden die Tests iterativ durchgeführt, um regelmäßig ausliefern zu können. In acht Fällen verwendeten die Firmen das Pipeline Framework mit einem extra Testteam um die System- und Integrationstests durchzuführen. Dafür definieren die Firmen einen synchronisierten Releaserythmus für die Entwicklerteams, der festlegt, wann und auf welchen Teststagen die Entwicklerteams ihre Features an das Testteam ausliefern. Dies ist wichtig für die Koordination des Testprozesses.

Im Bereich des Testens findet sich keine genaue Balance. Stattdessen müssen die Firmen entscheiden, welche Tests von einem extra Testteam durchgeführt werden müssen, um einen strukturierten Testprozess sicher zu stellen und trotzdem schnelles Feedback zu generieren.

Empfehlungen darüber, wann ein Projekt ein extra Testteam benötigt und wann die Integrations- und Systemtests von den Entwicklerteams durchgeführt werden können, wird in den folgenden Kapiteln diskutiert. Alle Interviewteilnehmer nennen jedoch Testautomatisierung als einen entscheidenden Faktor für eine schnelle Entwicklung, mehr Flexibilität und schnelles Feedback.

**Beobachtung 7:** Für die Balance im Bereich des Testens müssen die Firmen entscheiden, welche Tests von den Entwicklern durchgeführt werden können und welche durch ein extra Team (wenn nötig) durchgeführt werden müssen.

### 5.2.5 Balance in der Koordination

Selbstorganisation in Entwicklerteams steigert deren Performance [92]. Die Fähigkeit eines Projektteams zur Selbstorganisation wird jedoch durch ihre Größe bestimmt [92]. Für die Balance in der Koordination setzen die Firmen Einschränkungen bei der Selbstorganisation ein. Diese Einschränkungen ermöglichen es auf der einen Seite, dass die Firmen Kontrolle über Entscheidungen und Absprachen behalten. Auf der anderen Seite bleibt genug Raum für die Entwickler sich auszutauschen und Entscheidungen selbstständig zu treffen.

Die erste Einschränkung ist, dass sich in großen Projektteams die einzelnen Entwicklungsteams an das generelle Entwicklungsframework und den Rhythmus des Projektes halten müssen. Dadurch sind sie in ihrer Selbstorganisation eingeschränkt, jedoch gibt es eine klare Koordination des Projektes. Innerhalb ihres Teams sind die Entwickler somit frei, welche Praktiken sie anwenden wollen.

Die zweite Einschränkung existiert bei fundamentalen Entscheidungen, die nicht nur ein einzelnes Entwicklungsteam betreffen, sondern mehrere Teams oder sogar das gesamte Projekt. In diesen Fällen wird die Projektleitung in die Entscheidung mit eingebunden. Dies bedeutet, dass die Entwicklerteams innerhalb ihres eigenen Bereiches selbstständig Entscheidungen treffen können.

*Interviewteilnehmer 12: ”Die Selbstorganisation endet meist da, wo wir eine fundamentale Frage haben, die nicht nur unser Team betrifft, sondern auch andere Teams. [...] Fundamentale Fragen gehen meistens nach oben. Diese würden wir ungern selber organisieren. Zum Beispiel die Frage dürfen wir diese oder jene Anwendung lizenztechnisch benutzen. ”*

Die dritte Einschränkung existiert, wenn die Projektparameter (Budget oder Zeit) durch eine Entscheidung beeinflusst werden. Auch in diesen Fällen wird die Projektleitung in den Entscheidungsprozess mit eingebunden.

*Interviewteilnehmer 02: ”Wenn sie natürlich sagen, ja wir brauchen da jetzt soundso viel mehr Geld, das können sie nicht einfach selber entscheiden. Dann müssen sie erstmal kommen und müssen das sagen. ”*

Die letzte Einschränkung existiert, wenn ein Entwicklerteams fundamental von seiner Aufgabe abweichen muss und damit die Aufgabe hinfällig wird. Auch in diesen Fällen wird die Projektleitung oder der Product Owner einbezogen. Ansonsten sind die Teams angehalten, sich innerhalb ihrer Aufgabenstellung selbst zu organisieren. Das heißt, sie können Entscheidungen und Absprachen, die zur Erreichung der Aufgabenstellung nötig sind selbstständig treffen.

*Interviewteilnehmer 02: ”Alles was sozusagen innerhalb des Scopes der Aufgabenstellung erforderlich ist, damit dieses Ziel erreicht werden kann, können sie im Grunde selber entscheiden. ”*

**Beobachtung 8:** Selbstorganisation endet dort, wo Entscheidungen das gesamte Projekt beeinflussen, die Parameter des Projektes verändert werden müssen oder wenn das Entwicklungsteam grundsätzlich von ihrer zugewiesenen Aufgabe abweichen muss. In diesen Fällen werden zentrale Entscheidungen getroffen.

## 5.3 Zusammenfassung

Das vorherige Kapitel beschreibt die Schwierigkeiten von Firmen bei der Nutzung von APH-Ansätzen. Dabei zeigt sich, dass die Firmen besonders Schwierigkeiten bei der Balance von agiler und plan-basierter Entwicklung haben. Um zu untersuchen, wie die Firmen eine Balance in APH-Ansätzen erreichen, haben wir eine Interviewstudie mit Praktikern von APH-Ansätzen durchgeführt.

Die Daten der Interviewstudie zeigen, dass die Firmen eine Struktur und Stabilität für ihre Projekte erreichen wollen und gleichzeitig ausreichend flexibel bleiben möchten. Wir haben im Rahmen der Interviewstudie untersucht, wie Firmen diese Balance für die einzelnen Aktivitäten von Anforderungsanalyse, Architektur, Planung, Testen und Koordination erreichen. Die Interviewstudie beweist, dass Firmen in der Lage sind, eine Balance zwischen agiler und plan-basierter Entwicklung zu erreichen und dabei die jeweiligen Vorteile zu nutzen. Bei der Dokumentation hat es sich gezeigt, dass diese weniger abgewägt werden muss als die anderen Aktivitäten. Stattdessen wird jeweils von Senior Entwicklern bewertet, was für eine Feature dokumentiert werden muss, damit es von anderen Entwicklern verstanden werden kann. Dokumentationsarbeiten werden ähnlich zu der Entwicklung geplant, in dem diese zum Beispiel mit auf Story Cards vermerkt werden. Aus diesem Grund wird die Dokumentation in den folgenden Kapiteln nicht weiter behandelt.

Bei den Ergebnisse der Interviestudie zeigen sich Übereinstimmungen mit den Ergebnissen

aus der Literaturstudie (Kapitel 4). Es zeigt sich, dass die Tätigkeiten, die zu Beginn von Projekten durchgeführt werden in beiden Studien ähnlich sind. In beiden Studien wurde zum Beispiel identifiziert, dass die Firmen zu Beginn eine grobe Schätzung des Projektes durchführen. Daher nehme ich an, dass die Firmen in den Fällen der Literaturstudie (Kapitel 4) eine ähnliche Balance erreichen.

Obwohl die Firmen bei der Balance die zwei gleichen zentralen Ziele haben, unterscheiden sich die Ausprägungen bei der Balance zwischen den Firmen. So kann in einem Projekt, das mit sehr unsicheren Anforderungen umgehen muss, nicht unbedingt ein ausreichendes Verständnis zu Beginn des Projektes erreicht werden. Dies bedeutet, dass die Firmen ihren APH-Ansatz individuell balancieren müssen. Um dabei die Firmen zu unterstützen, wird in den nächsten Kapiteln beschrieben, welche generellen Risiken bei der Balance abgewägt werden müssen und wie der Kontext von Firmen die Balance beeinflusst.

# 6

# Ein Konzept für die Balance von APH-Ansätzen

Die Erschaffer der ersten agilen Methoden haben erkannt, dass es wichtig ist, dass Firmen verstehen, warum bestimmte Praktiken in einem Entwicklungsprozess angewendet werden sollen. Daher haben sie Prinzipien entwickelt, die die Philosophie von agiler Entwicklung beschreiben [26]. Diese Prinzipien bieten Richtlinien für die Anpassung von agilen Methoden an einen individuellen Projektkontext [16]. Zum Beispiel gibt das agile Manifest eine Richtlinie für die Länge eines Auslieferungszyklus an. Bei der Wahl zwischen verschiedenen Längen sollten sich Firmen für die kürzere Länge entscheiden [16].

Das Ziel dieser Dissertation ist, Firmen bei der Entwicklung von APH-Ansätzen zu unterstützen. Die vorherigen Kapitel zeigen, dass dabei besonders die Balance von agiler und plan-basierter Entwicklung eine wichtige Rolle spielt. Ähnlich zu agiler Entwicklung haben die vorangegangenen Kapitel gezeigt, dass diese Balance von den Gegebenheiten in einem Projekt abhängig ist. Firmen müssen demnach selbstständig eine ideale Balance für ihren jeweiligen Fall herstellen. Daher ist es wichtig, dass in einem Framework für APH-Ansätze Richtlinien und somit Prinzipien geschaffen werden, die Firmen bei der Balance ihres Entwicklungsansatzes anleitet.

In diesem Kapitel wird daher ein allgemeines Konzept vorgestellt, um die Kräfte bei der Balance von APH-Ansätzen zu beschreiben. Dieses Konzept basiert auf der Verwendung der Risk Exposure für die Wahl zwischen zwei Alternativen. Es dient dazu im folgenden Kapitel Prinzipien für die Balance von agiler und plan-basierter Entwicklung abzuleiten.

## 6.1 Risikobasierte Betrachtung der Balance von APH-Ansätzen

Besonders die Ergebnisse der Literaturstudie (Kapitel 4) haben gezeigt, dass die Balance von agiler und plan-basierter Entwicklung komplex ist. Die Komplexität entsteht dadurch, dass Firmen zwischen vielen verschiedenen Optionen abwägen müssen. Während der Anforderungsanalyse zu Beginn eines Projektes müssen sie zum Beispiel abwägen, ob sie diese fortsetzen oder abbrechen. Die Komplexität ergibt sich weiter dadurch, dass die Firmen bei diesem Abwägen viele Vor- und Nachteile berücksichtigen müssen.

Zum Beispiel ermöglicht eine Anforderungsanalyse zu Beginn des Projektes die Schaffung eines Überblicks über das Projekt. Es muss jedoch darauf geachtet werden, dass bei einer hohen Unsicherheit der Anforderungen genug Freiheitsgrade in der Anforderungsanalyse verbleiben. Zusätzlich zu den allgemeinen Vor- und Nachteilen ist das Abwägen in APH-Ansätzen kontextabhängig. Zum Beispiel hat die Schaffung eines Überblicks über das Projekt in Projekten mit einem großen Projektteam einen großen Vorteil. Dies liegt daran, dass ein Überblick über das Projekt die Koordination erleichtert. Hingegen ist dieser Vorteil in einem kleinen Projektteam durch den geringeren Koordinationsaufwand geringer.

Jeder Kontext verlangt nach unterschiedlichem Vorgehen. Durch die vielen unterschiedlichen Kontexte ist es daher unmöglich, eine Balance für agile und plan-basierte Entwicklung zu finden, die für alle Firmen gleichermaßen funktioniert [10]. Daher ist es wichtig, ein Konzept zu schaffen, das zeigt, welche Kräfte auf welche Weise bei der Balance interagieren und wie diese Kräfte im Zusammenhang mit dem Kontext abgewogen werden können. Dies unterstützt Firmen dabei, eigenständig die beste Balance zu finden.

Die Daten der Literatur- und Interviewstudie (Kapitel 4 und 5) beschreiben Probleme, die auftreten können, wenn zum Beispiel zu wenig plan-basierte Entwicklung oder agile Entwicklung eingesetzt werden. Ein Problem von zu wenig Anforderungsanalyse zu Beginn des Projektes ist, dass wichtige Anforderungen übersehen werden können. Dadurch kann es zu späteren Überarbeitungen der Software kommen und damit zu einer Verzögerung des Projektes. Es gilt somit, mögliche Probleme und deren potenzielle Schäden bei der Balance von agiler und plan-basierter Entwicklung abzuwägen.

Der Zusammenhang von Problem und Schaden entspricht dem Konzept des Risikos. Der

IEEE Standard 1540:2001 [105] definiert ein Risiko als ein potenzielles Problem. Ein Risiko besteht aus der Wahrscheinlichkeit, mit der ein potenzielles Problem eintreten kann und der negativen Auswirkungen, die das Problem verursacht. Die negativen Auswirkungen können einen Schaden an einem Projekt verursachen. Nach Boehm [106] betrifft dieser Schaden die unterschiedlichen Bereiche eines Projektes. Zum Beispiel kann sich ein Projekt durch die negativen Auswirkungen verlängern oder kostet mehr Geld.

Boehm und Turner [23] nutzen die Abwägung des Risikos, um Firmen bei der Entscheidung zwischen einem agilen, plan-basierten oder hybriden Vorgehen zu unterstützen. In diesem Vorgehen werden die Risiken, die jeweils ein agiles und plan-basiertes Vorgehen darstellen gegeneinander abgewogen. Wenn das Risiko durch ein agiles Vorgehen überwiegt, sollte eher ein plan-basiertes Vorgehen gewählt werden. Wenn im Gegensatz dazu die Risiken von einem plan-basierten Vorgehen überwiegen, sollte eher ein agiles Vorgehen gewählt werden. Boehm und Turner [23] sind so in der Lage, die verschiedenen Vor- und Nachteile und Kontexteinflüsse bezüglich agiler und plan-basierter Entwicklung abzuwägen. Des Weiteren hat Boruszewski [107] ein risiko-basiertes Vorgehen für die Optimierung von hybriden Anforderungslandschaften genutzt. Dies zeigt, dass die Verwendung des Risikos bereits bei der Entwicklung von APH-Ansätzen eingesetzt wurde. Zusätzlich ist das Konzept des Risikos durch das klassische Risikomanagement in Softwareprojekten weit verbreitet in der Softwareentwicklung und somit ein bekanntes Konzept [23]. Aus diesen Gründen habe ich mich für die Konzeptionierung der Balance von agiler und plan-basierter Entwicklung ebenfalls für die Verwendung des Risikos entschieden, um unterschiedliche Kräfte gegeneinander abzuwägen.

## 6.2 Grundlagen der Risikobetrachtung bei Alternativen

Bei der Balance von APH-Ansätzen geht es grundsätzlich darum, ein agiles gegen ein plan-basiertes Vorgehen abzuwägen. Um zwei Alternativen miteinander vergleichen zu können, müssen die Risiken der beiden Alternativen gegeneinander abgewogen werden können. Dafür ist es notwendig, dass Risiken Werten zugewiesen werden können, die es ermöglichen, unterschiedliche Risiken auf einer Skala miteinander zu vergleichen. Auf diese Weise kann ein Projektteam entscheiden, ob eine Alternative gegenüber einer anderen Alternative ein größeres oder geringeres Risiko darstellt.

Wie hoch ein Risiko ist, hängt von der Eintrittswahrscheinlichkeit eines potenziellen Problems ab. Ein Problem, das wahrscheinlich niemals eintritt, stellt kein großes Risiko dar. Dagegen ist ein Problem, das ganz sicher eintritt, sehr gefährlich. Des Weiteren hängt die Kritikalität eines Risikos vom potenziellen Schaden ab, der beim Eintreten eines Problems entstehen

kann. Wenn beim Eintreten eines Risikos zum Beispiel eine Verzögerung von einem Tag entsteht, ist dies weniger kritisch, als wenn ein Projekt sich um mehrere Wochen verlängert.

Der IEEE Standard 1540:2001 [105] verwendet daher den Begriff der Risk Exposure, um verschiedene Risiken zu bewerten. Die Risk Exposure gibt den möglichen Verlust für ein Projekt oder eine Firma an, der durch ein Risiko verursacht werden kann. Die Risk Exposure wird durch die Multiplikation der Eintrittswahrscheinlichkeit eines Problems mit dem Schaden, der durch die negativen Auswirkungen entsteht, gebildet. Der IEEE Standard [105] gibt an, dass die Wahrscheinlichkeit und der Schaden jeweils qualitativ und quantitativ angegeben werden kann. Eine quantitative Angabe der Wahrscheinlichkeit kann durch einen Wert zwischen 0 und 1 erfolgen. Der Schaden kann zum Beispiel durch die Zahl der Tage angegeben werden, um die sich ein Projekt beim Eintreten einer Gefahr verlängert. Beide Größen können auch qualitativ angegeben werden. Die Bewertung der Wahrscheinlichkeit und des Schadens können zum Beispiel durch eine Angabe von tief, mittel oder hoch ermöglicht werden. Die Risk Exposure kann demnach nach der folgenden Formel für ein Risiko r berechnet werden [107]:

$$RE(r) = P(r) * S(r)$$
$$P(r) \in [0,1] : Eintrittswahrscheinlichkeit\ des\ Risikos\ r$$
$$S(r) \in [0,30] : Potenzieller\ Schaden\ des\ Risikos\ r\ in\ Tagen$$

Ein Beispiel für das Abwägen von zwei Alternativen ist in Tabelle 6.1 dargestellt. In dem Beispiel gibt es die Alternativen 1 und 2, die gegeneinander abgewogen werden müssen. Die Wahl von Alternative 1 ist mit dem Risiko R1 verbunden. Dieses hat eine Eintrittswahrscheinlichkeit von 0,5 und einen potenziellen Schaden von 20 Tagen. Die Risk Exposure für dieses Risiko ist demnach 10. Dagegen steht die Alternative 2 mit dem Risiko von R2. Dieses hat eine Eintrittswahrscheinlichkeit von 0,3 und einen potenziellen Schaden von 60 Tagen. Die Risk Exposure für dieses Risiko ist demnach 18. Beide Alternativen können nun anhand der Risk Exposure verglichen werden. Aus diesem Vergleich ergibt sich, dass in diesem Beispiel die Alternative 1 die bessere Wahl ist, da der potenzielle Verlust in diesem Fall geringer ist.

**Tabelle 6.1:** Beispiel für das Abwägen von zwei Alternativen

| Alternative | Risiko | Eintrittwahrscheinlichkeit | Schaden | Risk Exposure |
|---|---|---|---|---|
| Alternative 1 | R1 | 0,5 | 20 Tage | $0,5 \cdot 20 = 10$ |
| Alternative 2 | R2 | 0,3 | 60 Tage | $0,3 \cdot 60 = 18$ |

## 6.3 Betrachtung der Risk Exposure als Funktion

Im vorherigen Abschnitt wurde beschrieben, wie zwei Alternativen basierend auf der Risk Exposure bewertet werden können. In dem Vorgehen von Boehm und Turner [23] werden die Risiken von agiler und plan-basierter Entwicklung einmal zu Beginn des Projektes abgewogen. Basierend darauf wird zu Beginn des Projektes eine Entscheidung über das Vorgehen getroffen. Ähnlich ist es beim Vorgehen von Boruszewski [107]. Hierbei wird zum Beispiel einmal die Risk Exposure für die Aufnahme eines Anforderungsartefaktes bestimmt. Bei der Balance von agiler und plan-basierter Entwicklung ist es jedoch nicht von Vorteil, bereits zu Beginn die Balance festzulegen. Dies hat drei Gründe.

Der erste Grund ist, dass es schwierig ist, Alternativen festzulegen. Dies zeigt sich am Beispiel von Requirements Engineering. Wenn bei einem Projekt die drei Alternativen von viel, mittel und wenig Anforderungsanalyse zu Beginn festlegt werden, werden möglicherweise existierende Mittelstufen nicht betrachtet. Bei der Balance von agiler und plan-basierter Entwicklung geht es darum, einen Mittelweg zwischen beiden Vorgehen zu finden. Durch Nutzung von nur drei Alternativen sind die Möglichkeiten für einen Mittelweg stark eingeschränkt. Durch die Wahl von Alternativen zu Beginn des Projektes schränkt sich ein Projektteam daher stark in den eigenen Handlungsmöglichkeiten ein.

Der zweite Grund ist, dass durch weitere Alternativen der Aufwand für die Bestimmung der Risk Exposure zu Beginn des Projektes für alle Alternativen steigt. Wie bereits beschrieben, müssen in diesem Prozess viele Vor- und Nachteile abgewogen werden. Dies bedeutet generell Aufwand für ein Projektteam. Mit einer steigenden Anzahl an Alternativen steigt dieser Aufwand zusätzlich an.

Der dritte Grund ist, dass ein Projektteam zu Beginn eines Projektes möglicherweise nicht genug Informationen hat, um die Risiken der Alternativen adäquat zu bewerten. Zum Beispiel existiert bei einer kurzen Anforderungsanalyse die Gefahr, dass wichtige Anforderungen übersehen werden. Diese Gefahr sinkt jedoch mit der Anzahl der erhobenen Anforderungen. Dies bedeutet, dass im Verlauf der Anforderungsanalyse das Projektteam mehr Informationen über die Risiken erhält. Diese Informationen sind wichtig für die Einschätzung des Risikos und somit auch für die Balance. Es ist daher für die Firmen von Vorteil, kontinuierlich die Risk Exposure zu der jeweiligen Menge von aufgenommenen Anforderungen zu bestimmen. Basierend darauf können sie entscheiden, ob sie die Anforderungsanalyse fortsetzen wollen oder nicht. Obwohl hier nur das Beispiel von Requirements Engineering betrachtet wird, existiert der gleiche Zusammenhang auch in den anderen Aktivitäten, wie Planung oder Architektur.

Dies spricht dafür, die Risk Exposure nicht für einzelne Alternativen zu bestimmen, sondern stetig in Abhängigkeit einer wachsenden Größe. In dem obigen Beispiel sinkt die Risk Expos-

ure durch mehr aufgenommene Anforderungen. Zur Veranschaulichung des Konzepts und die Abwägung von Risiken bei der Balance von agiler und plan-basierter Entwicklung wird daher die Risk Exposure jeweils als eine Funktion dargestellt, die von einer stetig wachsenden Größe abhängig ist.

## 6.4 Erreichung von Stabilität und Flexibilität in APH-Ansätzen

Boehm und Turner [5] schreiben, dass die Menge an Zeit und Aufwand, die in die Planung investiert wird, balanciert werden muss. Dabei steht das Risiko von inadäquaten Plänen auf der einen Seite und das Risiko vom Verlust von Marktanteilen auf der anderen Seite. Inadäquate Pläne sorgen für Unsicherheiten und Widersprüche und können zu Verzögerungen im Projekt führen. Der Verlust von Marktanteilen entsteht dadurch, dass das Projektteam zu viel Zeit für Planung verwendet und sich die Auslieferung von Software verzögert. Sie stellen die Risk Exposure dieser beiden Risiken als Funktionen dar. Beide Funktionen sind von der Zeit und dem Aufwand abhängig, die ein Projektteam in die Planung investiert. Die Summe aus beiden Funktionen stellt das Gesamtrisiko dar. Aus dem Tiefpunkt der Funktion der Risk Exposure für das Gesamtrisiko kann abgelesen werden, wie viel Zeit und Aufwand ein Projektteam in die Planung investieren sollte. Das heißt, dass es das Ziel von Boehm und Turner [5] ist, die investierte Zeit und den Aufwand für Planung zu optimieren.

Die Erkenntnisse aus der Interviewstudie (Kapitel 5) zeigen jedoch, dass es nicht das vorrangige Ziel ist, die ideale Menge an Zeit und Aufwand für eine Aktivität aufzuwenden. Stattdessen wollen die Firmen zum einen genug Stabilität für das Projekt, durch inhaltliche und technische Strukturen schaffen, zum anderen wollen sie nicht zu viele Strukturen schaffen, um genug Freiräume für Flexibilität bei Änderungen. Eine detailliertere Langzeitplanung sorgt dafür, dass Abhängigkeiten in einem Projekt berücksichtigt werden. Langzeitplanung sorgt so für Strukturen und verhindert Koordinationsprobleme. Langzeitplanung senkt so das Risiko, dass es zu wenig Stabilität gibt. Gleichzeitig wollen Projektteams auch nicht zu detaillierte Langzeitplanung betreiben. Detaillierte Pläne bedeuten einen hohen Aufwand in Fällen von Änderungen, da Meilensteine möglicherweise in Abhängigkeit geändert werden müssen. Eine zu detaillierte Planung macht ein Projektteam unflexibel. Sie haben daher gleichzeitig das Risiko von zu wenig Flexibilität. Zur Vereinfachung wird das Risiko von zu wenig Flexibilität im Folgenden als Risiko von zu viel Stabilität bezeichnet. Diese beiden Risiken von zu wenig und zu viel Stabilität stehen sich bei der Balance von agiler und plan-basierter Entwicklung gegenüber.

Die Literatur- und Interviewstudie (Kapitel 4 und 5) hat gezeigt, dass es auch in anderen Aktivitäten, wie Requirements Engineering und Architektur, es die Gegenüberstellung dieser

beiden Risiken von zu viel und zu wenig Stabilität gibt. Firmen wollen durch das Erheben von Anforderungen zu Beginn eines Projektes eine inhaltliche Struktur schaffen. Diese Struktur ermöglicht die gemeinsame Ausrichtung des Projektes und sorgt so für Stabilität für die Entwicklung. Sie wollen jedoch die Anforderungen auch nicht zu detailliert festlegen, da sonst zu viele Freiräume bei der Gestaltung der Anforderungen verloren gehen. Der Verlust von Freiheitsgraden kann wiederum zu teuren Änderungen im späteren Verlauf des Projektes führen. Es gibt also auch beim Requirements Engineering die Gegenüberstellung der Risiken von zu wenig und zu viel Stabilität.

Je mehr Anforderungen zu Beginn erhoben und verstanden werden, desto höher wird die Struktur und somit auch die Stabilität. Die Menge an aufgenommenen Anforderungen hat somit einen Einfluss auf die Risk Exposure von zu wenig Stabilität und auch auf die Risk Exposure von zu viel Stabilität. Bei der Balance von agiler und plan-basierter Entwicklung haben daher Projektteams nicht vorrangig das Ziel, die ideale investierte Zeit zu ermitteln. Stattdessen wollen sie einen idealen Stand an Struktur schaffen, der ihnen genug Stabilität und gleichzeitig genug Freiheiten bietet.

Zwischen der investierten Zeit für eine Aktivität und der erreichten Struktur gibt es keinen direkten Zusammenhang. Zum Beispiel kann sich trotz mehr investierter Zeit die Menge der erhobenen und verstandenen Anforderungen nicht ändern, da es zwischen diesen beiden Größen keinen direkten Zusammenhang gibt. Dies liegt daran, dass das Erheben und Verstehen von unterschiedlichen Anforderungen unterschiedlich lange dauern kann. Mit der mehr investierten Zeit würde sich in diesem Fall auch nicht der Stand der erreichten Struktur ändern.

Es wird daher in dieser Dissertation der allgemeine Begriff des *Strukturgrades* eingeführt. Dieser stellt den erreichten Stand an Struktur in einer Aktivität dar. Durch den Strukturgrad sind Projektteams in der Lage, Aussagen über die Risk Exposure von den Risiken von zu viel und zu wenig Stabilität zu treffen. Darüber können sie entscheiden, ob sie mehr oder weniger Struktur in einem Projekt benötigen.

Die Interviewstudie (Kapitel 5) hat gezeigt, dass es für alle in Kapitel 4 identifizierten Aktivitäten von Requirements Engineering, Architektur, Planung, Testen und Koordination die beiden Risiken von zu wenig und zu viel Stabilität gibt. Dies bedeutet, dass eine alleinige Sicht auf die Planung, wie im Ansatz von Boehm und Turners [5], nicht für eine vollständige Balance von agiler und plan-basierter Entwicklung ausreicht. Zudem reicht, wie oben diskutiert, eine einfache Betrachtung der Zeit nicht aus. Diese Lücken im Ansatz von Boehm und Turner [5] sollen in dieser Dissertation geschlossen werden. Der Risk Exposure Ansatz von Boehm und Turner [5] wird in dieser Dissertation genutzt, um ein allgemeines Konzept für die Balance von agiler und plan-basierter Entwicklung basierend auf dem Strukturgrad zu entwickelt. Dieses Konzept

wird in den folgenden Kapiteln der Dissertation genutzt, um die Balance in den einzelnen Aktivitäten zu analysieren und zu bestimmen. Aus der obigen Diskussion werden die folgenden Zusammenhänge für ein allgemeines Konzept für die Balance von agiler und plan-basierter Entwicklung abgeleitet:

1. Bei der Bestimmung der Balance von agiler und plan-basierte Entwicklung stehen sich in jeder Aktivität die beiden Risiken von zu wenig und zu viel Stabilität gegenüber. Diese müssen gegeneinander abgewogen werden.

2. Die Schaffung von mehr Struktur in einer Aktivität, wie Requirements Engineering oder Planung, lässt das Risiko von zu wenig Stabilität sinken und vergrößert gleichzeitig das Risiko von zu viel Stabilität. Beide Risiken stehen sich daher gegenüber und verhalten sich gegenläufig.

3. Die Risk Exposure für das Abwägen der beiden Risiken kann vom Projektteam in Abhängigkeit der erreichten Struktur ermittelt werden. Dafür wird an dieser Stelle der Begriff Strukturgrad eingeführt.

In diesem Kapitel werden die beiden Risiken von zu viel und zu wenig Stabilität allgemein beschrieben, um das allgemeine Konzept für die Balance vorzustellen. Eine detaillierte Analyse der Risiken bei der Balance der einzelnen Aktivitäten erfolgt im nächsten Kapitel.

## 6.5 Definition des Strukturgrades

In diesem Kapitel werden die beiden Risiken von zu wenig und zu viel Stabilität allgemein beschrieben. Daher wird zur Veranschaulichung in diesem Kapitel auch der Strukturgrad zunächst allgemein definiert. Eine detaillierte Beschreibung des Strukturgrades erfolgt im nächsten Kapitel. Dort werden die Balancen in den einzelnen Aktivitäten behandelt und der jeweilige entscheidende Strukturgrad vorgestellt.

Für die Definition eines allgemeinen Strukturgrades wird das von Boehm et al. [108] definierte Planungsspektrum verwendet. Auf der linken Seite dieses Spektrums befinden sich die Hacker, die keine Form von Planung und strukturiertem Vorgehen nutzen. Auf der anderen Seite des Spektrums befindet sich die Nutzung von Mikromeilensteinen und eisernen Verträgen. Diese Vorgehen gehen mit einer genau vorgegebenen Langzeitplanung einher und somit einer sehr strengen Struktur. Damit befindet sich auf der linken Seite des Spektrums überhaupt keine Struktur, jedoch sehr viel Flexibilität. Auf der anderen Seite des Spektrums gibt es sehr viel Struktur, jedoch keine Flexibilität. Zwischen diesen beiden Extrempunkten befinden sich von

links nach rechts die agilen Methoden und dann die mehr Meilenstein getriebenen Vorgehensweisen. Das Spektrum ist in Abbildung 6.1 auf der x-Achse abgebildet.

Die Risk Exposure bei der Balance der Risiken von zu wenig und zu viel Stabilität, wird in Abhängigkeit des Strukturgrades ermittelt. Dies bedeutet, dass die Wahrscheinlichkeiten und potenziellen Schäden der beiden Risiken jeweils von dem Strukturgrad eines Projektes abhängen. Damit die Funktion der Risk Exposure eine stetige Funktion ist, muss die Funktion anhand einer stetigen Variablen berechnet werden. Daher wird der allgemeine Strukturgrad s in dieser Dissertation auf einer Skala von 0 bis 100 definiert. Dabei beschreibt null die linke Seite des beschriebenen Spektrums (Hacker) und damit das Extremum von keiner Struktur im Entwicklungsprozess. Die hundert beschreibt die rechte Seite des Spektrums (eiserne Verträge) und damit das Extremum von maximaler Struktur und keiner Flexibilität im Entwicklungsprozess. Der Definitionsbereich der Funktion für die Risk Exposure, ist daher auf diesen Wertebereich beschränkt. Für die Skala des Strukturgrades werden die in Tabelle 6.2 aufgelisteten Intervalle festgelegt.

**Tabelle 6.2:** Intervalle auf der Skala des Strukturgrades

| **Wertebereiche für den Strukturgrad s** | **qualitative Beschreibung** |
|---|---|
| $0 \leq s \leq 20$ | sehr geringer Strukturgrad |
| $20 < s \leq 40$ | geringer Strukturgrad |
| $40 < s \leq 60$ | mittlerer Strukturgrad |
| $60 < s \leq 80$ | hoher Strukturgrad |
| $80 < s \leq 100$ | sehr hoher Strukturgrad |

## 6.6 Die Risk Exposure von zu wenig Stabilität

Das Risiko von zu wenig Stabilität während der Entwicklung ist eins der beiden Risiken, die bei der Balance von APH-Ansätzen berücksichtigt werden müssen. Dieses allgemeine Risiko entsteht dadurch, dass sich ein Projekt zu weit links auf dem Spektrum des Strukturgrades befindet. Dies bedeutet, dass sich ein Projekt zum Beispiel im Bereich des Hackings bewegt, wo es keine Regeln für das Vorgehen gibt. Hier werden zum Beispiel keine Anforderungen zu Beginn des Projektes gesammelt und keine Abhängigkeiten geplant. Die Gefahr ist dabei, dass ein Projekt in einen chaotischen Zustand gerät und Koordinations- und Kommunikationsprobleme auftreten.

Der Schaden, der dadurch entsteht, ist, dass möglicherweise wichtige Informationen verlorengehen und dass Fehler bei der Entwicklung passieren. Diese Fehler müssen später mit viel Aufwand wieder behoben werden. Dies kann die Kosten für ein Projekt in die Höhe treiben.

Für die Funktion der Risk Exposure von zu wenig Stabilität wird die Funktion $WS(s)$ definiert. Die Abkürzung WS steht hierbei für zu wenig Stabilität.

$$WS(s) = WWS(s) * SWS(s) \quad s \in [0, 100]$$

Die Funktion $WS(s)$ ist abhängig vom Strukturgrad s. Zur Bestimmung der Risk Exposure eines Risikos wird die Eintrittswahrscheinlichkeit mit dem potenziellen Schaden multipliziert. Zur Übertragung der Berechnung auf die Funktion werden daher ebenfalls die beiden Funktionen $WWS(s)$ und $SWS(s)$ definiert.

Die Funktion $WWS(s)$ gibt die Wahrscheinlichkeit für das Eintreten eines Problems in Abhängigkeit des Strukturgrades an. Wenn in einem Projekt mehr Anforderungen zu Beginn gesammelt werden, ist die Wahrscheinlichkeit geringer, dass wichtige Anforderungen am Anfang des Projektes übersehen werden. Im Gegensatz dazu ist die Wahrscheinlichkeit sehr hoch, wenn wenig bis gar keine Anforderungsanalyse betrieben wird.

Die zweite Funktion $SWS(s)$ bestimmt den potenziellen Schaden, der in Abhängigkeit eines Strukturgrades, beim Eintritt einer Gefahr entsteht. Zum Beispiel ist der Schaden, der durch das Übersehen von grundlegenden Anforderungen entsteht, größer als das Übersehen einer Farbe für einen Button. Abgeleitet von der allgemeinen Berechnung der Risk Exposure, wird die Funktion $WS(s)$ durch die Multiplikation der Funktionen $WWS(s)$ und $SWS(s)$ gebildet.

In Abbildung 6.1 ist die Funktion $WS(s)$ als blaue Kurve dargestellt. Alle dargestellten Kurven in diesem Kapitel können nur quantitativ analysiert werden, da für dieses allgemeine Konzept keine absoluten Zahlen verwendet und berechnet werden. Basierend auf den Ergebnissen der Literatur- und Interviewstudie (Kapitel 4 und 5) kann jedoch eine qualitative Diskussion über den Verlauf von Kurven durchgeführt werden. Daher werden alle Kurven in dieser Dissertation nur qualitativ diskutiert. Für die Risk Exposure von zu wenig Stabilität nehme ich folgenden Verlauf an: Die Risk Exposure ist bei einem niedrigen Strukturgrad hoch und sinkt stark. Je größer der Strukturgrad wird, desto schwächer sinkt die Funktion. Die Gründe für diese Annahme werden im folgenden Kapitel diskutiert. Zudem wird die Annahme durch die von Boehm und Turner [5] verwendete Darstellung der Risk Exposure von zu wenig Planung unterstützt.

## 6.7 Die Risk Exposure von zu viel Stabilität

Das Risiko von zu viel Stabilität ist das zweite allgemeine Risiko, das bei der Balance von agiler und plan-basierter Entwicklung berücksichtigt werden muss. Dieses Risiko entsteht dadurch, dass sich ein Projekt zu weit rechts auf dem Spektrum des Strukturgrades befindet.

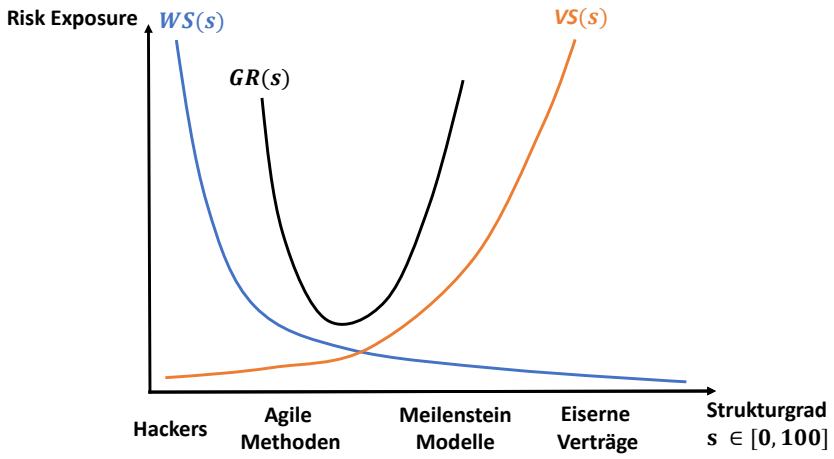


**Abbildung 6.1:** Die Funktionen der Risk Exposure von zu wenig Stabilität (blaue Kurve), zu viel Stabilität (orange Kurve) und des Gesamtrisikos (schwarze Kurve) in Abhängigkeit des Strukturgrades.

Dies bedeutet, dass für ein Projekt Mikromeilensteine existieren, die keinen Raum für Flexibilität lassen. Die Gefahr dadurch ist, dass Projekte zu langsam auf Änderungen oder Probleme reagieren können, da sie einen hohen Aufwand betreiben müssen, um solche Pläne zu ändern.

Der potenzielle Schaden, der dadurch entsteht, ist, dass sich ein Projekt verzögert oder gar nicht auf Anforderungsänderungen reagieren kann. Dies kann entweder zu mehr Kosten im Projekt oder zu einer Unzufriedenheit des Kunden führen.

Für die Risk Exposure des Risikos von zu viel Stabilität wird die folgende Funktion $VS(s)$ definiert. Die Abkürzung VS steht hierbei für zu viel Stabilität.

$$VS(s) = WVS(s) * SVS(s) \quad s \in [0, 100]$$

Die Funktion $VS(s)$ ist abhängig vom Strukturgrad s. Auch hier werden zusätzliche Funktionen für die Berechnung der Wahrscheinlichkeit und des potenziellen Schadens definiert. Die Funktion $WVS(s)$ berechnet abhängig vom Strukturgrad die Wahrscheinlichkeit, mit der die oben beschriebenen Probleme eintreten. Zum Beispiel wächst die Wahrscheinlichkeit, dass es zu Verzögerungen kommt, je detaillierter ein Projektteam plant (hoher Strukturgrad).

Die Funktion $SVS(s)$ bestimmt in Abhängigkeit des Strukturgrades den potenziellen Schaden, der beim Eintreten eines Risikos entstehen kann. Je detaillierter ein Projekt plant, desto größer wird der Schaden, wenn es zu Änderungen kommt, da viele Pläne geändert werden und diese Änderungen im Projekt verteilt werden müssen. Dies kostet viel Zeit und sorgt daher für Verzögerungen.

In Abbildung 6.1 ist die Funktion der Risk Exposure von zu viel Stabilität $VS(s)$ als oran-

ge Kurve eingezeichnet. Für die Kurve nehme ich folgenden Verlauf an: Bei einem niedrigen Strukturgrad ist die Risk Exposure niedrig und steigt schwach an. Je größer der Strukturgrad wird, desto stärker steigt auch die Risk Exposure an. Die Gründe für den Verlauf dieser Kurve werden im nächsten Kapitel diskutiert. Zudem wird die Annahme durch die von Boehm und Turner [5] verwendete Darstellung der Risk Exposure von zu viel Planung unterstützt.

## 6.8 Die Gesamtfunktion der Risk Exposure

Die beiden Risiken von zu wenig und zu viel Stabilität stehen sich bei der Balance von agiler und plan-basierter Entwicklung gegenüber. Welche Risk Exposure diese Risiken haben, ist dabei abhängig vom Strukturgrad. So ist zum Beispiel bei einem geringen Strukturgrad das Risiko von zu wenig Stabilität hoch und bei einem hohen Strukturgrad ist es niedrig. Gegenläufig verhält es sich für das Risiko von zu viel Stabilität.

Beim Risikomanagement geht es stets darum, Risiken zu minimieren [5]. Dies bedeutet, dass ein Projektteam eine Minimierung der Risiken von zu wenig und zu viel Stabilität erreichen wollen. Dafür reicht es nicht aus, dass für den Strukturgrad die Risiken von zu wenig und zu viel Stabilität getrennt voneinander betrachtet werden. Stattdessen muss das Gesamtrisiko betrachtet werden, das ein Strukturgrad für ein Projekt darstellt. Dafür muss die Summe aus der Risk Exposure von zu wenig und zu viel Stabilität in Abhängigkeit des Strukturgrades betrachtet werden.

Für die Betrachtung des Gesamtrisikos müssen die Risiken von zu viel und zu wenig Stabilität summiert werden. Boruszewski [107] beweist in ihrer Arbeit, dass für die Berechnung der gesamten Risk Exposure einer Alternative die Risk Exposure von mehreren unterschiedlichen Risiken unter zwei Voraussetzung addiert werden können. Die eine Voraussetzung ist, dass die Eintrittswahrscheinlichkeiten der Risiken voneinander unabhängig sind. Die zweite Voraussetzung ist, dass die potenziellen Schäden der Risiken voneinander unabhängig sind.

### 6.8.1 Unabhängigkeit der Risiken

Die Unabhängigkeit von Risiken beschreibt, dass die Eintrittswahrscheinlichkeit eines Risikos nicht vom Eintreten eines anderen Risikos abhängt. Für die Risiken von zu wenig und zu viel Stabilität bedeutet dies Folgendes: Die Eintrittswahrscheinlichkeit des Risikos von zu wenig Stabilität wird weder begünstigt noch vermindert durch das Eintreten des Risikos von zu viel Stabilität.

In dieser Dissertation wird angenommen, dass die Eintrittswahrscheinlichkeit der Risiken

von zu wenig und zu viel Stabilität voneinander unabhängig sind. Warum diese Annahme getroffen wird, soll hier an einem Beispiel im Bereich der Architektur verdeutlicht werden. Dafür werden die folgenden zwei Risiken betrachtet:

R1: Das Projektteam hat keinen Überblick über die Software

R2: Die Architektur muss bei Anforderungsänderungen angepasst werden

Das Risiko R1 kann eintreten, wenn das Projektteam zu Beginn eines Projektes zu wenig technische Struktur für eine Software definiert. Dadurch ist es möglich, dass sich die die Architektur im weiteren Verlauf des Projektes eher zufällig entwickelt und das Projektteam den Überblick über die Software verliert. Dieses Risiko ist daher bei einem geringen Strukturgrad hoch und steht stellvertretend für das Risiko von zu wenig Stabilität.

Das Risiko R2 kann eintreten, wenn das Projektteam die Architektur zu Beginn in einem zu großen Detailgrad entwickelt und keine Freiräume lässt. Die Architektur muss dadurch im Fall von Anforderungsänderungen und neuen Erkenntnissen angepasst werden.

R1 wird demnach durch zu wenig Struktur und R2 durch zu viel Struktur ausgelöst. Daran ist zu sehen, dass die Auslöser für diese Risiken gegenläufig sind und damit nicht zusammen auftreten. Das Eintreten des Risikos, dass das Projektteam keinen Überblick über die Architektur hat, begünstigt demnach nicht das Risiko, dass die Architektur bei Anforderungsänderungen angepasst werden muss. Stattdessen sind die beiden Eintrittswahrscheinlichkeiten durch die Gegensätzlichkeit der Risiken von zu wenig und zu viel Stabilität nur vom jeweiligen Strukturgrad abhängig. Aus diesem Grunde wird angenommen, dass die Risiken voneinander unabhängig sind und somit die Funktionen $WWS(s)$ und $WVS(s)$ voneinander unabhängig sind.

### 6.8.2 Unabhängigkeit des Schadens

Die zweite Voraussetzung ist, dass die potenziellen Schäden der Risiken von zu wenig und zu viel Stabilität voneinander unabhängig sind. Zum Beispiel entsteht für R1 ein Schaden von S1 und für R2 ein Schaden von S2. Eine Unabhängigkeit der Schäden bedeutet, dass wenn R1 und R2 zusammen eintreten, der entstehende Gesamtschaden die Summe von S1 und S2 ist. Wenn beide Schäden von einander abhängig wären, würde dies bedeuten, dass beim Eintreten von zwei Schäden der Gesamtschaden entweder kleiner oder größer ist als die Summe aus beiden Schäden. In dieser Dissertation wird angenommen, dass die potenziellen Schäden der Risiken von zu wenig und zu viel Stabilität von einander unabhängig sind.

Zur Erläuterung dieser Annahme wird auch hierfür das Architekturbeispiel verwendet und die beiden Risiken R1 und R2 betrachtet. Bei beiden Risiken muss die Software überarbeitet

werden, wenn die Risiken eintreten. Diese Überarbeitung stellt einen Schaden dar, da sie die Projektlaufzeit verlängert. Es kann jedoch nicht direkt davon ausgegangen werden, dass die beiden Schäden unabhängig voneinander sind. Wenn beide Risiken zusammen eintreten, kann es passieren, dass sich die Schäden überlappen, weil durch die Änderungen gleiche Teile der Software betroffen sind. So kommt es nur zu einer großen Änderung, die möglicherweise insgesamt kleiner ausfällt, als wenn beide Ereignisse getrennt voneinander eingetreten wären. Der Gesamtschaden wäre somit kleiner als die Summe aus beiden Einzelschäden.

In dieser Dissertation wird trotzdem angenommen, dass die Schäden voneinander unabhängig sind. Die Argumentation stützt sich hierbei auf die Betrachtung der Verläufe von $WS(s)$ und $VS(s)$ in Abbildung 6.1. An der Abbildung und der Beschreibung der beiden Risiken ist ersichtlich, dass bei einem geringen Strukturgrad das Risiko von zu wenig Stabilität dominant gegenüber dem Risiko von zu viel Stabilität ist. So ist zum Beispiel der potenzielle Schaden, der bei einem geringen Strukturgrad durch eine nicht sorgfältig definierte Architektur entsteht, viel größer als der durch zu wenige Freiräume. Dies liegt daran, dass nicht sichergestellt ist, dass das Projektteam eine ausreichende Struktur für die Architektur geschaffen hat, um einen Überblick über die Software zu gewinnen. Gleichzeitig hat das Projektteam nur die grundlegenden Architekturentscheidungen getroffen, sodass die Architektur frei weiterentwickelt werden kann. Ein potenzieller Schaden durch eine Überarbeitung der Architektur durch Anforderungsänderungen, ist demnach gering.

Das gleiche gilt umgekehrt bei einem hohen Strukturgrad. Hier sind die Schäden durch zu wenige Freiheitsgrade in der Software deutlich höher als die durch eine zu geringe Architekturdefinition zu Beginn des Projektes. Dies liegt daran, dass zwar viel Struktur für die Architektur geschaffen wurde, gleichzeitig aber auch viele Freiheitsgrade eingeschränkt wurden. Dies bedeutet, dass bei einem geringen Strukturgrad der Schaden von zu viel Stabilität nur einen kleinen Einfluss auf den Gesamtschaden hat und der Schaden von zu wenig einen sehr großen. Gegensätzlich verhält es sich bei einem hohen Strukturgrad. Hier hat der Schaden von zu viel Stabilität einen großen Einfluss und der Schaden von zu wenig Stabilität einen geringen. In dieser Dissertation wird angenommen, dass jeweils der geringere Schaden gegenüber dem größeren sehr klein ausfällt und daher vernachlässigt werden kann. Daraus folgt die Annahme, dass jeweils nur der dominante Schaden den Gesamtschaden beeinflusst. Es wird somit angenommen, dass die potenziellen Schäden der Risiken von zu viel und zu wenig Stabilität unabhängig sind.

Für die Funktionen $SWS(s)$ und $SVS(s)$ wird angenommen, dass diese voneinander unabhängig sind. Dies geschieht unter der Voraussetzung, dass bei einem geringen Strukturgrad $SVS(s)$ vernachlässigt werden kann und $SWS(s)$ bei einem hohen.

### 6.8.3 Gesamtfunktion der Risk Exposure

Unter Berücksichtigung der beiden getroffenen Annahmen kann für die Funktion des Gesamtrisikos (GR) eines Strukturgrades die beiden Funktionen $WS(s)$ und $VS(s)$ miteinander addiert werden. Die entstehende Funktion ist die folgende:

$$GR(s) = WS(s) + VS(s) = WWS(s) * SWS(s) + WVS(s) * SVS(s) \quad s \in [0, 100]$$

Der Verlauf dieser Kurve ist in Abbildung 6.1 als schwarze Kurve dargestellt. Aus den angenommenen Verläufen von $WS(s)$ und $VS(s)$ kann der dargestellte Ablauf abgeleitet werden. Die Risk Exposure von zu wenig Stabilität bei einem niedrigen Strukturgrad ist sehr hoch. Gleichzeitig ist die Risk Exposure von zu viel Stabilität sehr niedrig. Mit steigendem Strukturgrad nähern sich die beiden Risiken an. Danach gehen beide Funktionen, wieder auseinander, da mit einem hohen Strukturgrad die Risk Exposure von zu wenig Stabilität niedrig ist und diese von zu viel Stabilität sehr hoch. Dies führt dazu, dass ich für die Funktion von $GR(s)$ den folgenden Verlauf annehme: Bei einem niedrigen Strukturgrad ist die Risk Exposure hoch und fällt ab, bis sie einen Tiefpunkt erreicht hat. Nach dem Tiefpunkt steigt die Funktion wieder an. Diese Annahme wird durch die von Boehm und Turner [5] verwendete Darstellung der Risk Exposure für das Gesamtrisiko bei der Planung unterstützt.

Wie bereits beschrieben geht es darum, das Gesamtrisiko in einem Projekt zu minimieren. Dies bedeutet, dass ein Projektteam den Strukturgrad soweit erhöhen muss, dass das Gesamtrisiko minimal ist. Damit ist eine ideale Balance der beiden Risiken erreicht. Für die Bestimmung der idealen Balance muss daher der Strukturgrad identifiziert werden, an dem $GR(s)$ den Tiefpunkt hat. Der Tiefpunkt der Kurve liegt an der Stelle, an der die Ableitung der Kurve gleich null ist. Um diese Stelle zu identifizieren wird die Ableitung von $GR(s)$ betrachtet. Zur Vereinfachung werden nur die Funktionen $WS(s)$ und $VS(s)$ ohne die dazugehörigen Wahrscheinlichkeits- und Schadensfunktionen betrachtet. Zur Bestimmung des Tiefpunktes wird die folgende Berechnung angestellt:

$$\begin{aligned} GR'(s) &= WS'(s) + VS'(s) \\ \iff 0 &= WS'(s) + VS'(s) \mid -WS'(s) \\ \iff -WS'(s) &= VS'(s) \end{aligned}$$

Aus der Betrachtung der Ableitung von $GR(s)$ ergibt sich, dass der Tiefpunkt dieser Kurve

erreicht ist, wenn bei einem Strukturgrad die Risk Exposure von zu wenig Stabilität genau so stark sinkt wie die Risk Exposure von zu viel Stabilität steigt. Für die Balance von agiler und plan-basierter Entwicklung bedeutet das daher für die Firmen, dass sie bei der Steigerung des Strukturgrades beständig überprüfen müssen, ob sie dadurch das Risiko von zu wenig Stabilität genau so stark senken, wie sie das Risiko von zu viel Stabilität steigern. Auf diese Weise erkennen Projektteams die Balance von agiler und plan-basierter Entwicklung. Damit dient die Analyse der Risk Exposure nicht nur dazu, die theoretischen Zusammenhänge bei der Balance darzustellen. Stattdessen bietet sie eine Unterstützung für Projektteams, um die Balance in ihrem Entwicklungsprozess zu finden. Dies stellt neben der Schaffung eines allgemeinen Konzepts eine Erweiterung des Ansatzes von Boehm und Turner [5] dar. Diese haben sich nicht damit beschäftigt, wie Projektteams eine ideale Balance in der Planung finden können. Stattdessen beschränken sich die Boehm und Turner [5] nur auf binäre Empfehlungen, zum Beispiel zwischen mehr oder weniger Aufwand für die Planung. Diese Empfehlungen sind sehr abstrakt, da nicht genau klar ist, was mit wenig oder viel Aufwand bei der Planung gemeint ist. Die Verwendung des Strukturgrades ermöglicht hingegen konkretere Handlungsempfehlungen bei der Balance für Firmen zu geben. Das allgemeine Konzept zur Findung der Balance, wird im nächsten Kapitel auf die einzelnen Aktivitäten übertragen.

## 6.9 Kontextbetrachtung bei der Risk Exposure

Die vorangegangenen Abschnitte stellen die zwei generellen Risiken von zu wenig und zu viel Stabilität vor, die bei der Balance von agiler und plan-basierter Entwicklung berücksichtigt werden müssen. In jedem Projekt existieren diese Risiken und sollten von den Projektteilnehmern berücksichtigt werden. Diese Abschnitte beschreiben jedoch einen vom Kontext losgelösten Zusammenhang dieser beiden Risiken.

Die Risiken bestehen nicht nur in einer generellen Weise, sondern sind zudem vom Kontext abhängig. Zum Beispiel ist in einem Projekt, das aus vielen Entwicklerteams besteht, Langzeitplanung von entscheidendem Vorteil, um die Abhängigkeiten zwischen diesen Teams zu planen. Das heißt, das Risiko von zu wenig Langzeitplanung (hier gleich bedeutend mit dem Risiko von zu wenig Stabilität) ist in einem solchen Fall generell viel größer als in einem Projekt, das nur aus einem Entwicklerteam besteht.

Im Gegensatz dazu ist das Risiko Pläne ständig überarbeiten zu müssen (hier gleich bedeutend mit dem Risiko von zu viel Stabilität) in einem Projekt mit vielen Unsicherheiten deutlich höher als in einem Projekt mit viel Sicherheit. Diese Sicherheit kann zum Beispiel die Bekanntheit der Anforderungen sein. In einem Projekt, in dem die Anforderungen sehr bekannt sind, ist

das Risiko Pläne überarbeiten zu müssen generell geringer.

Dies bedeutet, dass für die Balance von agiler und plan-basierter Entwicklung neben dem Strukturgrad auch der Kontext betrachtet werden muss. Der Kontext eines Projektes hat einen Einfluss auf den Verlauf der Risk Exposure von zu wenig und zu viel Stabilität und somit auch auf den Tiefpunkt der Kurve von $GR(s)$. Diese Argumentation wird gestützt durch Boehm und Turner [5], die den Einfluss der Systemgröße auf die Risk Exposure bei der Planung diskutieren. Sie beschreiben, dass sich bei einer kleinen Systemgröße der Tiefpunkt des Gesamtrisikos bei der Planung nach links verschiebt. Darüber hinaus diskutieren sie die Rolle des Kontextes nur für die Wahl des Entwicklungsansatzes (agil, plan-basiert oder hybrid) und gehen nicht weiter darauf ein, welche anderen Kontextfaktoren einen Einfluss auf die Balance haben. Zudem findet keine Diskussion über die Balance in den anderen Aktivitäten statt. Diese Lücken werden von dieser Dissertation geschlossen. Damit der Kontext bei der Balance von allen Aktivitäten mit einbezogen werden kann, muss das allgemeine Konzept dieser Dissertation um den Kontext erweitert werden.

Da der Kontext einen Einfluss auf die Funktionen $WS(s)$, $VS(s)$ und $GR(s)$ hat, muss dieser als weitere Variable in diese Funktionen integriert werden. Dies kann auf zwei Arten geschehen. Zum einen kann der Kontext eine weitere Variable sein, von der die drei Funktionen abhängig sind. Dies würde bedeuten, dass für diese Funktionen jeweils eine dreidimensionale Kurve entstehen würde. Die Interpretation einer solchen Kurve ist durch die Verzerrung bei der Darstellung jedoch deutlich schwerer als die einer zweidimensionalen Kurve. Die Analyse der Balance würde damit deutlich verkompliziert werden. Ein weiterer Nachteil ist, dass es für die Balance von agiler und plan-basierter Entwicklung suggeriert, dass der Kontext eines Projektes variabel ist. In dieser Dissertation wird jedoch angenommen, dass der Kontext eines Projektes statisch ist und die Balance daran ausgerichtet wird.

Eine bessere Möglichkeit ist daher, die Funktion für die Risk Exposure als Funktionsschar zu definieren. Dies bedeutet, dass in einem Fall der Kontext eines Projektes als feststehend betrachtet wird und nur den Verlauf der Funktionen $WS(s)$, $VS(s)$ und $GR(s)$ verändert. Darüber hinaus bleiben die Funktionen auf diese Weise nur vom Strukturgrad abhängig .

### 6.9.1 Definition des Kontextes

Bei der Bildung eines Entwicklungsansatzes müssen unterschiedliche Aspekte der Umgebung eines Projektes berücksichtigt werden [10]. Zum Beispiel müssen in einem Projekt mit einem großen Projektteam andere Methoden angewendet werden als mit einem kleinen. Weitere Aspekte, die einen Einfluss auf einen Entwicklungsansatz haben, sind zum Beispiel die Menge an Stakeholdern, die Bekanntheit der Anforderungen oder die Einstellung des Kunden [10].

Diese Umgebung des Projektes wird in dieser Dissertation als *Kontext* definiert. Ein Kontext besteht dabei aus den unterschiedlichen Aspekten, die für die Bildung eines Entwicklungsansatzes berücksichtigt werden. Diese Aspekte werden in dieser Dissertation *Kontextfaktoren* genannt. Beispiele für solche Kontextfaktoren sind die Größe oder Verteilung eines Projektteams.

Damit entschieden werden kann, wieviel Einfluss ein Kontextfaktor auf die Erstellung eines Entwicklungsansatzes hat, muss dieser von einem Projektteam bewertet werden können. Dafür muss dem Kontextfaktor jeweils ein Wert zugewiesen werden. Bei der Größe des Projektteams kann dieser zum Beispiel durch die involvierte Personenanzahl angegeben werden. Es ist auch möglich, eine qualitative Aussage zu treffen, in dem die Stufen klein, mittel und groß definiert werden. Anhand des Wertes für einen Kontextfaktor, kann ein Projekt Entscheidungen über den Entwicklungsansatz treffen. Ein Projekt mit vielen Personen sollte zum Beispiel mehr Wert auf Dokumentation legen, ein Projekt mit wenigen Personen dagegen mehr auf offene Kommunikation [10]. Basierend auf dem Kontextfaktor und dem zugehörigen Wert, kann ein Projekt Entscheidungen zu einem Entwicklungsansatz treffen.

Ein Kontextfaktor und der dazugehörige Wert werden in dieser Dissertation als Tupel definiert. Diese Tupel werden *Kontextelemente* genannt. Eine Menge an Kontextelementen bildet den *Kontext* eines Projektes.

$$Kontextelement\ E_i = (Kontextfaktor\ f, Wert\ w)\ \ i \in \mathbb{N}$$
$$Kontext\ K = \{E_1, E_2, ..., E_n\}\ \ n \in \mathbb{N}$$

### 6.9.2 Kontextbetrachtung der Risk Exposure von zu wenig Stabilität

In diesem Abschnitt wird der Einfluss des Kontextes auf die Risk Exposure des Risikos von zu wenig Stabilität betrachtet. Dafür wird zuerst die bereits vorgestellte Funktion $WS(s)$ um den Kontext erweitert und als Funktionsschar definiert:

$$WS_K(s) = WWS_K(s) * SWS_K(s);\ \ s \in [0, 100]$$

Im Folgenden wird die Auswirkung des Kontextes auf den Verlauf der Kurve der Risk Exposure von zu wenig Stabilität diskutiert. Dafür nehme ich drei beispielhafte Kontexte $K_1$, $K_2$ und $K_3$ an, um jeweils einen anderen Verlauf der Risk Exposure betrachten zu können. Die beispielhaften Kontexte sind in Tabelle 6.3 aufgelistet.

Kritische nicht-funktionale Anforderungen werden als solche nicht-funktionalen Anforderun-

**Tabelle 6.3:** Beispielhafte Kontexte für die Analyse der Risk Exposure von zu wenig Stabilität

| Kontext | Beschreibung | Kontexfaktoren | Werte |
|---|---|---|---|
| $K_1$ | In diesem Kontext gibt es neben dem Kunden keine weiteren Stakholder und keine kritischen nicht-funktionalen Anforderungen und Abhängigkeiten. | Menge an Stakholdern | keine |
| | | Menge der kritischen nicht-funktionalen Anforderungen | keine |
| | | Menge an Abhängikeiten | keine |
| $K_2$ | In diesem Kontext gibt es keine kritischen nicht-funktionalen Anforderungen. Es gibt jedoch im Vergleich zu $K_1$ eine Menge an Stakeholdern und Abhängigkeiten zu anderen Projekten. | Menge der Stakeholder | wenige |
| | | Menge der kritischen nicht-funktionalen Anforderungen | keine |
| | | Menge an Abhängigkeiten | wenige |
| $K_3$ | In diesem Kontext gibt es viele kritische nicht-funktionale Anforderungen. Zudem gibt es mehr Stakeholder und Abhängigkeiten zu anderen Projekten als in $K_2$. | Menge der Stakeholder | viele |
| | | Menge der kritischen nicht-funktionalen Anforderungen | viele |
| | | Menge an Abhängigkeiten | viele |

gen betrachtet, die direkt zu Beginn des Projektes berücksichtigt werden müssen, damit sie erfüllt werden können. Zusätzlich zu diesen drei beispielhaften Kontexten nehme ich zusätzlich einen beispielhaften Strukturgrad $s_1$ auf der Abszissenachse an. Dieser ist in Tabelle 6.4 beschrieben und stellt einen geringen Stand erreichter Struktur dar.

Basierend auf diesem Strukturgrad wird die Risk Exposure von zu wenig Stabilität zwischen den definierten Kontexten verglichen. Das Risiko von zu wenig Stabilität entsteht im Bereich des Requirements Engineerings dadurch, dass zu Beginn eines Projektes wichtige Anforderungen übersehen werden. Im Strukturgrad $s_1$ beschränkt sich das Projektteam nur auf die funktionalen Anforderungen. Generell entsteht dadurch die Gefahr, dass zu Beginn wichtige Anforderungen übersehen werden.

**Tabelle 6.4:** Beschreibung von Strukturgrad $s_1$

| Beschreibung | | Wert |
|---|---|---|
| $s_1$ | Es werden von dem Projektteam nur die funktionalen Anforderungen erhoben. | gering<br>$20 < s_1 \leq 40$ |

Die Wahrscheinlichkeit, dass dies bei diesem Strukturgrad passiert ist im Kontext $K_3$ besonders hoch, da das Projekt viele kritische nicht-funktionale Anforderungen, Abhängigkeiten und Stakeholder hat. Auch der potenzielle Schaden ist in diesem Kontext sehr hoch, da kritische nicht-funktionale Anforderungen direkt zu Beginn eines Projektes berücksichtigt werden müssen. Ansonsten kann die Missachtung dieser Anforderungen zu späteren Überarbeitungen

der Software führen. Dies verlängert die Projektlaufzeit und kostet Geld. Damit ist die Risk Exposure von zu wenig Stabilität in Kontext $K_3$ generell groß.

Die Risk Exposure von zu wenig Stabilität in Kontext $K_2$ ist geringer als in $K_3$, da das Projekt $K_2$ in weniger Stakeholder und Abhängigkeiten hat als $K_3$. Die Wahrscheinlichkeit, Anforderungen zu übersehen, ist daher bei gleichem Strukturgrad in $K_2$ geringer als in $K_3$. Der Schaden ist ebenfalls geringer, da das Projekt in $K_2$ keine kritischen nicht-funktionalen Anforderungen besitzt. Dies bedeutet, dass übersehene nicht-funktionale Anforderungen auch noch später in die Software integriert werden können und dabei weniger Aufwand als in $K_3$ verursachen. Die Risk Exposure des Strukturgrades $s_1$ ist daher in $K_2$ kleiner als in $K_3$.

Das Risiko von zu wenig Stabilität ist in $K_1$ im Vergleich zu $K_2$ und $K_3$ am wenigsten kritisch. Dadurch dass es keine Stakeholder neben dem Kunden und keine Abhängigkeiten gibt, ist die Wahrscheinlichkeit im Vergleich zu $K_2$ und $K_3$ gering, dass Anforderungen übersehen werden. Das Projekt besteht vorwiegend aus funktionalen Anforderungen. Wenn hierbei Anforderungen übersehen werden, ist es einfacher, diese später noch hinzuzufügen. Damit ist die Risk Exposure von $s_1$ in $K_1$ kleiner als in $K_2$ und $K_3$.

Aus dieser Betrachtung wird ersichtlich, dass der Kontext einen Einfluss auf die Risk Exposure eines bestimmten Strukturgrades hat und damit auch auf die gesamte Kurve der Risk Exposure. In Abbildung 6.2 sind die drei Kurven, die durch die drei Kontexte entstehen, dargestellt. Nach der obigen Betrachtung ist die Funktion von $WS_{K_3}(s)$ am größten und die Funktion von $WS_{K_1}(s)$ am kleinsten. In der Mitte von beiden Funktionen liegt die Kurve von $WS_{K_2}(s)$. Durch die obige Betrachtung, kann abgeleitet werden, dass das das Risiko von zu wenig Stabilität in $K_3$ sehr kritisch, in $K_2$ moderat kritisch und in $K_1$ wenig kritisch ist. Dies bedeutet, dass die Risk Exposure von $K_1$ zu $K_3$ immer größer wird. Die Funktion $WS_K(s)$ verschiebt sich dadurch im Koordinatensystem immer weiter nach rechts oben und wird damit immer größer. Dieser Zusammenhang ist in Abbildung 6.2 durch den schwarzen Pfeil dargestellt.

An dieser Stelle wird die *Kritikalität* des Risikos von zu wenig Stabilität als der Grad der Verschiebung der Funktion $WS_K(s)$ nach rechts oben im Koordinatensystem definiert. Dadurch wird festgelegt, wie gefährlich beziehungsweise kritisch das Risiko durch einen Kontext ist. Zur Vereinfachung wird an dieser Stelle angenommen, dass der Kontext nur für eine Verschiebung der Kurve sorgt und nicht deren Verlauf ändert.

### 6.9.3 Kontextbetrachtung der Risk Exposure von zu viel Stabilität

In diesem Abschnitt wird der Einfluss des Kontextes auf die Risk Exposure von zu viel Stabilität betrachtet. Dafür wird auch hier zuerst die bereits vorgestellte Funktion $VS(s)$ für die Risk Exposure um den Kontext erweitert und ebenfalls als Funktionsschar definiert:

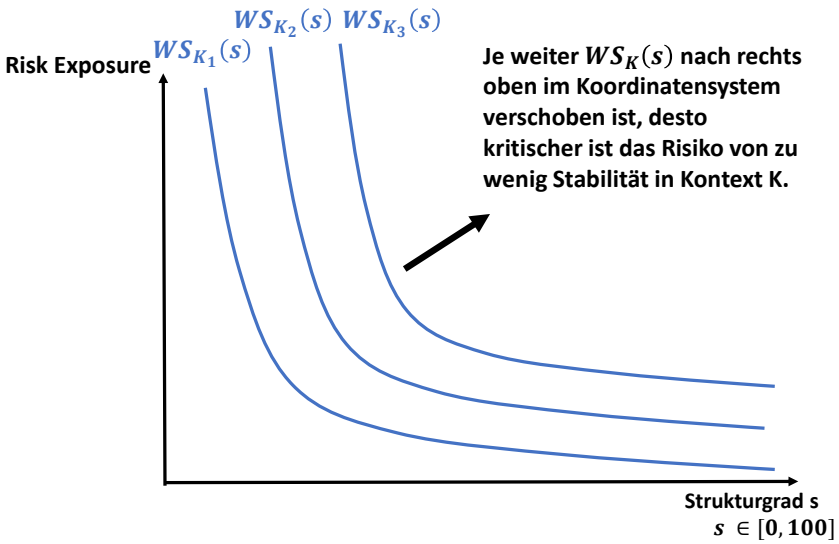


**Abbildung 6.2:** Die Verschiebung von $WS_K(s)$ durch unterschiedliche Kontexte

$$VS_K(s) = WVS_K(s) * SVS_K(s); \;\; s \in [0, 100]$$

Im Folgenden wird die Auswirkung des Kontextes auf den Verlauf der Risk Exposure von zu viel Stabilität diskutiert. Dafür nehme ich drei beispielhafte Kontexte $K_4$, $K_5$ und $K_6$ an. Die beispielhaften Kontexte sind in Tabelle 6.5 aufgelistet.

**Tabelle 6.5:** Beispielhafte Kontexte für die Analyse der Risk Exposure von zu viel Stabilität

| Kontexte | Beschreibung | Kontextfaktoren | Werte |
|---|---|---|---|
| $K_4$ | In diesem Kontext ist sowohl bekannt aus welchen Features die Software bestehen soll, als auch die Funktionen dieser Features. | Bekanntheit der Anforderungen | hoch |
| $K_5$ | In diesem Kontext ist bekannt aus welchen Features die Software bestehen soll, jedoch herrscht teilweise Unsicherheit über die Funktionen der Features. | Bekanntheit der Anforderungen | mittel |
| $K_6$ | In diesem Kontext ist sowohl unsicher aus welchen Features die Software bestehen soll, als auch welche Funktionen diese haben sollen. | Bekanntheit der Anforderungen | gering |

Zusätzlich zu diesen drei beispielhaften Kontexten nehme ich einen beispielhaften Strukturgrad $s_2$ auf der Abszissenachse an. Dieser ist in Tabelle 6.6 beschrieben und stellt einen sehr hohen Stand erreichter Struktur dar.

Basierend auf dem Strukturgrad $s_2$ wird die Risk Exposure von zu viel Stabilität zwischen den drei Kontexten verglichen. Das Risiko von zu viel Stabilität entsteht im Bereich des Requi-

**Tabelle 6.6:** Beschreibung des Strukturgrades $s_2$

| Beschreibung | | Wert |
|---|---|---|
| $s_2$ | Das Projektteam hat zu Beginn des Projektes alle Anforderungen des Projektes zusammen mit dem Umfang festgelegt. | sehr hoch $80 < s_2 \leq 100$ |

rements Engineering dadurch, dass zu viele Anforderungen bereits zu Beginn des Projektes festgelegt werden. Im Strukturgrad $s_2$ legt das Projektteam bereits alle Anforderungen zu Beginn des Projektes fest.

Dies bedeutet, dass sich Stakeholder trotz Unsicherheit auf Anforderungen festlegen müssen. Dies kann dazu führen, dass falsche Anforderungen aufgenommen werden, die später zu einer Änderung der Software führen.

In Kontext $K_4$ sind die Anforderungen vollständig bekannt. Die Wahrscheinlichkeit ist also gering, dass am Anfang falsche Anforderungen aufgenommen werden, die später zu Change Requests führen. Der entstehende Schaden ist ebenfalls gering, da sich bei einer hohen Sicherheit der Anforderungen eher Details ändern, als ganze Hauptfeatures. Damit ist das Risiko von zu viel Stabilität gering in diesem Kontext.

Im Vergleich ist das Risiko in Kontext $K_5$ kritischer, da hier eine größere Unsicherheit der Anforderungen existiert. Die Wahrscheinlichkeit ist hier größer, dass zu Beginn falsche Anforderungen festgelegt werden, die am Ende des Projektes zu einer Unzufriedenheit der Kunden beitragen. Der Schaden ist in $K_5$ ebenfalls größer, da bei falschen funktionalen Anforderungen Teile der Software überarbeitet werden müssen. Dies verlängert die Projektlaufzeit und erhöht die Kosten. Damit ist das Risiko von zu Stabilität in $K_5$ größer als in $K_4$

Am kritischsten ist das Risiko in $K_6$, da im Gegensatz zu $K_5$ auch die Funktionen der Features unsicher sind. Die Wahrscheinlichkeit, dass am Anfang falsche Anforderungen aufgenommen werden, ist in diesem Kontext daher generell höher. Zudem kann in diesem Kontext ein größerer Schaden entstehen. Wenn die Software am Ende nicht die richtigen Features oder Funktionen besitzt, ist sie möglicherweise vom Kunden nicht nutzbar und muss komplett geändert werden. Dies stellt einen hohen Schaden in Kosten und Reputation dar.

Aus dieser Betrachtung wird ersichtlich, dass der Kontext ebenfalls einen Einfluss auf die Risk Exposure von zu viel Stabilität hat. In Abbildung 6.3 sind die drei Kurven der Risk Exposure, die durch die drei unterschiedlichen Kontexte entstehen, dargestellt. Nach der obigen Diskussion ist die Funktion $VS_{K_6}(s)$ am größten und die Funktion $VS_{K_4}(s)$ am kleinsten. In der Mitte zwischen beiden Funktionen befindet sich die Funktion $VS_{K_5}(s)$. Durch die obige Betrachtung, kann abgeleitet werden, dass das Risiko von zu viel Stabilität in $K_4$ wenig kritisch, in $K_5$ moderat kritisch und in $K_6$ am kritischsten ist. Dies bedeutet, dass die Risk Exposure

von $K_4$ zu $K_6$ immer größer wird. Die Kurve von $VS_K(s)$ verschiebt sich dadurch immer weiter nach links oben im Koordinatensystem und wird somit immer größer. Dieser Zusammenhang ist in Abbildung 6.3 durch den schwarzen Pfeil dargestellt.

An dieser Stelle wird die Kritikalität des Risikos von zu viel Stabilität als Grad der Verschiebung der Funktion $VS_K(s)$ nach links oben im Koordinatensystem definiert. Dadurch wird festgelegt, wie gefährlich beziehungsweise kritisch das Risiko durch einen Kontext ist.

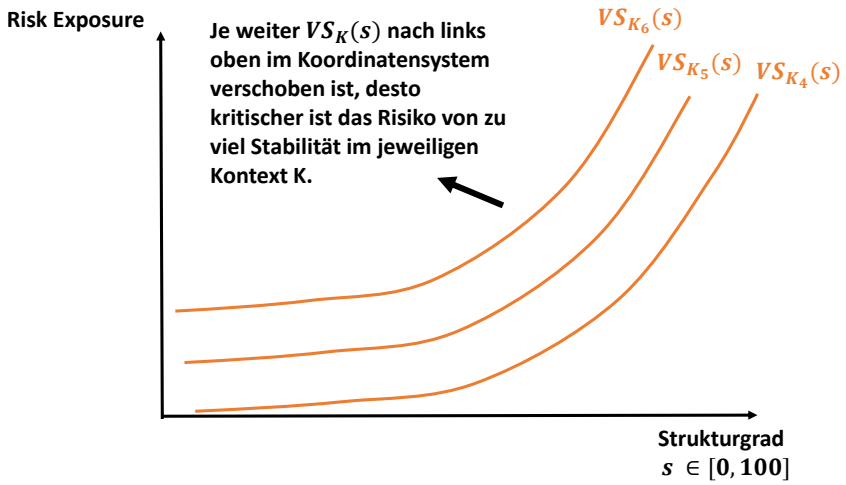


**Abbildung 6.3:** Die Verschiebung von $VS_K(s)$ durch unterschiedliche Kontexte

### 6.9.4 Kontextbetrachtung der Gesamtfunktion

In den vorherigen Abschnitten wurde der Einfluss des Kontextes jeweils auf die Risk Exposure von zu wenig und zu viel Stabilität analysiert. In diesem Abschnitt erfolgt die Analyse des Einflusses des Kontextes auf die Gesamtfunktion der Risk Exposure. Dafür wird auch hier zunächst die Funktion $GR(s)$ um den Kontext erweitert und ebenfalls als Funktionsschar definiert:

$$GR_K(s) = WWS_K(s) * SWS_K(s) + WVS_K(s) * SVS_K(s); \;\; s \in [0, 100]$$

Im Folgenden wird die Auswirkung der unterschiedlichen Kontexte auf die Balance der beiden Risiken von zu wenig und zu viel Stabilität untersucht. Dafür wird analysiert, welche Auswirkung ein Kontext auf die Lage des Tiefpunktes von $GR_K(s)$ hat, da dieser den idealen Balancepunkt zwischen den beiden Risiken darstellt.

Für die Betrachtung von $GR_K(s)$ werden die bereits beispielhaft angenommenen Kontexte $K_1$ - $K_6$ genutzt. Die Kontexte $K_1$ - $K_3$ haben einen Einfluss auf den Verlauf von $WS_K(s)$ und

bilden die Menge $SK = \{K_1, K_2, K_3\}$. Die Kontexte $K_4$ - $K_6$ haben einen Einfluss auf den Verlauf von $VS_K(s)$ und bilden die Menge $FK = \{K_4, K_5, K_6\}$.

In den Kontexten der beiden Gruppen, ist die Kritikalität der beiden Risiken von zu wenig und zu viel Stabilität unterschiedlich. Für die Analyse von $GR_K(s)$ wird untersucht, welchen Einfluss die unterschiedlichen Verläufe der Funktionen $WS_K(s)$ und $VS_K(s)$ auf den Verlauf von $GR_K(s)$ haben. Dafür müssen auch die jeweiligen Teilkontexte aus $SK$ und $FK$ zusammen betrachtet werden, da diese jeweils einen unterschiedlichen Verlauf der Funktionen $WS_K(s)$ und $VS_K(s)$ herbeiführen. Es wird daher jeweils ein Teilkontext aus $SK$ mit einem Teilkontext aus $FK$ vereinigt (siehe Tabelle 6.7). Dies ist möglich, da es zwischen diesen beiden Mengen keine Überlappung der Kontextfaktoren gibt. In der Menge $SK$ werden die Art der Anforderungen betrachtet, die es in einem Projekt gibt und in Menge $FK$ die Bekanntheit der Anforderungen.

Für die Untersuchung von $GR_K(s)$ werden die Auswirkungen von Kombinationen der unterschiedlichen Teilkontexte aus $SK$ und $FK$ analysiert. Zur Vereinfachung der Untersuchung wird nur zwischen wenig, moderat und hoch kritischen Teilkontexten unterschieden. Dies bedeutet, dass zum Beispiel die Auswirkungen von der Kombination von $K_1$ mit $K_4$ auf $GR_K(s)$ analysiert wird. In Tabelle 6.7 sind alle neun möglichen Kombinationen aufgelistet:

**Tabelle 6.7:** Liste der Kontextkombinationen

| Kritikalität des Risikos von zu wenig Stabilität | Kritikalität des Risikos von zu viel Stabilität | Gesamtkontext |
|---|---|---|
| gering ($K_1$) | gering ($K_4$) | $K_1 \cup K_4$ |
| gering ($K_1$) | moderat ($K_5$) | $K_1 \cup K_5$ |
| gering ($K_1$) | hoch($K_6$) | $K_6 \cup K_4$ |
| moderat ($K_2$) | gering ($K_4$) | $K_2 \cup K_4$ |
| moderat ($K_2$) | moderat ($K_5$) | $K_2 \cup K_5$ |
| moderat ($K_2$) | hoch ($K_6$) | $K_2 \cup K_6$ |
| hoch ($K_3$) | gering ($K_4$) | $K_3 \cup K_4$ |
| hoch ($K_3$) | moderat ($K_5$) | $K_3 \cup K_5$ |
| hoch ($K_3$) | hoch ($K_6$) | $K_3 \cup K_6$ |

Bei den Beschreibungen in diesem Kapitel handelt es sich um ein Konzept, daher wird die Risk Exposure in allen Fällen ausschließlich qualitativ analysiert. Für die qualitative Analyse der Auswirkung der unterschiedlichen Kontextkombinationen benötigt es einen Fall, zu dem diese Kombinationen verglichen werden können. Hierfür wird die Kombination von $K_2$ und $K_5$ gewählt.

Im Teilkontext $K_2$ ist das Risiko von zu wenig Stabilität moderat kritisch. Gleichzeit ist im Teilkontext $K_5$ das Risiko von zu viel Stabilität moderat kritisch. Bei einer Vereinigung dieser beiden Teilkontexte sind die Risiken von zu wenig und zu viel Stabilität jeweils moderat kritisch in einem Projekt. Beide Risiken sind gleichstark in dem Kontext $K_2 \cup K_5$ und keines ist domi-

nant gegenüber dem anderen. Beide Risiken sind somit ausgeglichen. $WS_K(s)$ und $VS_K(s)$ haben im Kontext $K_2 \cup K_5$ einen gleichstarken Einfluss auf $GR_K(s)$ und somit auf die Balance von beiden Risiken. In anderen Kombinationen der Kontexte haben die Risiken teilweise einen unterschiedlich starken Einfluss bei der Balance.

Diese Kombination eignet sich daher als Ausgangssituation für $GR_K(s)$, mit der die anderen Teilkontextkombinationen verglichen werden können Die Ausgangssituation von $GR_{K_2 \cup K_5}$ wird als schwarze Kurve in Abbildung 6.4 dargestellt. Zur besseren Übersicht wurde auf eine zusätzliche Darstellung der Funktionen von $WS_K(s)$ und $VS_K(s)$ in dieser Abbildung verzichtet.

Auf Basis dieser Ausgangssituation wird das folgende Beispiel betrachtet. In der Ausgangssituation wird $K_5$ durch $K_4$ ersetzt. Mit diesem Austausch verschiebt sich die Kurve der Risk Exposure von zu viel Stabilität ($VS_K(s)$) nach rechts unten im Koordinatensystem, da in Kontext $K_4$ das Risiko von zu viel Stabilität weniger kritisch ist als in $K_5$. Die Risk Exposure von zu viel Stabilität sinkt generell in dieser Kontextkombination. Die Funktion hat damit einen geringeren Einfluss auf das Gesamtrisiko und das Risiko von zu wenig Stabilität gewinnt einen größeren Einfluss. Beide Risiken sind nun nicht mehr ausgeglichen, sondern das Risiko von zu viel Stabilität wird durch das Risiko von zu wenig Stabilität dominiert. Diese Dominanz hat einen Einfluss darauf, wie sich der Tiefpunkt verschiebt.

Für die Untersuchung der einzelnen Kontextkombinationen kommt es daher darauf an, zu untersuchen, in welchen Fällen, ausgehend von der Normalsituation, eines der beiden Risiken gegenüber dem anderen Risiko einen größeren Einfluss auf das Gesamtrisiko hat. Dafür werden die aufgelisteten Kombinationen in drei Fälle aufgeteilt:

**1. Fall:** Im ersten Fall hat die Funktion $WS_K(s)$ im Vergleich zu der Ausgangssituation einen stärkeren Einfluss auf $GR_K(s)$ als $VS_K(s)$. Damit ist die Risk Exposure von zu wenig Stabilität im Vergleich zu der Risk Exposure von zu viel Stabilität dominant. Dies ist in den folgenden Kontextkombinationen der Fall: $K_2 \cup K_4$, $K_3 \cup K_4$ und $K_3 \cup K_5$.

**2. Fall:** Im zweiten Fall hat die Funktion $VS_K(s)$ im Vergleich zur Ausgangssituation einen stärkeren Einfluss auf $GR_K(s)$ als $WS_K(s)$. Damit ist die Risk Exposure von zu viel Stabilität im Vergleich zur Risk Exposure von zu wenig Stabilität dominant. Dies ist in den folgenden Kontextkombinationen der Fall: $K_1 \cup K_5$, $K_1 \cup K_6$ und $K_2 \cup K_6$.

**3. Fall:** Im dritten Fall haben beide Funktionen $WS_K(s)$ und $VS_K(s)$ einen gleich großen oder geringen Einfluss auf $GR_K(s)$. Damit sind die Risk Exposure von zu viel und zu wenig Stabilität ausgeglichen und keines der Risiken ist gegenüber dem anderen dominant. Dies ist in den folgenden Kontextkombinationen der Fall: $K_1 \cup K_4$, $K_2 \cup K_5$ und $K_3 \cup K_6$.

**1. Fall: Risk Exposure von zu wenig Stabilität ist dominant**

Ausgehend von der Ausgangssituation wird die Kombination von $K_3 \cup K_5$ betrachtet. Bei dieser Kombination ist die Funktion $WS_K(s)$ im Vergleich zur Ausgangssituation nach rechts oben im Koordinatensystem verschoben und somit größer. Die Funktion $VS_K(s)$ hat währenddessen den gleichen Verlauf. Durch die Verschiebung von $WS_K(s)$ stellt sich die Gleichheit der Ableitung erst bei einem höheren Strukturgrad ein. Damit wandert der Tiefpunkt von $GR_K(s)$ und damit die Balance im Vergleich zur Ausgangssituation weiter nach rechts. Die Verschiebung von $GR_K(s)$ ist in Abbildung 6.4 als blaue Kurve eingezeichnet. Der höhere Strukturgrad ist nötig, um in diesem Kontext dem größeren Risiko von zu wenig Stabilität zu begegnen. Der Tiefpunkt liegt ebenfalls auf der y-Achse höher, da durch die Verschiebung ein insgesamt höheres Gesamtrisiko erreicht wird.

Der Tiefpunkt von $GR_K(s)$ wandert im Vergleich zur Ausgangssituation immer weiter nach rechts auf der x-Achse, je mehr $WS_K(s)$ nach rechts oben verschoben ist und somit größer wird. Wenn gleichzeitig $VS_K(s)$ kleiner wird, wie zum Beispiel in der Kombination $K_3 \cup K_4$, wandert der Tiefpunkt noch weiter nach rechts, da das Risiko von zu viel Stabilität immer mehr an Einfluss verliert und ein höherer Strukturgrad genutzt werden kann.

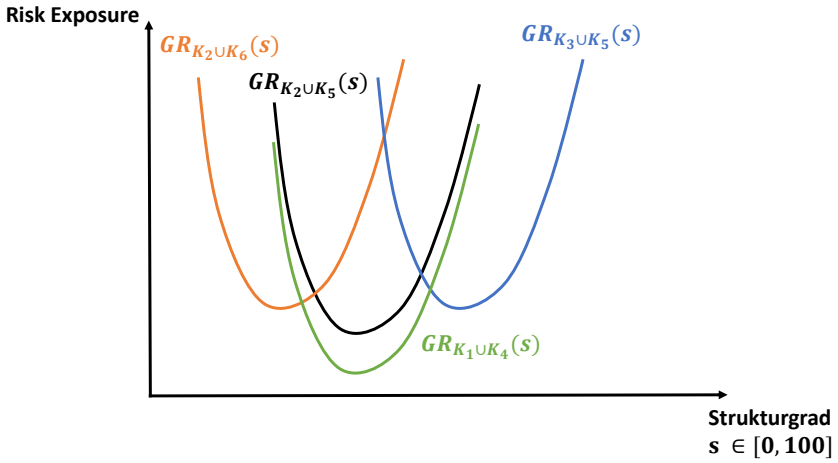


**Abbildung 6.4:** Darstellung der Verschiebung von $GR_K(s)$ in Abhängigkeit des Kontextes

**2. Fall: Risk Exposure von zu viel Stabilität ist dominant**

Für diesen Fall wird die Kontextkombination $K_2 \cup K_6$ betrachtet. Bei dieser Kombination ist die Funktion $VS_K(s)$ nach links oben im Koordinatensystem verschoben und größer als in der Ausgangssituation, während $WS_K(s)$ unverändert bleibt. Durch die Verschiebung von $VS_K(s)$ nach links oben stellt sich die Gleichheit der Ableitung bei einem kleineren Strukturgrad ein. Damit wandert der Tiefpunkt von $GR_K(s)$ und damit die Balance im Vergleich zur Ausgangssi-

tuation auf der x-Achse nach links. Die Verschiebung der Kurve von $GR_K(s)$ ist für diesen Fall in Abbildung 6.4 als orange Kurve eingezeichnet. Der geringe Strukturgrad ist nötig, um dem erhöhten Risiko von zu viel Stabilität zu begegnen. Auch hier wandert der Tiefpunkt auf der y-Achse nach oben, da das Gesamtrisiko im Vergleich zur Ausgangssituation größer geworden ist.

Der Tiefpunkt wandert immer weiter nach links auf der x-Achse, je größer $VS_K(s)$ wird. Wenn gleichzeitig auch $WS_K(s)$ kleiner wird, wie zum Beispiel in der Kombination von $K_1 \cup K_6$, wandert der Tiefpunkt noch weiter nach links. Dies liegt daran, dass das Risiko von zu viel Stabilität immer mehr an Einfluss gewinnt und ein geringerer Strukturgrad genutzt werden muss.

**3. Fall: Risk Exposure von zu wenig und zu viel Stabilität ist ausgeglichen**

In diesem Fall werden die Funktionen $WS_K(s)$ und $VS_K(s)$ gleichzeitig verschoben. Zum Beispiel bei der Kontextkombination von $K_1 \cup K_4$ wird $WS_K(s)$ im Vergleich zur Ausgangssituation nach links unten und $VS_K(s)$ nach rechts unten verschoben. Dadurch bleibt die Stelle, an der die Ableitung der beiden Funktionen gleich ist, an der gleichen Stelle auf der x-Achse. Damit verbleibt auch der Tiefpunkt von $GR_K(s)$ und damit die Balance an der selben Stelle. Dadurch dass beide Funktionen kleiner werden, wandert der Tiefpunkt auf der y-Achse weiter nach unten, da das Gesamtrisiko gesunken ist. Die Verschiebung von $GR_K(s)$ ist für diesen Fall in Abbildung 6.4 als grüne Kurve dargestellt.

In der Kontextkombination von $K_3 \cup K_6$ geschieht das Gegenteil. In dieser Kombination werden die beiden Funktionen im Vergleich zur Ausgangssituation größer. Die Balance verbleibt an der selben Stelle. Der Tiefpunkt wandert jedoch auf der y-Achse weiter nach oben, da das Gesamtrisiko gestiegen ist.

# 7

# Prinzipien für die Balance von APH-Ansätzen

Im vorherigen Kapitel 6 wurde ein allgemeines Konzept für die Balance von agiler und plan-basierter Entwicklung vorgestellt. Dabei wurde beschrieben, wie die allgemeinen Risiken von zu wenig und zu viel Stabilität bei der Balance von agiler und plan-basierter Entwicklung gegeneinander abgewägt werden können. Dieses allgemeine Konzept muss nun auf die Balance in den einzelnen Aktivitäten eines Entwicklungsprozesses übertragen werden. In Kapitel 4 und 5 wurde identifiziert, dass für eine Kombination von agiler und plan-basierter Entwicklung die Aktivitäten von Requirements Engineering, Architektur, Planung, Testen und Koordination balanciert werden müssen.

In diesem Kapitel wird das allgemeine Konzept für die Balance auf die einzelnen Aktivitäten übertragen. Dafür wird für jede Aktivität beschrieben worin, das Risiko von zu viel und zu wenig Stabilität in der jeweiligen Aktivität besteht. Zusätzlich wird beschrieben, worin der Strukturgrad für die einzelnen Aktivitäten besteht und welche Kontextfaktoren die Kritikalität des jeweiligen Risikos beeinflussen.

Um Firmen eine Anleitung für die Balance von agiler und plan-basierter Entwicklung zur Verfügung zu stellen, werden daraus Prinzipien für die Balance entwickelt. Beck und Andres

[16] definieren Prinzipien als domänenspezifische Richtlinien, um Verhalten in einem Softwareprojekt zu steuern. Basierend darauf werden in diesem Kapitel Prinzipien definiert, um Richtlinien für die Balance von agiler und plan-basierter Entwicklung zu erstellen.

## 7.1 Ableitung von Risikofaktoren

Für die Balance von agiler und plan-basierter Entwicklung in den einzelnen Aktivitäten muss bestimmt werden woraus, das Risiko von zu wenig und zu viel Stabilität in den einzelnen Aktivitäten besteht. Dafür werden die Daten aus der Literatur- und Interviewstudie (Kapitel 4 und 5) verwendet. In diesen Daten werden Vor- und Nachteile genannt, die agiles oder plan-basiertes Vorgehen haben. An dieser Stelle werden Ziele, die die Firmen mit der Anwendung von agiler und plan-basierter Entwicklung haben, ebenfalls als Vorteile betrachtet. Diese Vor- und Nachteile aus den Daten, werden dafür verwendet, um die Risiken für zu wenig und zu viel Stabilität abzuleiten. In Appendix C sind Aussagen und Zitate aus der Literatur- und Interviewstudie zu den jeweiligen Risiken aufgelistet.

**Ableitung von Risiken aus Nachteilen** Ein Nachteil stellt oftmals ein konkretes Problem und somit auch Schaden für ein Projekt dar. Auf diese Weise kann von einem Nachteil direkt auf ein Risiko geschlossen werden, das für Projektteams existiert. Zum Beispiel ist der Nachteil von zu detaillierter Langzeitplanung, dass ein Projekt schwer auf Änderungen reagieren kann, da in solchen Fällen ganze Pläne überarbeitet werden müssen. Das Risiko von einer zu detaillierten Langzeitplanung ist demnach, dass ein Projekt nicht auf Änderungen reagieren kann.

**Ableitung von Risiken aus Vorteilen und Zielen** In den Daten der Literatur- und Interviewstudie finden sich Ziele, die Firmen jeweils mit einem agilen oder plan-basierten Vorgehen haben. Dies bedeutet, dass die Firmen einen Vorteil in dem agilen oder plan-basierten Vorgehen sehen. Die Daten enthalten zudem die Nennungen von direkten Vorteilen. Zum Beispiel hat Langzeitplanung den Vorteil, dass Abhängigkeiten berücksichtigt und Koordinationsprobleme vermieden werden können.

Für die Ableitung von Risiken aus Vorteilen müssen nach Boruszewski [107] zwei Fälle unterschieden werden. Im ersten Fall entsteht durch einen Verzicht auf einen Vorteil ein tatsächlicher Schaden an einem Projekt. Wenn ein Projekt zum Beispiel die Langzeitplanung nicht nutzt, entgeht ihnen der Vorteil, dass sie Abhängigkeiten planen können und einen Überblick über diese behalten. Dies kann dazu führen, dass Koordinationsprobleme auftreten und ein

tatsächlicher Schaden durch Verzögerungen auftreten. Das Risiko, das in diesem Fall aus dem Vorteil abgeleitet wird ist, dass Koordinationsprobleme entstehen können.

Im zweiten Fall entsteht durch einen Verzicht auf einen Vorteil kein tatsächlicher Schaden in einem Projekt. Die Nutzung eines Vorteils würde für ein Projekt aber eine zusätzliche Ersparnis oder eine Erleichterung bedeuten. Zum Beispiel unterstützt die Verwendung einer Roadmap die Darstellung der Vision eines Projektes. Möglicherweise reicht die Verwendung von Epics schon aus, um die Vision und das Verständnis des Projektes zu transportieren. Eine zusätzliche Roadmap ermöglicht jedoch einen zusätzlichen Überblick während der Laufzeit des Projektes. Daraus ergibt sich eine zusätzliche Erleichterung. Aus diesem entgangenen Vorteil kann das Risiko abgleitet werden, dass das Projektteam eine Unsicherheit über das Projektziel hat.

## 7.2 Requirements Engineering

Die erste Balance, die betrachtet wird, ist in der Requirements Engineering Aktivität. Für eine Balance muss ein Projektteam entscheiden wie, viel Anforderungsanalyse es am Anfang betreibt und wann es diese abbricht und zu einem kontinuierlichem Requirements Engineering wechselt. Bevor für die Requirements Engineering Aktivität die Risiken betrachtet werden, die für die Balance wichtig sind, wird der Strukturgrad für diese Aktivität beschrieben.

### 7.2.1 Der Strukturgrad im Requirements Engineering

Projektteams nutzen eine Anforderungsanalyse zu Beginn eines Projektes, um die Anforderungen der Kunden und Stakeholder und deren Umsetzung zu verstehen. Dafür sammeln die Projektteams die Anforderungen der Stakeholder ein. Das reine Aufnehmen von Anforderungen verschafft noch kein Verständnis, da diese unterschiedlich verstanden werden können oder es Widersprüche in den Anforderungen geben kann [109]. Daher gehört zum Verständnis auch die Analyse der Anforderungen. Diese besteht darin, dass das Projektteam gemeinsam die Anforderungen interpretiert und Widersprüche zusammen mit den Stakeholdern auflöst. Die Analyse beinhaltet zudem, dass das Projektteam die Umsetzbarkeit der Anforderungen untersucht und verschiedene Möglichkeiten der Umsetzung in Betracht zieht. Im weiteren Verlauf dieses Abschnitts wird der Begriff *Anforderungsanalyse* verwendet. Dieser Begriff fasst in dieser Dissertation sowohl das Aufnehmen als auch das Verstehen von Anforderungen zusammen. Je mehr Anforderungsanalyse durchgeführt wird, desto mehr Anforderungen werden aufgenommen und analysiert.

Die Projektteams wollen ein Verständnis der Anforderungen zu Beginn gewinnen, um die

Entwicklung planen zu können und Probleme frühzeitig zu erkennen. Durch die Anforderungsanalyse erfährt jedes Projektmitglied die Richtung, in die sich das Projekt bewegen wird. Es wird also durch die Anforderungsanalyse zu Beginn eines Projektes gefährliche Unsicherheit vermieden [110]. Das Verständnis der Anforderungen sorgt somit für eine inhaltliche Struktur im Projekt, an der sich das Projektteam orientieren kann. Je detaillierter und vollständiger ein Projektteam die Anforderungen zu Beginn festlegt, desto mehr Struktur wird erzeugt, da immer weniger Unsicherheit herrscht. Ein Projektteam, das zum Beispiel nur eine grobe Vision für ihr Projekt festlegt, schafft wenig Struktur. Ein Projektteam hingegen, das sehr detailliert die Anforderungen aller Features, deren Funktionen und sogar technische Anforderungen für die Umsetzung festlegt, schafft sehr viel Struktur.

Der Strukturgrad in der Requirements Aktivität besteht daher aus dem Umfang der analysierten und festgelegten Anforderungen. Der allgemeine Strukturgrad ist auf einer Skala von 0 bis 100 definiert. Dies wird auf den Strukturgrad im Requirements Engineering übertragen. Bei einem Wert von 0 analysiert ein Projektteam keine der Anforderungen zu Beginn des Projektes und legt diese fest. Bei einem Wert von 100 werden alle Anforderungen analysiert und festgelegt. Der Strukturgrad im Requirements Engineering wird als Variable $a \in [0, 100]$ definiert, die den Anteil der analysierten und festgelegten Anforderungen repräsentiert.

Zwischen den beiden beschriebenen Extremwerten gibt es weitere Abstufungen. Rupp [109] hat für die Anforderungsanalyse fünf Stufen definiert, die Projekte als Abstraktionsstufen bei der Anforderungsanalyse nutzen sollten. Sie schreibt, dass in einem idealen Projekt diese Abstraktionslevel bei der Analyse der Anforderungen durchlaufen werden sollten. Die Anforderungen auf einem niedrigen Abstraktionslevel verfeinern die Anforderungen von einer höheren Stufe. Ein Projektteam fängt mit einer hohen Abstraktion der Anforderungen an und verfeinert diese immer weiter. Die Stufen, die durchlaufen werden, sind die Folgenden:

**Stufe 1:** Auf dieser Stufe analysiert ein Projektteam die Vision, die Systembeschreibung und das Geschäftsziel eines Projektes und legt diese fest.

**Stufe 2:** Auf dieser Stufe identifiziert ein Projektteam die Anwendungsfälle der Stakeholder und legt diese fest.

**Stufe 3:** Auf dieser Stufe analysiert ein Projektteam die Anforderungen an die Funktionen der einzelnen Anwendungsfälle und legt diese fest.

**Stufe 4:** Auf dieser Stufe analysiert ein Projektteam die Systemanforderungen und legt diese fest. Dafür werden die Anwenderanforderungen in Nutzeraktionen und Systemanforderungen aufgeteilt. Eine Nutzeraktion ist die Aktion, die ein Nutzer ausführt, um eine Sys-

temfunktion zu nutzen. Eine Systemanforderung beschreibt die Tätigkeiten des Systems, die durch eine Nutzeraktion ausgelöst werden.

**Stufe 5:** Auf dieser Stufe analysiert ein Projektteam die technischen Anforderungen und legt diese fest. Diese ergeben sich aus den Anforderungen, die auf Stufe 4 festgelegt werden.

Mehr Anforderungsanalyse sorgt für einen höheren Strukturgrad und damit einer höheren Stabilität im Projekt. Abgeleitet aus den beiden allgemeinen Risiken von zu wenig und zu viel Struktur, gibt es daher für die Requirements Engineering das Risiko von zu wenig und zu viel Anforderungsanalyse. Im Folgenden werden die Risikofaktoren beschrieben, aus denen diese beide Risiken bestehen. Zusätzlich werden die Kontextfaktoren beschrieben, die die Kritikalität des jeweiligen Risikos beeinflussen. Die Liste der Risikofaktoren mit den dazugehörigen Kontextfaktoren ist in Tabelle 7.1 dargestellt.

**Tabelle 7.1:** Liste der Risikofaktoren von zu wenig und zu viel Anforderungsanalyse und der dazugehörigen Kontextfaktoren

| **Requirements Engineering** | | |
|---|---|---|
| **Risiskofaktoren für zu wenig Anforderungsanalyse** | | **Kontextfaktoren** |
| R1.1 | Unsicherheit über die Projektrichtung und Abhängigkeiten | Anzahl der Stakeholder |
| | | Verfügbarkeit der Stakeholder |
| | | Verteilung der Entwicklung |
| | | Größe des Projektteams |
| | | Art der Abhängigkeiten |
| R1.2 | Projektteam gibt falsche Schätzung ab | Bedürfnis nach fixierten Ressourcen |
| | | Existenz von Deadlines |
| R1.3 | Übersehen von kritischen nicht-funktionalen Anforderungen | Anzahl von kritischen nicht-funktionalen Anforderungen |
| **Risikofaktoren für zu viel Anforderungsanalyse** | | **Kontextfaktoren** |
| R1.4 | Aufnehmen von falschen Anforderungen | Bekanntheit der Anforderungen |

### 7.2.2 Die Risikofaktoren von zu wenig Anforderungsanalyse

In diesem Abschnitt werden die Risikofaktoren für das Risiko von zu wenig Anforderungsanalyse beschrieben.

**R1.1: Unsicherheit über die Projektrichtung und Abhängigkeiten** Dieses Risiko entsteht, wenn das Projektteam zu wenig Anforderungsanalyse durchführt und so kein gemeinsames Verständnis über den Inhalt und das Ziel (Vision) des Projektes erreicht. Ohne dieses gemeinsame Verständnis hat das Projektteam keine Vorstellung darüber, in welche Richtung sich das

Projekt bewegt, wie die einzelnen Anforderungen zusammenhängen und welche möglichen Abhängigkeiten auch zu anderen Projekten existieren.

Auf diese Weise haben die Entwicklerteams keinen Überblick über das Projekt und es kann zu Koordinationsschwierigkeiten kommen. Durch eine fehlende gemeinsame Ausrichtung entwickeln die Projektteams möglicherweise in unterschiedliche Richtungen. Dadurch kann es passieren, dass Komponenten nicht zusammenpassen oder Teams aufeinander warten müssen. Dies kann zu Verzögerungen im Projekt führen, die weitere Kosten verursachen.

Wenn unter den Stakeholdern kein gemeinsames Verständnis über den Projektinhalt hergestellt wird, haben diese möglicherweise unterschiedliche Vorstellungen darüber, in welche Richtung das Projekt geht. Wenn die Erwartungen von Stakeholdern nicht erfüllt werden, kann dies zu einer Unzufriedenheit der Stakeholder und späten Änderungen im Projekt führen. Dies kostet oft viel Zeit und Geld.

**Kontextfaktoren:** Die Kritikalität dieses Risikos wird von den Stakeholdern und der Größe und Verteilung des Projektteams beeinflusst. Mit einer hohen Anzahl an Stakeholdern und einer geringen Verfügbarkeit ist es schwierig, die Gruppe der Stakeholder während der Projektlaufzeit regelmäßig zusammenzubringen, um die Anforderungen zu besprechen. Ohne ein gemeinsames Verständnis unter den Stakeholdern zu Beginn des Projektes ist daher die Wahrscheinlichkeit höher, dass während der Projektlaufzeit Missverständnisse auftreten und möglicherweise falsche Features entwickelt werden. Aus diesen Gründen steigert eine große Menge an Stakeholdern und eine schlechte Verfügbarkeit die Kritikalität dieses Risikos.

Bei einem großen Projektteam und/oder einer sehr verteilten Entwicklung, zum Beispiel über verschiedene Länder, ist es ebenfalls von Vorteil, mit einem gemeinsamen Verständnis und Ziel in ein Projekt zu starten. Nach dem Start des Projektes teilt sich das Projektteam in unterschiedliche Entwicklerteams, die möglicherweise getrennt voneinander arbeiten. Auch wenn die einzelnen Entwicklerteams teilweise eigenständige Entscheidungen treffen, sind sie trotzdem über ein gemeinsames Verständnis verbunden. Dieses gemeinsame Verständnis dient als Richtlinie für das Treffen von Entscheidungen. Aus diesen Gründen steigert ein großes und/oder verteiltes Projektteam die Kritikalität dieses Risikos.

Wenn ein Projekt Abhängigkeiten zu anderen Projekten hat, müssen diese zu Beginn identifiziert und untersucht werden, damit es während der Entwicklung nicht zu Koordinationsschwierigkeiten kommt. Solche Abhängigkeiten steigern somit die Kritikalität dieses Risikos.

**R1.2: Projektteam gibt eine falsche Schätzung ab** Dieses Risiko entsteht, wenn die Anforderungsanalyse auf einer zu niedrigen Stufe des Strukturgrades verbleibt. Dies kann dazu führen, dass die Umsetzung von Anforderungen zu Beginn eines Projektes nicht ausrei-

chend untersucht und somit verstanden wird. Die Gefahr, ist, dass das Projektteam eine falsche Schätzung für ein Projekt abgibt, da unvorhergesehene Probleme auftreten können. Unvorhergesehene Probleme können zu Verzögerungen im Projekt führen und so zu einer Abweichung zwischen der tatsächlichen und geschätzten Laufzeit des Projektes. Diese Abweichung führt nicht nur zu unvorhergesehenen Kosten, sondern auch zu einer Unzufriedenheit des Kunden.

**Kontextfaktoren:** Dieses Risiko ist besonders kritisch, wenn die Kunden ein großes Bedürfnis nach einem möglichst kleinen Toleranzbereich bei der Schätzung haben. Sie erwarten eine möglichst genau Schätzung des Budgets und der benötigten Zeit und wollen diese Ressourcen festgelegen. Dann ist es wichtig, dass das Projektteam die Anforderungen zu Beginn ausreichend gut versteht. Eine hohes Bedürfnis nach fixierten Ressourcen steigert daher die Kritikalität dieses Risikos.

**R1.3: Übersehen von kritischen nicht-funktionalen Anforderungen** Dieses Risiko entsteht dadurch, dass sich das Projektteam auf die funktionalen Anforderungen fokussiert. Dabei kann es passieren, dass kritische nicht-funktionale Anforderungen übersehen werden. Kritische nicht-funktionale Anforderungen sind solche Anforderungen, die von Anfang an in einem Projekt berücksichtigt werden müssen, damit sie umgesetzt werden können. Beispiele dafür sind Performance oder Security. Wenn diese Anforderungen nicht berücksichtigt werden, kann dies dazu führen, dass sich die gesamte Anwendung in eine falsche Richtung entwickelt und später überarbeitet werden muss. Dies führt zu einer langen Verzögerung des Projektes und zu mehr Kosten.

**Kontextfaktoren:** Die Kritikalität dieses Risiko wird von der Art der nicht-funktionalen Anforderungen bestimmt, die in einem Projekt vorkommen. Wenn das Projekt keinerlei nicht-funktionale Anforderungen besitzt, die direkt zu Beginn des Projektes berücksichtigt werden müssen, ist dieses Risiko allgemein gering. Im Gegensatz dazu, wenn es viele kritische nicht-funktionale Anforderungen gibt, wie Performance, ist dieses Risiko insgesamt größer.

#### Diskussion der Risk Exposure von zu wenig Anforderungsanalyse

Ohne eine Anforderungsanalyse ist die Eintrittswahrscheinlichkeit der oberen beschriebenen Risiken hoch, da das Projektteam zum Beispiel weder Unsicherheiten beseitigt noch nicht-funktionale Anforderungen untersucht. Die Vision und die Anwendungsfälle stellen die Pfeiler in einem Projekt dar, die die Richtung des Projektes definieren. Die Daten der Interviewstudie zeigen zudem, dass die Analyse der Anwendungsfälle für die Erstellung einer groben Schätzung des Projektes besonders wichtig ist, da die Projektteams darüber die wichtigsten und meisten Informationen erhalten. Zu Beginn der Anforderungsanalyse betrachtet das Projektteam eine

Anwendung als Ganzes, bevor sich die Analyse in die einzelnen Komponenten aufgliedert. Auf dieser Ebene werden auch die besonders kritischen nicht-funktionalen Anforderungen besprochen, die für das ganze Projekt gelten sollen.

Die beschriebene Anforderungserhebung findet auf den Stufen 1 und 2 statt. Damit sinkt die Eintrittswahrscheinlichkeit der oben beschriebenen Risiken auf diesen Stufen bereits stark. Mit den steigenden Stufen kommen weniger relevante Informationen dazu, die eine Ausrichtung des Projektes und die grobe Schätzung verbessern oder die nicht-funktionalen Anforderungen sind nicht mehr kritisch. Zum Beispiel führen Diskussionen über Buttonfarben nicht dazu, dass sich eine grobe Schätzung oder die Vision verbessert. Damit sinkt die Eintrittswahrscheinlichkeit der Risiken mit steigendem Strukturgrad weiter, jedoch weniger stark.

Die gleiche Argumentation gilt auch für den Schaden. Auf den Stufen 1 und 2 geht es um die Ausrichtung des gesamten Projektes und damit die Ausrichtung von allen Entwicklerteams. Wenn Koordinationsprobleme zwischen Entwicklerteams auftreten, wird häufig ein hoher Schaden verursacht, da ganze Teams aufeinander warten müssen. Wenn ganze Anwendungsfälle für eine Schätzung nicht betrachtet werden, ist die potenzielle Abweichung von der Schätzung sehr groß. Wie oben beschrieben, bergen die Anwendungsfälle die meisten Informationen, die für eine grobe Schätzung wichtig sind. Das Übersehen von nicht-funktionalen Anforderungen auf der anfänglichen Stufe hat einen hohen potenziellen Schaden, da dort Anforderungen für das gesamte Projekt betrachtet werden. Wenn Anforderungen auf diesen Stufen übersehen werden, bedeutet dies, dass wahrscheinlich die gesamte Software überarbeitet werden muss. Damit sinkt der potenzielle Schaden auf den Stufen 1 und 2 bereits stark.

Mit steigendem Strukturgrad erhebt das Projektteam immer mehr Anforderungen, die die Entwicklung in den einzelnen Teams betrifft. Dies sind zum Beispiel Anforderungen darüber, wie ein Anwendungsfall im Detail ablaufen soll. Diese Anforderungen betreffen die Anforderungen für die einzelnen Entwicklerteams. Der Schaden ist hier geringer, da Koordinationsprobleme oder Änderungen nur auf Entwicklerteamebene auftreten, nicht aber für das gesamte Projekt. Dadurch, dass die Entwickler in den einzelnen Entwicklerteams eng zusammenarbeiten, können Probleme und Änderungen schnell gelöst und durchgeführt werden. Die Daten der Interviewstudie zeigen zudem, dass es ausreicht, detaillierte Anforderungen im Verlauf des Projektes zu erheben. Damit sinkt der potenzielle Schaden mit steigendem Strukturgrad zwar weiter, jedoch weniger stark.

Aus dieser Diskussion folgt, dass die Risk Exposure von zu wenig Anforderungsanalyse bei einem niedrigen Strukturgrad hoch ist und gleichzeitig stark fällt. Je größer der Strukturgrad wird, desto niedriger wird die Risk Exposure, fällt aber gleichzeitig schwächer. Daher kann für den Verlauf der Risk Exposure von zu wenig Anforderungsanalyse die in Abbildung 6.1

eingezeichnete blaue Kurve angenommen werden.

### 7.2.3 Das Risiko von zu viel Anforderungsanalyse

In diesem Abschnitt wird der Risikofaktor für das Risiko von zu viel Anforderungsanalyse beschrieben.

**R1.4: Aufnehmen von falschen Anforderungen** Dieses Risiko entsteht, wenn die Anforderungen zu Beginn bereits zu sehr im Detail festgelegt werden. Je detaillierter die Anforderungen betrachtet werden, desto unsicherer sind sie. Das liegt zum Beispiel daran, dass Stakeholder die Details von Anforderungen noch nicht genau wissen. Wenn in einem Projekt trotzdem die Anforderungen ins Detail spezifiziert werden, geschieht dies daher unter Unsicherheit und kann zur Festlegung von falschen Anforderungen führen. Festgelegte, falsche Anforderungen in einem Projekt bedeuten im Lauf eines Projektes viel Aufwand. Wenn entdeckt wird, dass falsche Anforderungen aufgenommen wurden, müssen ganze Dokumentationen geändert werden und möglicherweise auch die Software.

**Kontextfaktoren:** Die Kritikalität dieses Risikos wird durch die Bekanntheit der Anforderungen beeinflusst. In einem Projekt, in dem bereits alle Anforderungen zu Beginn bekannt sind, ist das Risiko falsche Anforderungen zu erheben gering. In einem Projekt, in dem die Anforderungen unsicher sind, ist dieses Risiko dagegen generell höher. Die Kritikalität dieses Risikos steigt damit mit einer geringen Bekanntheit der Anforderungen.

#### Diskussion der Risk Exposure für zu viel Anforderungsanalyse

Auf den Stufen 1 und 2 definiert das Projektteam die Vision und die Hauptanwendungsfälle. Diese Anforderungen sind allgemein und auf einem hohen Abstraktionslevel und beschreiben grundlegende Bedürfnisse der Stakeholder. Die Unsicherheit bei diesen allgemeinen Anforderungen ist geringer als bei den genauen detaillierten Funktionen der Anwendungsfälle. Zum Beispiel ist die Wahrscheinlichkeit höher, dass sich die Farbe eines Buttons nochmal ändert, als ein Grundbedürfnis an eine Anwendung.

Kunden und Stakeholder haben zu Beginn eines Projektes eine bessere Vorstellung über ihre Grundbedürfnisse als über detaillierte Abläufe der Software. Daher startet die Anforderungsanalyse meist mit der Bildung einer Vision [109]. Damit wird die Anforderungsanalyse auf den steigenden Stufen immer unsicherer und die Wahrscheinlichkeit, dass falsche Anforderungen aufgenommen werden, steigt immer stärker an.

Der potenzielle Schaden steigt mit den geringen Stufen schwach an. Die Anforderungen auf diesen Stufen sind abstrakt, sodass sie genug Freiheitsgrade bei der Ausgestaltung lassen. In

Fällen von Änderungen dieser Anforderungen können daher Anwendungsfälle einfach ausgetauscht werden, ohne dass viele zusätzliche Anforderungen berücksichtigt werden müssen. Mit einer detaillierteren Festlegung der Anforderungen steigt der Aufwand durch Änderungen, weil mehr Anforderungen in Abhängigkeit geändert werden müssen. Damit steigt der potenzielle Schaden mit steigendem Strukturgrad stark an.

Aus dieser Diskussion folgt, dass die Risk Exposure von zu viel Anforderungsanalyse bei einem niedrigen Strukurgrad niedrig ist und gleichzeitig nur schwach ansteigt. Mit steigendem Strukturgrad wird die Risk Exposure größer und steigt dabei auch immer stärker an. Aus diesem Grund kann für den Verlauf der Risk Expsoure von zu viel Anforderungsanalyse die in Abbildung 6.1 orange eingezeichnete Kurve angenommen werden.

### 7.2.4 Diskussion der Balance im Requirements Engineering

Für die Bestimmung der Balance im Requirements Engineering wird das Gesamtrisiko für diese Aktivität analysiert. Aus der Analyse der Risk Exposure von zu wenig und zu viel Anforderungsanalyse kann für das Gesamtrisiko die in Abbildung 6.1 eingezeichnete schwarze Kurve als Verlauf angenommen werden. Für die Balance von beiden Risiken wird basierend auf dem Konzept der Tiefpunkt des Gesamtrisikos betrachtet.

Abgeleitet aus dem allgemeinen Konzept ist dieser erreicht, wenn in einem Strukturgrad das Risiko von zu wenig Anforderungsanalyse genau so stark sinkt, wie das von zu viel Anforderungsanalyse steigt. Konkreter bedeutet dies, dass während die Firmen die Risikofaktoren von zu wenig Anforderungsanalyse minimieren, sie darauf achten müssen, dass sie nicht gleichzeitig zu viele Freiheitsgrade in den Anforderungen einschränken.

Am Anfang des Projektes dominiert das Risiko von zu wenig Anforderungsanalyse. Ein Projektteam sollte sich zu Beginn des Projektes auf die Verringerung dieses Risikos fokussieren. Daher ist es zu Beginn wichtig, ein gemeinsames Verständnis (Anforderungen und Umsetzung) herzustellen und sicherzustellen, dass die wichtigen kritischen, nicht-funktionalen Anforderungen und Abhängigkeiten erhoben werden. Die Interviewstudie (Kapitel 5) zeigt, dass ein gemeinsames Verständnis meist erreicht ist, wenn das Projektteam die bekannten Anwendungsfälle identifiziert und deren groben Funktionen analysiert hat und es eine grobe Schätzung für das Projekt abgeben kann (Stufe 2 und 3). Eine grobe Analyse der Funktionen bedeutet eine Analyse von Anwendungsfällen auf der Ebene von Epics. Dabei werden die detaillierten Nutzer- und Systemschritte der Anwendungsfälle nicht definiert (Stufe 4).

Wenn detailliertere Anforderungen erhoben werden, steigt die Gefahr falsche Anforderungen aufzunehmen. Wenn das Projektteam merkt, dass keine Informationen mehr dazukommen oder Details besprochen werden, die die Schätzung des Projektes nicht mehr verbessern, soll-

te die Anforderungsanalyse zu Beginn des Projektes abgebrochen werden. Diese Details sind zum Beispiel die genauen Nutzer- und Systemschritte in den Anwendungsfällen. Diese können auch noch während der Laufzeit des Projektes definiert und ausformuliert werden. Ein weiterer Indikator für das Erreichen eines gemeinsamen Verständnisses ist, wenn die Projektteilnehmer das Gefühl haben, genug Informationen zu haben, um mit dem Projekt starten zu können.

Der beschriebene Stand an erhobenen Anforderungen ermöglicht dem Projektteam, dass es ein ausreichendes Verständnis für das Projekt gewinnt. Gleichzeitig lässt der Fokus auf die Identifikation der Anwendungsfälle genug Freiraum bei der Ausgestaltung der Anforderungen, da die Nutzer- und Systemschritte im Laufe des Projektes definiert werden. Wenn sich ein Projektteam in Details verliert, zieht sich die Anforderungsanalyse unnötig in die Länge, da auf Grund der Unsicherheit Anforderungen immer wieder neu besprochen werden. Firmen wollen eine möglichst kurze Anforderungsanalyse, um Ressourcen zu schonen. Mit dem Abbruch der Aktivität zu Beginn des Projektes nach dem beschriebenen Stand an erhobenen Anforderungen stellt das Projektteam sicher, nicht den Überblick über die Anforderungen zu verlieren und die Anforderungsanalyse nicht unnötig zu verlängern. Aus der Betrachtung der Balance werden drei folgende Prinzipien abgeleitet:

Prinzip 1: Ein Projektteam sollte sich zu Beginn eines Projektes darauf konzentrieren, ein gemeinsames Verständnis zu erreichen, das ausreicht, um eine grobe Schätzung des Projektes durchzuführen. Hierfür sind die Analyse der Anwendungsfälle, die Identifizierung von Abhängigkeiten und die Analyse der kritischen nicht-funktionalen Anforderungen besonders wichtig. Das Projektteam sollte die Vision für das Projekt als Roadmap darstellen.

Prinzip 2: Ein Projektteam erkennt das Erreichen eines gemeinsamen Verständnisses daran, dass das Projektteam in der Lage ist, eine grobe Schätzung für das Projekt abzugeben und zuversichtlich ist, das Projekt mit den vorhandenen Informationen durchführen zu können.

Prinzip 3: Wenn keine Informationen mehr dazukommen, die zu der Erfüllung von Prinzip 1 oder 2 beitragen, sollte das Projektteam die Anforderungsanalyseanalyse abbrechen und in ein kontinuierliche Anforderungsanalyse übergehen. Dafür sollte das Projektteam die Nutzer- und Systemschritte von Anwendungsfällen möglichst erst während der Laufzeit des Projektes definieren.

## 7.3 Architekturdefinition

Die zweite Balance findet in der Architekturaktivität statt. Für eine Balance muss das Projektteam entscheiden, wie viel der Architektur sie zu Beginn eines Projektes definieren und wann sie die Aktivität zu Beginn des Projektes abbrechen und zu einer kontinuierlichen Weiterentwicklung der Architektur übergehen. Bevor diskutiert wird, woraus das Risiko von zu wenig und zu viel Stabilität in dieser Aktivität besteht, wird zuerst der Strukturgrad für diese Aktivität beschrieben.

### 7.3.1 Der Strukturgrad in der Architekturdefinition

Während der Architekturdefinition zu Beginn eines Projektes wird eine technische Struktur für das Projekt geschaffen. Die technische Struktur entsteht durch die Definition der einzelnen Komponenten, Schichten und Schnittstellen in der Architektur der Software. Diese legt die Richtung fest, in die sich die Architektur während der Entwicklung weiterentwickelt und sorgt so für Stabilität für die Entwicklerteams. Die beschriebene Struktur stellt eine abstrakte Definition der Architektur dar. Mit der Ausarbeitung ganzer Klassendiagramme und ähnlichem wird die Architektur immer vollständiger definiert. Je detaillierter und damit vollständiger die Architektur definiert wird, desto mehr steigert sich die Struktur. Für den Strukturgrad im Bereich der Architektur wird die Variable $d \in [0, 100]$ definiert, die den Definitionsgrad der Architektur angibt. Bei einem Wert von 0 werden keine Teile der Architektur zu Beginn eines Projektes definiert und bei einem Wert von 100 wird die Architektur vollständig definiert.

Zwischen diesen beiden Extremwerten gibt es weitere Abstufungen. Waterman et al. [76] haben drei Stufen in der Definition der Architektur identifiziert:

**Entstehende Architektur (Stufe 1):** Auf dieser Stufe trifft ein Projektteam nur die minimalen Architekturentscheidungen. Diese sind zum Beispiel die Liste der verwendeten Technologiedienste und die Wahl von Architekturpattern auf der höchsten Ebene der Software. Auf dieser Ebene betrachtet ein Projektteam nur die Anforderungen, die unmittelbar umgesetzt werden müssen. Andere Anforderungen, auch solche, die langfristig umgesetzt werden müssen, beachtet das Projektteam nicht.

**Adressierung von Risiken (Stufe 2):** Auf dieser Stufe wird die Architektur soweit definiert, um das Risiko von kritischen nicht-funktionalen Anforderungen ausreichend zu minimieren. Dafür wird die Architektur in einem ausreichenden Umfang und Detail definiert, sodass das Projektteam zuversichtlich ist die nicht-funktionalen Anforderungen umzuset-

zen. Architekturentscheidungen, die weniger Risiko enthalten, werden gleichzeitig zurückgehalten.

**Vollständige Architekturdefinition (Stufe 3):** Auf dieser Stufe wird die Architektur vollständig zu Beginn definiert und es werden keine Architekturentscheidungen zurückgehalten, wie in den beiden vorherigen Stufen. Diese Stufe entspricht einem Strukturgrad von 100 im Bereich der Architektur.

Mit mehr Entscheidungen während der Definition der Architektur zu Beginn eines Projektes schafft ein Projekt mehr Struktur. Abgeleitet aus den beiden allgemeinen Risiken von zu viel und zu wenig Struktur, gibt es daher für die Architektur das Risiko von zu wenig und zu viel Architekturdesign zu Beginn des Projektes. Im Folgenden werden die Risikofaktoren beschrieben, aus denen die beiden Risiken bestehen. Zusätzlich werden die Kontextfaktoren beschrieben, die die Kritikalität der Risiken beeinflussen. Die Liste der Risikofaktoren mit den dazugehörigen Kontextfaktoren ist in Tabelle 7.2 dargestellt.

**Tabelle 7.2:** Die Liste der Risikofaktoren für zu wenig und zu viel Architekturdefinition und der Kontextfaktoren

| **Architektur** | | |
|---|---|---|
| **Risikofaktoren für zu wenig Architekturdefinition** | | **Kontextfaktoren** |
| R2.1 | Architektur bildet sich zufällig | Größe des Projektteams |
| | | Nutzung von Anwendungsframeworks |
| | | Verteilung der Entwicklung |
| R2.2 | Auftreten von Koordinationsschwierigkeiten | Größe des Projektteams |
| | | Verteilung der Entwicklung |
| | | Art der Abhängigkeiten |
| R2.3 | Nichterfüllung kritischer nicht-funktionaler Anforderungen | Anzahl von kritischen nicht-funktionalen Anforderungen |
| **Risikofaktoren für zu viel Architekturdefinition** | | **Kontextfaktoren** |
| R2.4 | Überarbeitung der Architektur | Bekanntheit der Anforderungen |
| | | Bekanntheit der Technologie |

### 7.3.2 Das Risiko von zu wenig Architekturdefinition

In diesem Abschnitt werden die Risikofaktoren des Risikos von zu wenig Architekturdefinition beschrieben.

**R2.1: Architektur bildet sich zufällig** Dieses Risiko entsteht dadurch, dass zu Beginn nicht die Architekturpattern und der Aufbau der Architektur definiert werden. Dadurch haben die Entwicklerteams während der Entwicklung keine Strukturen, die vorgeben, wie sich die Architektur

während der Entwicklung der Software weiterentwickeln soll. Dies bedeutet, dass die Architektur nach Bedarf erweitert wird, ohne dabei einem Plan zu folgen. Dieses Vorgehen kann jedoch dazu führen, dass sich die Architektur zufällig entwickelt und möglicherweise schlecht wartbar ist, was wiederum zusätzliche Kosten und Unzufriedenheit bei den Kunden bedeuten kann.

**Kontextfaktoren:** Die Kritikalität dieses Risikos wird zum einen von der Größe und der Verteilung des Projektteams beeinflusst. Zum anderen davon, ob für die Entwicklung existierende Anwendungsframeworks, wie zum Beispiel Django[1], verwendet werden können. Bei großen oder verteilten Projektteams wird die Architektur in den einzelnen Entwicklungsteams während eines Projektes weiterentwickelt. Für die Koordination der Entwicklerteams bei der Weiterentwicklung der Architektur ist es von Vorteil, wenn die Entwicklerteams Architekturpatterns haben, an denen sie sich orientieren können. Kleinen Projektteams können sich leichter Koordinieren. Die Kritikalität dieses Risikos steigt demnach mit der Größe und Verteilung des Projektteams.

Wenn in einem Projekt vorgefertigte Anwendungsframeworks eingesetzt werden, sind bereits viele Architekturentscheidungen und Patterns durch das Anwendungsframework vorgegeben und die Komplexität der Software ist reduziert [76]. Damit ist dieses Risiko besonders kritisch, wenn keine Anwendungsframeworks verwendet werden können und die Architektur vom Projektteam definiert werden muss.

**R2.2: Auftreten von Koordinationsschwierigkeiten** Dieses Risiko besteht, wenn zu Beginn nicht die einzelnen Komponenten und deren Schnittstellen in der Architektur definiert werden. Ohne diese Definition wissen die Entwicklerteams nicht, wie die einzelnen Komponenten zusammenhängen und mit welchen anderen Entwicklerteams sie während der Entwicklung kommunizieren müssen. Dieser fehlende Überblick erschwert die allgemeine Koordination des Projektes und kann zu Verzögerungen führen, da möglicherweise Teams aufeinander warten müssen.

**Kontextfaktoren:** Dieses Risiko ist besonders kritisch in großen und/oder verteilten Projektteams. Die Definition der Komponenten und der Schnittstellen unterstützt besonders dabei, dass das Projektteam einen Überblick über die Software behält und sich einzelne Entwicklerteams darüber besser koordinieren können [84,89,91]. In kleinen Projektteams ist die Koordination weniger abhängig von der Architektur, da sich die Entwickler leichter austauschen können. Die Kritikalität dieses Risikos steigt daher mit der Größe und Verteilung des Projektteams.

Abhängigkeiten zu anderen Projekten und Systemen erhöhen die Koordinationsschwierig-

[1] www.djangoproject.com

keiten zwischen Projekten. Diese Abhängigkeiten müssen sorgfältig in der Architektur berücksichtigt werden, damit Projektteams nicht aneinander vorbei arbeiten und verschiedene Systeme später nicht integriert werden können. Diese Arten von Abhängigkeiten steigern somit die Kritikalität dieses Risikos

**R2.3: Nichterfüllung kritischer nicht-funktionaler Anforderungen** Kritische nicht-funktionale Anforderungen gelten oftmals für die gesamte Software und müssen direkt zu Beginn berücksichtigt werden. Dieses Risiko entsteht daher dadurch, dass diese Anforderungen zu Beginn eines Projektes bei der Architekturdefinition nicht berücksichtigt werden. Das bedeutet, dass die notwendigen Entscheidungen für die Erfüllung dieser Anforderungen nicht getroffen werden und die Software diese Anforderungen nicht erfüllt. Das kann zu einer Unzufriedenheit des Kunden und möglicherweise zu aufwändigen Änderungen in der Software führen. Die Auslieferung von Software verzögert sich dadurch und kostet mehr Geld.

**Kontextfaktoren:** Die Kritikaltiät dieses Risikos hängt von der Art der nicht-funktionalen Anforderungen ab. Dieses Risiko ist besonders kritisch, wenn es in einem Projekt viele Anforderungen gibt, die direkt zu Beginn des Projektes implementiert werden müssen. In einem Projekt ohne solche nicht-funktionalen Anforderungen ist auch das Risiko generell geringer.

#### Diskussion der Risk Exposure von zu wenig Architekturdefinition

Ohne irgendeine anfängliche Definition der Architektur ist die Eintrittswahrscheinlichkeit der oben beschriebenen Risiken sehr groß, da das Projektteam für die Architektur keinerlei Richtung definiert und keine nicht-funktionalen Anforderungen berücksichtigt. Besonders Entscheidungen über die Architekturpattern und Schnittstellen auf den niedrigen Stufen des Strukturgrades unterstützen dabei, dass eine Vorgabe für Weiterentwicklung der Architektur während der Entwicklung existiert und dass Koordinationsprobleme verhindert werden. Durch die Architekturpattern und die Schnittstellen bekommt das Projektteam einen Überblick über die Architektur. Zudem liegt der Fokus zu Beginn bei der Architekturdefinition auf den Anforderungen, die für die gesamte Software gelten und unmittelbar umgesetzt werden müssen [39]. Dazu gehören insbesondere die kritischen nicht-funktionalen Anforderungen. Damit sinkt die Eintrittswahrscheinlichkeit der oben beschriebenen Risiken bereits stark auf den niedrigen Stufen des Strukturgrades.

Mit steigendem Strukturgrad werden immer mehr Architekturdetails für die Komponenten festgelegt oder Entscheidungen getroffen, die auch später getroffen werden können. Zum Beispiel gibt es auch nicht-funktionale Anforderungen, die auch später umgesetzt werden können. Die Festlegung der Architektur in den Komponenten trägt weniger dazu bei, dass sich das

gesamte Projektteam über eine gemeinsame Architektur koordinieren kann. Dies liegt daran, dass die einzelnen Komponenten mehr die Arbeit in den einzelnen Entwicklerteams betrifft. Das heißt, dass mit steigendem Strukturgrad die Eintrittswahrscheinlichkeit der Risiken weiter sinkt, jedoch weniger stark.

Die gleiche Argumentation gilt auch für den potenziellen Schaden. Zu Beginn der Architekturdefinition werden die Architekturentscheidungen hauptsächlich auf Projektebene getroffen. Wenn auf dieser Ebene die nötigen Entscheidungen nicht getroffen werden, ist der potenzielle Schaden sehr hoch, da zum Beispiel die kritischen nicht-funktionalen Anforderungen nicht berücksichtigt werden oder es Koordinationsprobleme zwischen einzelnen Entwicklerteams gibt. Diese Probleme verursachen einen hohen Schaden, da möglicherweise die gesamte Software umgebaut werden muss oder ganze Entwicklerteams aufeinander warten. Die Festlegung von Schnittstellen und der Architekturpattern unterstützt dabei, dass das Projektteam einen gesamten Überblick über die Architektur hat. Die niedrigen Stufen des Strukturgrades unterstützten daher dabei, den potenziellen Schaden stark zu senken.

Die obige Diskussion für die Eintrittswahrscheinlichkeit gilt auch für den Schaden. Immer detailliertere Architekturentscheidungen tragen weniger dazu bei, dass der potenzielle Schaden sinkt. Dies liegt daran, dass immer mehr Entscheidungen zu den einzelnen Komponenten getroffen werden. Wenn diese Entscheidungen geändert werden müssen, muss nicht mehr die gesamte Software umgebaut werden, sondern nur noch einzelne Teile. Diese Entscheidungen tragen daher weniger dazu bei, dass der potenzielle Schaden gesenkt wird. Mit einem steigenden Strukturgrad sinkt daher der potenzielle Schaden zwar weiter, jedoch weniger stark.

Aus der vorangegangenen Diskussion folgt, dass die Risk Exposure von zu wenig Architekturdefinition mit einem geringen Strukturgrad hoch ist und gleichzeitig stark sinkt. Je größer der Strukturgrad wird, desto weniger stark sinkt die Risk Exposure. Daher kann für den Verlauf der Risk Exposure von zu wenig Architekturdefinition die in Abbildung 6.1 eingezeichnete blaue Kurve angenommen werden.

### 7.3.3 Das Risiko von zu viel Architekturdefinition

In diesem Abschnitt wird der Risikofaktor für das Risiko von zu viel Architekturdefinition beschrieben.

**R2.4: Überarbeitung der Architektur** Dieses Risiko entsteht, wenn die Architektur bereits zu Beginn zu detailliert definiert wird. Dadurch werden keine Freiräume für die Entwicklerteams bei der Weiterentwicklung der Architektur gelassen. Dies erschwert den Umgang mit unsicheren Anforderungen, die sich im Laufe des Projektes möglicherweise nochmal ändern. Zudem ent-

steht dieses Risiko, wenn bereits Entscheidungen zur Umsetzung getroffen werden, während die Umsetzung noch unsicher ist. So kann es passieren, dass im Laufe des Projektes Teile der Software wieder umgestaltet werden müssen, da die Umsetzung einen anderen Weg nehmen muss.

Eine zu detaillierte Architektur kann daher zu Überarbeitungen der Software führen. Dies bedeutet Verzögerungen im Projekt, die zu einer Unzufriedenheit des Kunden und mehr Kosten führen können.

**Kontextfaktoren:** Die Kritikalität dieses Risikos ist von der Bekanntheit der Anforderungen und der Technologie abhängig. In einem Projekt, in dem die Anforderungen bereits vollständig bekannt sind, ist das Risiko, zu viele Freiheitsgrade einzuschränken und damit spätere Änderungen an der Architektur vornehmen zu müssen insgesamt geringer. In einem Projekt hingegen, das unsichere Anforderungen besitzt, ist dieses Risiko insgesamt größer. Dasselbe gilt für die Bekanntheit der Technologie. Wenn ein Projektteam mit der verwendete Technologie sehr vertraut ist, ist das Projektteam nicht darauf angewiesen mit der Entwicklung zu lernen und kann bereits mehr Architekturentscheidungen treffen. In einem Projekt mit einer geringen Bekanntheit der Technologie ist das Gegenteil der Fall. Hier kann es passieren, dass ein Projektteam später Änderungen vornehmen muss, da sie in einem Lernprozess feststellen, dass ein eingeschlagener Weg nicht funktioniert. Damit steigt die Kritikalität dieses Risikos mit einer geringen Bekanntheit der Anforderungen und Technologie.

#### Diskussion der Risk Exposure von zu viel Architekturdefinition

Der niedrige Strukturgrad besteht darin, eine grundlegende Plattform zu definieren, auf der die Entwicklung der weiteren Anforderungen aufsetzen kann. Ausgehend von dieser grundlegenden Plattform ist das Projekt sehr frei darin, in welche Richtung es sich bewegt. Die Wahrscheinlichkeit, dass diese grundlegende Plattform geändert werden muss, ist gering. Die Eintrittswahrscheinlichkeit für dieses Risiko ist daher mit einem geringen Strukturgrad ebenfalls gering und steigt schwach an, da die Architekturentscheidungen viele Freiräume lassen.

Entscheidungen auf einem höheren Strukturgrad stellen gravierendere Einschränkungen der Freiheiten dar, wie zum Beispiel die Festlegung von Klassendiagrammen und Ähnlichem. An dieser Stelle werden bereits detaillierte Anforderungen umgesetzt. Die Wahrscheinlichkeit, dass sich diese ändern oder dass falsche Entscheidungen hinsichtlich der Umsetzung getroffen werden, ist größer. Damit steigt die Wahrscheinlichkeit, dass es zu Änderungen der Software kommt, mit einem höheren Strukturgrad stärker an.

Der Schaden steigt bei einem geringen Strukturgrad ebenfalls schwach an, da genug Entscheidungsfreiheiten gelassen werden. Die Architektur muss daher nicht ständig umgebaut

werden, sondern kann sich mit der Entwicklung zusammen entwickeln. Mit steigendem Strukturgrad sind die Freiheiten der Entwickler mehr eingeschränkt. Um die Architektur in eine andere Richtung zu lenken, müssen daher große Teile der Software umgebaut werden. Dies lässt den potenziellen Schaden mit steigendem Strukturgrad stark ansteigen.

Aus der Diskussion folgt, dass die Risk Exposure von zu viel Architekturdefinition bei einem geringen Strukturgrad niedrig ist und schwach ansteigt. Mit einem steigenden Strukturgrad steigt die Risk Exposure immer stärker an. Aus diesem Grund kann für den Verlauf der Risk Exposure von zu viel Architekturdefinition die in Abbildung 6.1 orange eingezeichnete Kurve angenommen werden.

### 7.3.4 Diskussion der Balance bei der Architektur

Für die Bestimmung der Balance in der Architekturdefinition wird das Gesamtrisiko für diese Aktivität betrachtet. Aus der Analyse der Risk Exposure von zu wenig und zu viel Architekturdefinition kann für das Gesamtrisiko die in Abbildung 6.1 eingezeichnete schwarze Kurve angenommen werden. Für die Balance von beiden Risiken wird basierend auf dem Konzept der Tiefpunkt des Gesamtrisikos betrachtet.

Abgeleitet aus dem allgemeinen Konzept ist dieser erreicht, wenn in einem Strukturgrad das Risiko von zu wenig Architekturdefinition genau so stark sinkt, wie das von zu viel Architekturdefinition steigt. Dies bedeutet, dass während die Firmen die Architektur zu Beginn eines Projektes definieren, um die Risiken zu minimieren, sie gleichzeitig darauf achten müssen, dass sie nicht im gleichen Maße Freiräume einschränken.

Am Anfang der Architekturdefinition dominiert das Risiko von zu wenig Architekturdefinition. Ein Projektteam sollte sich daher zu Beginn darauf konzentrieren, einen Leitfaden für die Entwicklung der Architektur zu schaffen, die Komponenten und Schnittstellen für die Koordination zu definieren und die kritischen nicht-funktionalen Anforderungen zu berücksichtigen. Der Leitfaden entsteht dadurch, dass das Projekt das grundlegende Schema der Architektur festlegt. Zum Beispiel wird dabei festgelegt, ob die Architektur dem Schichtmodell folgen soll.

Der Fokus auf diese Entscheidungen ermöglicht, dass das Projektteam ein Gerüst hat, auf das es sich bei der Weiterentwicklung der Architektur stützen kann. Gleichzeitig sind ausreichend Freiheiten gelassen, sodass Änderungen leicht umgesetzt werden können. Besonders die Beschränkung auf Komponenten und Schnittstellen ermöglicht den Entwicklerteams Freiheiten darin, wie sie die einzelnen Komponenten entwickeln und wie die Architektur hinter den Schnittstellen aussieht. Nach dem Punkt, indem die Risiken von zu wenig Architekturdefinition minimiert sind, dominiert das Risiko von zu viel Architekturdefinition. Wenn das Projektteam merkt, dass die Entscheidungen bei der Architektur die oben genannten Risiken nicht mehr

senkt, sondern im gleichen Maße die Freiheiten einschränkt, sollte die Architekturdefinition zu Beginn des Projektes abgebrochen werden. Die weiteren Architekturentscheidungen sollten dann während der Entwicklung getroffen werden. Ein generelles Ziel des Projektteams ist es, eine möglichst kurze Architekturdefinition zu Beginn des Projektes zu haben, um auch hier Ressourcen zu schonen. Der Abbruch der Aktivität zu Beginn des Projektes, an diesem Punkt, stellt dies sicher. Für die Balance bei der Architekturdefinition werden die folgenden Prinzipien definiert:

Prinzip 4: Ein Projektteam sollte sich zu Beginn des Projektes darauf konzentrieren einen Leitfaden für die Entwicklung der Architektur zu schaffen (grundlegendes Schema für die Architektur), die Komponenten und Schnittstellen für die Koordination zu definieren und die kritischen nicht-funktionalen Anforderungen zu berücksichtigen.

Prinzip 5: Ein Projektteam erkennt das Erreichen von Prinzip 4 daran, wenn es gemeinsam zu dem Entschluss kommt, dass es die Software mit dem Stand von Architekturdefinition erfolgreich entwickeln kann.

Prinzip 6: Wenn Architekturentscheidungen nicht mehr zur Erfüllung des Prinzips 4 beitragen und stattdessen im gleichen Maße die Freiheiten einschränken, sollte die Architekturdefinition zu Beginn des Projektes abgebrochen werden. Die weiteren Architekturentscheidungen sollten während der Entwicklung getroffen werden.

## 7.4 Planung

Die dritte Balance findet in der Planungsaktivität statt. Für eine Balance muss ein Projektteam entscheiden, für welchen Zeitabschnitt und in welchem Detailgrad sie für ein Projekt Langzeitplanung betreiben. Bevor diskutiert wird, woraus das Risiko von zu viel und zu wenig Stabilität in dieser Aktivität besteht, wird zuerst der Strukturgrad für die Planungsaktivität beschrieben.

### 7.4.1 Der Strukturgrad in der Planung

Bei der Planung entsteht Struktur dadurch, dass das Projektteam durch eine zentrale Projektleitung den Ablauf eines Projektes bestimmt und Meilensteine definiert [5]. Auf diese Weise werden die Schritte, die zum Erreichen des Projektzieles nötig sind, in eine zeitliche Reihenfolge gebracht und mit einer benötigten Dauer versehen. Mit Hilfe dieser Planung können sich Entwicklerteams in einem Projekt orientieren. Es sorgt daher für Stabilität in einem Projekt.

Je kleinschrittiger die Schritte definiert werden und je länger der Zeitraum ist, für den diese Planung stattfindet, desto mehr Planungsstruktur wird in einem Projekt erreicht. Dies wird in dieser Arbeit als der Detailgrad von Planung betrachtet. Für den Strukturgrad in der Planung wird die Variable $p \in [0, 100]$ definiert, die den Detailgrad der Planung in einem Projekt angibt. Bei einem Wert von 0 findet überhaupt keine Planung der Entwicklung statt. Bei einem Wert von 100 wird das Projekt sehr kleinschrittig und für die gesamte Dauer geplant.

Zwischen diesen beiden Extremwerten gibt es weitere Abstufungen. Diese bestehen aus den Planungsmodi, die in Kapitel 5 beschreiben werden. Hierbei wird der Iterationsmodus weggelassen, da es sich dabei nicht um Langzeitplanung handelt, sondern um eine kontinuierliche Planung.

**Flexibilitätsmodus (Stufe 1):** Auf dieser Stufe schätzt ein Projektteam die ungefähr benötigte Zeit und das Budget für das Projekt und planen dafür den groben Anforderungsumfang des Projektes. Zusätzlich plant es auf Projektebene die Entwicklungsreihenfolge von Features, ohne dabei konkrete Meilensteine anzugeben.

**Meilensteinmodus (Stufe 2):** Auf dieser Stufe legt ein Projektteam zusätzlich zum groben Verlauf des Projektes Meilensteine für die Entwicklung fest. Mit steigendem Strukturgrad werden Meilensteine immer kleinschrittiger und für einen längeren Zeitraum definiert. Zudem ist die Einhaltung der Meilensteine mit steigendem Strukturgrad strenger.

**Managementmodus (Stufe 3):** Auf dieser Stufe wird bereits für das gesamte Projekt die Dauer und das Budget festgelegt. Daher existieren bereits sehr detaillierte Meilensteine, die nur wenige Freiheiten bieten.

Mit mehr Langzeitplanung werden Meilensteine kleinschrittiger und für einen längeren Zeitraum geplant. Zu dem müssen Meilensteine strenger eingehalten werden. Mehr Langzeitplanung bedeutet mehr Kontrolle über die Entwicklerteams, da das Projekt nicht nur auf Projektebene, sondern auch auf Teamebene von der zentralen Projektleitung geplant wird. Eine zentrale Planung auf Teamebene bedeutet, dass die Arbeit der Entwickler in den einzelnen Sprints (die tägliche Arbeit) zentral von der Projektleitung geplant wird. Dadurch bekommt die zentrale Projektleitung mehr Kontrolle und es erreicht mehr Stabilität für ein Projekt. Abgeleitet aus den beiden allgemeinen Risiken von zu wenig und zu viel Stabilität gibt es daher für die Planung die beiden Risiken von zu wenig und zu viel Langzeitplanung. Im Folgenden werden die Risikofaktoren beschrieben, aus denen die beiden Risiken bestehen. Zusätzlich werden ebenfalls die Kontextfaktoren beschrieben, die die Kritikalität der Risiken beeinflussen. Eine Liste der Risikofaktoren mit den dazughörigen Kontextfaktoren befindet sich in Tabelle 7.3.

**Tabelle 7.3:** Liste der Risikofaktoren für zu wenig und zu viel Langzeitplanung und der Kontextfaktoren

| **Planung** | | |
|---|---|---|
| **Risikofaktoren für zu wenig Langzeitplanung** | | **Kontextfaktoren** |
| R3.1 | Auftreten von Koordinationsproblemen | Größe des Projektteams |
| | | Art der Abhängigkeiten |
| | | Verteilung der Entwicklung |
| R3.2 | Nichteinhaltung der Projektparameter | Bedürfnis nach fixierten Ressourcen |
| | | Existenz von Deadlines |
| **Risikofaktoren für zu viel Langzeitplanung** | | **Kontextfaktoren** |
| R3.3 | Verminderung der Flexibilität | Bekanntheit der Anforderungen |
| | | Bekanntheit der Technologie |
| | | Verfügbarkeit des Projektteams |

### 7.4.2 Das Risiko von zu wenig Langzeitplanung

In diesem Abschnitt werden die Risikofaktoren für das Risiko von zu wenig Langzeitplanung beschrieben.

**R3.1: Auftreten von Koordinationsproblemen** Dieses Risiko entsteht dadurch, dass die Entwicklerteams Abhängigkeiten innerhalb des Projektes nicht berücksichtigen. Das bedeutet, dass sie nicht gemeinsame Meilensteine zwischen Teams oder Projekten definieren. So kann es vorkommen, dass Features in einer falschen Reihenfolge entwickelt werden und Entwicklerteams aufeinander warten müssen. Dies kann zu Verzögerungen im Projekt und so zu zusätzlichen Kosten führen.

**Kontextfaktoren:** Die Kritikalität dieses Risikos ist abhängig von der Größe und Verteilung des Projektteams und der Art an Abhängigkeiten. Die Langzeitplanung unterstützt die Koordination von großen und verteilten Projektteams, indem sie Klarheit über die Entwicklungsarbeit herstellt. Dadurch haben die Entwicklungsteams eine gemeinsame Grundlage, anhand derer sie sich bei der Koordination orientieren können [82]. Langzeitplanung ist besonders hilfreich in großen und/oder verteilten Projektteams, die generell Schwierigkeiten haben, sich zu koordinieren. Damit steigt die Kritikalität dieses Risikos durch die Größe und Verteilung des Projektteams. Besonders Abhängigkeiten zu anderen Projekten müssen beachtet werden und führen bei Missachtung zu Koordinationsproblemen. Deshalb ist dieses Risiko in einem Projekt mit Abhängigkeiten zu anderen Projekten besonders kritisch.

**R3.2: Nichteinhaltung der Projektparameter** Dieses Risiko entsteht, wenn ein Projektteam nicht ausreichend analysiert, welche Schritte sie benötigen, um ein Projektziel zu erreichen und wie lange sie für diese Schritte benötigen. Ohne diese Planung hat das Projektteam keinen An-

haltspunkt darüber, wann sie mit Tätigkeiten beginnen müssen, um rechtzeitig mit dem Projekt fertig zu werden. Zum Beispiel kann es passieren, dass ein Projektteam ein neues Feature zu einem bestimmten Zeitpunkt fertig stellen möchte, jedoch zu spät mit dem Sammeln von Anforderungen beginnt. Der Schaden, der dadurch entsteht ist, dass vereinbarte Zeiten und Budgets nicht eingehalten werden. Dies führt zu einem finanziellen Schaden für die Softwarefirma, da diese meist die Mehrkosten tragen müssen, aber auch zu einer Unzufriedenheit beim Kunden.

**Kontextfaktoren:** Dieses Risiko ist besonders kritisch in Projekten, bei denen die Kunden ein Bedürfnis nach fixierten Budgets haben und es Deadlines gibt. In solchen Fällen ist die Planung besonders wichtig, um sicherzustellen, dass die zu Beginn gemachte Schätzung eingehalten wird. Dafür ist es wichtig, dass die Projektteams die Entwicklung im voraus planen. In Fällen, in denen ein sehr großer Toleranzbereich für die Schätzung genutzt werden kann, ist auch dieses Risiko weniger kritisch.

#### Die Risk Exposure von zu wenig Langzeitplanung

Ohne irgendeine Art von Langzeitplanung ist die Eintrittswahrscheinlichkeit der oben beschriebenen Risiken hoch, fällt jedoch bereits mit einem geringen Strukturgrad stark. Die Daten der Interviewstudie haben gezeigt, dass Projektteams in APH-Ansätzen zusammen mit dem Kunden nur eine grobe Festlegung der Projektparameter durchführen. Diese grobe Schätzung besteht darin, dass es einen Toleranzbereich für die Zeit und das Budget des Projektes gibt. Dadurch muss ein Projektteam nicht im Detail genau planen, wie lange es für einzelne Features benötigt. Stattdessen unterstützt eine Roadmap auf Projektebene bereits dabei, dass das Projektteam plant, wann es mit einzelnen Features beginnen will. Diese grobe Planung unterstützt bereits dabei, dass das Projektteam ihre grobe Schätzung einhalten kann. Zudem unterstützt eine Roadmap dabei, dass Abhängigkeiten zu anderen Projekten berücksichtigt werden. Damit sinkt die Eintrittswahrscheinlichkeit dieses Risikos bereits stark mit einem niedrigen Strukturgrad.

Die Definition von immer detaillierteren Meilensteinen und genau definierten Zeiten verbessert die Einhaltung einer groben Schätzung dagegen weniger. Zum Beispiel trägt die zentrale Planung von einzelnen Tagen in einem Sprint eines Entwicklerteams nicht dazu bei, dass Abhängigkeiten zu anderen Projekten berücksichtigt werden. Dadurch sinkt die Eintrittswahrscheinlichkeit der oben genannten Risiken weiter, jedoch weniger stark.

Die gleiche Argumentation gilt auch für den Schaden. Die grobe Planung trägt bereits dazu bei, dass sich die tatsächliche Laufzeit eines Projektes innerhalb und nicht außerhalb des Toleranzbereichs bewegt, oder dass nicht ganze Teams aufeinander warten müssen. Eine immer detailliertere Planung trägt jedoch weniger dazu bei, dass ein Projektteam die grobe Schätzung

einhält. Eine solche Planung würde auch die tägliche Arbeit in den Sprints der einzelnen Entwicklerteams planen. Eine solche Planung hat einen geringen Einfluss darauf, dass Koordinationsprobleme zwischen Entwicklerteams auftreten und verringert somit den potenziellen Schaden weniger stark. Damit sinkt der potenzielle Schaden mit einem steigenden Strukturgrad schwächer.

Aus der vorangegangenen Diskussion folgt, dass das Risk Exposure von zu wenig Langzeitplanung bei einem niedrigen Strukturgrad hoch ist und gleichzeitig stark sinkt. Mit steigendem Strukturgrad sinkt die Risk Exposure immer schwächer. Aus diesem Grund kann für den Verlauf der Risk Exposure von zu wenig Langzeitplanung die in Abbildung 6.1 eingezeichnete blaue Kurve angenommen werden.

### 7.4.3 Das Risiko von zu viel Langzeitplanung

In diesem Abschnitt wird der Risikofaktor für zu viel Langzeitplanung beschrieben.

**R3.3: Verminderung der Flexibilität** Dieses Risiko entsteht dadurch, dass ein Projektteam eine zu detaillierte Planung macht. Dies bedeutet, dass das Projektteam bei der Langzeitplanung die Schritte in einem Projekt zu kleinschrittig plant und damit zu wenige Freiräume bei der Planung lässt. Mit so einer Planung greift die zentrale Projektleitung in die Arbeit der Entwickler während der Sprints ein. In einem Projekt kann es immer wieder zu Anforderungsänderungen oder zu Erkenntnissen bei der Entwicklung kommen, die dafür sorgen, dass gemachte Pläne geändert werden müssen. Wenn Pläne zu kleinschrittig erstellt werden, bedeutet dies immer viel Aufwand beim Ändern der Pläne und das Projektteam kann nicht schnell auf Änderungen reagieren. Wenn das Projektteam nicht in der Lage ist, schnell auf Änderungswünsche zu reagieren, kann dies zu Verzögerungen und zu einer Unzufriedenheit beim Kunden führen.

**Kontextfaktoren:** Die Kritikalität dieses Risikos wird von der Bekanntheit der Anforderungen, der Technologie und der Ressourcen beeinflusst. In Projekten, in denen die Anforderungen bereits zu Beginn bekannt sind, ist die Wahrscheinlichkeit generell geringer, dass Anforderungsänderungen auftreten und Pläne daraufhin geändert werden müssen. Dies trifft ebenfalls auf die Bekanntheit der Technologie zu. Wenn ein Projektteam bereits oft mit einer Technologie gearbeitet hat, ist es unwahrscheinlicher, dass Probleme während der Entwicklung auftreten und Pläne geändert werden müssen.

Die Kritikalität dieses Risikos ist zudem abhängig von der Verfügbarkeit des Projektteams. Es kann zum Beispiel vorkommen, dass Entwickler ständig von einem Projekt abgezogen werden und nur von Zeit zu Zeit dem Projekt zugeteilt werden. In einem solchen Fall kann ein

Projektteam schlecht auf lange Sicht planen und die Wahrscheinlichkeit ist generell größer, dass gemachte Pläne angepasst werden müssen.

**Die Risk Exposure von zu viel Langzeitplanung**

Bei einem geringen Strukturgrad hat ein Projektteam viele Freiräume bei der Planung. Dies bedeutet, dass zum Beispiel Meilensteine mehrere Wochen auseinanderliegen und das Projektteam innerhalb dieser Wochen flexibel agieren kann. Durch diese Flexibilität ist die Wahrscheinlichkeit, dass Änderungen an einer bestehenden Planung gemacht werden müssen gering. Gleichzeitig lässt dieser geringe Strukturgrad die Eintrittswahrscheinlichkeit damit nur schwach ansteigen. Je detaillierter Langzeitplanung betrieben wird, desto größer wird die Wahrscheinlichkeit, dass geplante Meilensteine geändert werden müssen und sich dadurch die gesamte Planung verschiebt. Dies liegt daran, dass während eines Projektes immer Änderungen oder Probleme auftreten können und eine zu detaillierte Planung dem Projektteam nicht ermöglicht, auf diese flexibel zu reagieren.

Die gleiche Argumentation gilt für den potenziellen Schaden. Bei einem niedrigen Strukturgrad ist der Aufwand, Pläne zu ändern, gering, da viele Freiräume existieren und einzelne Meilensteine ohne Auswirkung auf die Gesamtplanung geändert werden können. Bei einer sehr detaillierten Planung wird der Aufwand im Fall von Änderungen immer gravierender. Das liegt daran, dass bei einem hohen Strukturgrad Änderungen an einer Stelle im Plan Auswirkungen auf die restliche Planung haben. Das bedeutet, dass alle anderen Meilensteine ebenfalls neu geplant werden müssen. Damit steigt der potenzielle Schaden mit einem geringen Strukturgrad schwach an und mit einem steigenden Strukturgrad immer stärker.

Aus der vorangegangenen Diskussion folgt, dass die Risk Exposure von zu viel Langzeitplanung bei einem niedrigen Strukturgrad niedrig ist und schwach ansteigt. Mit einem steigenden Strukturgrad steigt die Risk Exposure immer stärker an. Aus diesem Grund kann für die Risk Exposure von zu viel Langzeitplanung die in Abbildung 6.1 eingezeichnete orange Kurve angenommen werden.

### 7.4.4 Diskussion der Balance in der Planung

Für die Bestimmung der Balance bei der Langzeitplanung wird das Gesamtrisiko für diese Aktivität betrachtet. Aus der Analyse der Risk Exposure von zu wenig und zu viel Langzeitplanung kann für das Gesamtrisiko die in Abbildung 6.1 eingezeichnete schwarze Kurve angenommen werden. Für die Balance von beiden Risiken wird basierend auf dem Konzept der Tiefpunkt des Gesamtrisikos betrachtet.

Abgeleitet aus dem allgemeinen Konzept ist dieser erreicht, wenn in einem Strukturgrad das Risiko von zu wenig Langzeitplanung genau so stark sinkt, wie das von zu viel Langzeitplanung steigt. Dies bedeutet, dass Firmen, während sie den beschriebenen Risiken von Langzeitplanung begegnen, darauf achten müssen, dass sie nicht zur gleichen Zeit die Flexibilität des Projektes zu stark einschränken.

Bei einem niedrigen Strukturgrad in der Planung dominiert das Risiko von zu wenig Langzeitplanung. Ein Projektteam sollte sich daher zuerst darauf konzentrieren, dass das Risiko von Koordinationsschwierigkeiten und das Nichteinhalten von Projektparametern ausreichend gesenkt ist. Die Interviewstudie zeigt, dass dafür in APH-Ansätzen bereits die Definition einer Roadmap mit losen Meilensteinen ausreicht, die den Entwicklungsverlauf von Features zeigt. Abgeleitet von den Risiken sollte die Roadmap die Integrationspunkte in einem Projekt enthalten und eine grobe Planung über die zeitliche Dauer von Aktivitäten zeigen. Dies beinhaltet zum Beispiel eine Planung darüber, wann die Anforderungen für ein neues Feature feststehen müssen, damit es rechtzeitig entwickelt werden kann.

Es sollte jedoch davon Abstand genommen werden, dass die Durchführung der Entwicklungsarbeit in den Sprints durch Meilensteine geplant wird und dadurch zentrale Planung auch auf der Teamebene stattfindet. Dies vermindert die Flexibilität der Entwickler und würde zu vielen Überarbeitungen von Plänen führen. Zudem sollte eine Langzeitplanung nicht für einen längeren Zeitraum durchgeführt werden, als es die Minimierung der Risiken notwendig macht. Beides nimmt der Entwicklung zu viele Freiheiten und macht sie langsam. Wenn ein Projektteam merkt, dass durch eine detailliertere Planung die oben genannten Risiken von zu wenig Langzeitplanung nicht mehr maßgeblich gesenkt und stattdessen die Freiräume immer mehr eingeschränkt werden, sollte Langzeitplanung nicht weiter betrieben werden. Stattdessen sollte das Projektteam für die weitere Planung kontinuierliche Planung verwenden.

Eine kontinuierliche Planung könnte zum Beispiel so aussehen, dass basierend auf einer Roadmap, die Projektleitung immer analysiert, was in der nächsten Iteration entwickelt werden soll und welche Tätigkeiten in der Iteration durchgeführt werden müssen. Dies bedeutet, dass Langzeitplanung nur die globale Steuerung des Projektes übernimmt. Die Entwicklerteams und das Anforderungsteam planen ihre Arbeit jedoch selbstständig ohne eine zentrale Planung Meilensteine. Zum Beispiel planen die Entwicklerteams ihre Arbeit, indem sie die Sprint Backlogs für jeweils die nächsten drei Sprints im Vorfeld planen. Auf diese Weise werden die Risiken von zu wenig Langzeitplanung ausreichend minimiert, jedoch bleibt genug Flexibilität. Aus der Analyse der Balance der Planung werden die folgenden Prinzipien definiert:

Prinzip 7: Ein Projektteam sollte sich bei der Langzeitplanung darauf fokussieren, Abhängigkeiten, die grobe Dauer von Aktivitäten und die Entwicklungsreihenfolge von Features zu

planen.

Prinzip 8: Zur Erreichung von Prinzip 7 reicht in APH-Ansätzen eine Roadmap auf Projektebene aus, die grobe Meilensteine zu Integrationspunkten und Tätigkeiten enthält. Diese sollte sich auf die globale Planung von unterschiedlichen Teams fokussieren. (Entwicklerteams, Anforderungsteam, Testteam, etc.)

Prinzip 9: Die Langzeitplanung sollte nur so weit in die Zukunft betrieben werden, wie es die Erreichung von Prinzip 7 nötig macht. Mit steigender Planung in die Zukunft sollte die Planung als weniger fest und als unsicher betrachtet werden.

Prinzip 10: Die Arbeit in den einzelnen Teams sollte nicht durch eine Langzeitplanung geplant werden. Stattdessen sollten diese ihre eigene Planung kontinuierlich durchführen.

## 7.5 Testen

Die vierte Balance befindet sich in der Testaktivität. Für eine Balance muss ein Projektteam entscheiden, wie sehr sie die Durchführung der Tests von der Entwicklung trennen. Bevor diskutiert wird, woraus das Risiko von zu wenig und zu viel Struktur in dieser Aktivität besteht, wird zuerst der Strukturgrad für die Testaktivität beschrieben.

### 7.5.1 Der Strukturgrad beim Testen

Beim Testen geht es darum, sicherzustellen, dass eine entwickelte Software keine Fehler enthält. Damit die Entwickler sich nicht selber überprüfen, sondern dies an einer unabhängigen Stelle geschieht, wird die Durchführung der Tests von Firmen häufig von der Entwicklung getrennt. In der Testaktivität entsteht Struktur dadurch, dass die Durchführung der Tests von den Entwicklern getrennt ist. Die Struktur steigt, je stärker diese Trennung betrieben wird.

Die Durchführung des Tests kann dabei personell oder zeitlich von der Entwicklung getrennt sein. Eine personelle Trennung bedeutet, dass andere Personen als die Entwickler für die Durchführung der Tests verantwortlich sind. Hierbei bedeutet eine starke Trennung, dass die Tests in einem gesonderten Testteam durchgeführt werden. Die zeitliche Trennung ist dadurch bestimmt, wie viel Zeit zwischen der Fertigstellung eines Features und der Durchführung der Tests vergeht. Die Tests können zum Beispiel direkt nach der Fertigstellung des Features erfolgen oder erst am Ende des Projektes. Der Strukturgrad bezieht sich hier auf die Durchführung der Integrations- und Systemtests, da die Interviewstudie gezeigt hat, dass alle Unit Tests direkt

in den Entwicklerteams durchgeführt werden. Für den Strukturgrad beim Testen wird die Variable $t \in [0, 100]$ definiert. Bei einem Wert von 0 werden alle Tests direkt nach der Fertigstellung des Features von den Entwicklern durchgeführt. Bei einem Wert von 100 werden alle Tests für alle Features von einem separierten Testteam erst am Ende des Projektes durchgeführt. Zwischen diesen beiden Extremwerten gibt es Abstufungen. Dafür werden an dieser Stelle die folgenden Stufen definiert:

**Stufe 1:** Auf dieser Stufe gibt es in jedem Entwicklerteam extra Tester, die sich um die Tests nach der Fertigstellung eines Features kümmern.

**Stufe 2:** Auf dieser Stufe gibt es am Ende eines Sprints oder Iteration einen extra definierten Zeitraum, in dem alle Tests für den Sprint oder die Iteration vom Entwicklerteam durchgeführt werden.

**Stufe 3:** Auf dieser Stufe, gibt es ein zur Entwicklung parallel arbeitendes Testteam (Pipeline Framework), das die Tests nach der Fertigstellung der Features durchführt.

Die Interviewstudie zeigt, dass eine stärkere Trennung von Entwicklung und Durchführen der Tests größere Struktur und damit Stabilität für die Qualität einer Software bedeutet. Abgeleitet aus den beiden allgemeinen Risiken von zu wenig und zu viel Struktur, gibt es daher für das Testen die beiden Risiken von zu wenig und zu viel Trennung beim Testen. Im Folgenden werden die Risikofaktoren beschrieben, aus denen die beiden Risiken bestehen. Zusätzlich werden ebenfalls die Kontextfaktoren beschrieben, die die Kritikalität der Risiken beeinflussen. Eine Liste der Risikofaktoren mit den dazugehörigen Kontextfaktoren befindet sich in Tabelle 7.4.

**Tabelle 7.4:** Liste der Risikofaktoren für zu wenig und zu viel Trennung beim Testen und der Kontextfaktoren

| **Testen** | | |
|---|---|---|
| **Risiko von zu wenig Trennung beim Testen** | | **Kontextfaktoren** |
| R4.1 | Qualitätsverlust bei der Integration | Größe des Projektteams |
| | | Verteilung der Entwicklung |
| | | Qualitätsstandards |
| | | Kritikalität der Integration |
| | | Aufwand der Integration |
| | | Verlust bei Fehlern |
| | | Eingebettete Software |
| **Risiko von zu viel Trennung beim Testen** | | **Kontextfaktoren** |
| R4.2 | Verzögerungen bei der Entwicklung | - |

### 7.5.2 Das Risiko von zu wenig Trennung beim Testen

In diesem Abschnitt wird der Risikofaktor des Risikos von zu wenig Trennung beim Testen beschrieben.

**R4.1: Qualitätsverlust bei der Integration** Dieses Risiko entsteht dadurch, dass es keine klare Verantwortung und Zentralisierung bei der Durchführung des Tests gibt. Einzelne Entwickler haben oft nicht den Überblick über die gesamte Software, um die Integrationstests und Systemtests für ihre Features eigenständig durchzuführen. Dadurch kann es passieren, dass Fehler bei der Integration entstehen und nicht entdeckt werden. Zudem bedarf es oftmals für die Erfüllung von Qualitätsstandards einer zentralen Stelle, die eine Aufsicht über die Durchführung der Tests hat und diese sorgfältig dokumentiert. Ohne klare Testverantwortung kann es passieren, dass der Testprozess chaotisch ist.

Der Schaden, der dadurch entstehen kann, ist, dass die Qualität der Software leidet und Qualitätsstandards nicht erfüllt werden. Es kann passieren, dass bei der Nutzung der Software Ausfälle beim Kunden auftreten. Diese zu beheben kostet die Softwarefirma viel Geld und kann zu einer einer Unzufriedenheit beim Kunden führen.

**Kontextfaktoren:** Die Kritikalität dieses Risikos wird durch eine Reihe von Kontextfaktoren beeinflusst. Bei einem großen Projektteam und/oder einer verteilten Entwicklung müssen verschiedene Softwareteile integriert werden. Dass dabei Fehler entstehen oder übersehen werden, ist generell wahrscheinlicher als in einem Projekt, dass nur ein kleines Entwicklerteam hat.

Zudem ist dieses Risiko kritisch, wenn die Integration großen manuellen Aufwand benötigt. Dadurch ist die Wahrscheinlichkeit größer, dass dabei Fehler passieren, im Gegensatz zu einer automatisierten Integration. Des Weiteren hängt das Risiko von der Kritikalität der Integration ab. Die Integration ist besonders kritisch, wenn einzelne Fehler in der Integration zu einem Systemversagen führen. In diesen Fällen muss die Integration besonders sorgfältig betrieben werden.

Bei einer eingebetteten Software kann diese erst richtig getestet werden, wenn auch die dazugehörige Hardware fertig gestellt ist. Bei der Entwicklung von eingebetteter Software ist es daher wichtig, dass diese zusammen mit der Hardware getestet wird. Wenn ein Projekt viele Qualitätsstandards einzuhalten hat oder die Software im Fehlerfall einen hohen Schaden, wie zum Beispiel den Verlust von Menschenleben, verursacht, ist das Risiko Fehler zu übersehen besonders kritisch.

**Diskussion der Risk Exposure von zu wenig Trennung beim Testen**

Die Interviewstudie hat gezeigt, dass die Verlagerung der Integrationstests zu extra Testexperten oder in ein extra Testteam bereits einen stark positiven Einfluss auf die Qualität hat. Eine stärkere Verlagerung dieser Tests, zum Beispiel ans Ende des Projektes, führt jedoch nicht zu einer deutlichen Verbesserung der Qualität. Aus diesem Grund sinkt die Wahrscheinlichkeit dieses Risikos mit einem niedrigen Strukturgrad bereits stark und mit einem höheren immer schwächer.

Der potenzielle Schaden ist mit einem geringen Strukturgrad sehr hoch, da möglicherweise viele Integrationsfehler übersehen werden und die Dokumentation der Tests chaotisch verläuft. Die Existenz von Testexperten, in denen sich die Verantwortungen und Expertise bündelt, hat bereits einen positiven Einfluss auf die Qualität. Damit sinkt der potenzielle Schaden mit einem geringen Strukturgrad bereits stark. Auch hier hat ein höherer Strukturgrad einen weniger starken Einfluss darauf Fehler zu finden. Daher sinkt der Schaden mit einem höheren Strukturgrad schwächer.

Aus dieser Diskussion folgt, dass die Risk Exposure von zu wenig Trennung beim Testen bei einem niedrigen Strukturgrad hoch ist und dabei stark sinkt. Mit einem steigenden Strukturgrad sinkt die Risk Exposure immer schwächer. Aus diesem Grund kann für die Risk Exposure von zu wenig Trennung beim Testen die in Abbildung 6.1 eingezeichnete blaue Kurve angenommen werden.

### 7.5.3 Das Risiko von zu viel Trennung beim Testen

In diesem Abschnitt wird der Risikofaktor für das Risiko von zu viel Trennung beim Testen beschrieben.

**R4.2: Verzögerungen bei der Entwicklung** Dieses Risiko tritt auf, wenn es eine zu starke Trennung zwischen der Fertigstellung von Features und der Durchführung der Tests gibt. Bei einem zu großen zeitlichen Abstand müssen sich die Entwickler für die Behebung des Fehlers erst wieder in das entwickelte Feature einarbeiten. Dies entfällt, wenn die Tests direkt im Anschluss an die Fertigstellung durchgeführt werden. Das Risiko tritt ebenfalls ein, wenn es eine zu starke personelle Trennung zwischen den Entwicklern und den Testern gibt. Dabei sind die Entwickler bei der Behebung der Fehler weniger flexibel, da die Mitteilung und Behebung durch einen formelleren Prozess geschieht.

Der Schaden, der in solchen Fällen auftritt, ist, dass die Entwicklungsgeschwindigkeit der Entwickler vermindert wird. Fehler müssen von den Entwicklern in ihrem laufenden Sprint behoben werden. Dies bedeutet, dass sie möglicherweise nicht ihr Sprintziel schaffen. Dies führt

zu Verzögerungen und mehr Kosten.

**Kontextfaktoren:** Für dieses Risiko gibt es keine expliziten Kontextfaktoren, die die Kritikalität dieses Risikos beeinflussen.

#### Diskussion der Risk Exposure von zu viel Trennung beim Testen

Die Eintrittswahrscheinlichkeit des oben beschriebenen Risikos ist auf den geringen Stufen des Strukturgrades niedrig und steigt schwach an. Das liegt daran, dass bei diesem Strukturgrad die Durchführung der Tests und die Fertigstellung der Features im gleichen Sprint stattfindet. Mit einer stärkeren Trennung steigt jedoch die Wahrscheinlichkeit, dass es zu Verzögerungen kommt, da immer mehr Zeit zwischen Fertigstellung und Durchführung der Tests liegt und die Entwickler sich erst wieder einarbeiten müssen.

Auch der Schaden ist mit einem geringen Strukturgrad gering und steigt schwach an, da die Entwickler direkt Feedback bei Fehlern bekommen und diese direkt beheben können. Es kostet die Entwickler dadurch weniger Zeit, als wenn viel Zeit dazwischen vergangen ist. Dadurch steigt der Schaden mit einem steigenden Strukturgrad stärker an.

Aus der Diskussion folgt, dass die Risk Exposure von zu viel Trennung beim Testen bei einem niedrigen Strukturgrad gering ist und schwach ansteigt.Mit einem steigenden Strukturgrad steigt die Risk Exposure immer stärker an. Aus diesem Grund kann für die Risk Exposure von zu viel Trennung beim Testen die in Abbildung 6.1 eingezeichnete orange Kurve angenommen werden.

### 7.5.4 Diskussion der Balance im Bereich des Testens

Für die Bestimmung der Balance beim Testen wird das Gesamtrisiko für diese Aktivität betrachtet. Aus der Analyse der Risk Exposure von zu wenig und zu viel Trennung beim Testen kann für das Gesamtrisiko die in Abbildung 6.1 eingezeichnete schwarze Kurve angenommen werden. Für Balance von beiden Risiken wird basierend auf dem Konzept der Tiefpunkt des Gesamtrisikos betrachtet.

Abgeleitet aus dem allgemeinen Konzept ist dieser erreicht, wenn ein Strukturgrad das Risiko von zu wenig Trennung beim Testen genau so stark sinken lässt, wie das von zu viel Trennung steigt. Dies bedeutet, dass Firmen, während sie das Risiko von Qualitätseinbußen bei der Integration mindern, darauf achten müssen, dass sie nicht gleichzeitig zu große Verzögerungen bei der Entwicklung erzeugen.

Bei einem niedrigen Strukturgrad dominiert beim Testen das Risiko von zu wenig Trennung. Ein Projektteam sollte sich daher zu Beginn darauf konzentrieren, das Risiko von Qualitätsverlusten bei der Integration zu mindern. In einem Projekt, das nur aus einem oder zwei

Projektteams besteht, kann dies durch Testexperten im Entwicklerteam geschehen. In einem Projekt mit mehreren Entwicklerteams sollte dagegen ein extra Testteam genutzt werden, das parallel zur Entwicklung arbeitet. Bei einer stärkeren Trennung dominiert das Risiko eine unnötige Verlangsamung der Entwicklung zu verursachen. Daher ist es an dieser Stelle wichtig, dass möglichst wenig Zeit zwischen der Fertigstellung der Features und der Durchführung der Tests durch das extra Testteam liegt. Die Interviewstudie zeigt, dass eine weitestgehende Automatisierung des gesamten Prozesses dabei hilfreich ist. Um weitere Verzögerungen zu vermeiden, ist es wichtig, dass Fehler schnell und einfach vom Testteam an die Entwickler zurückgemeldet werden können.

In Projekten, in denen die oben genannten Kontextfaktoren für das Risiko eines Qualitätsverlusts bei der Integration nicht vorkommen, ist auch das Risiko zu vernachlässigen. In einem solchen Projekt können die Integrations- und Systemtests in den Entwicklerteams durchgeführt werden. Hierbei kann es aber trotzdem von Vorteil sein, eine zusätzliche Expertenrolle für Tests zu haben, damit sich die Entwickler vollständig auf die Entwicklung konzentrieren können. Daraus ergeben sich die folgenden Prinzipien:

Prinzip 11: Ein Projektteam sollte sich darauf fokussieren eine klare Testverantwortung zu schaffen. In einem kleinen Projekt kann dies durch extra Testexperten passieren. In größeren Projekten durch ein extra Testteam.

Prinzip 12: Es sollte möglichst wenig Zeit zwischen der Fertigstellung von Features und der Durchführung von Tests liegen. Zudem sollte es einen schnellen und wenig formalen Meldeprozesses für gefundene Fehler geben.

Prinzip 13: Es sollte eine möglichst große Automatisierung des Testprozesses geben.

## 7.6 Koordination

Die letzte Balance findet in der Koordinationsaktivität statt. Für eine Balance muss ein Projektteam entscheiden, wie viel zentrale Koordination sie in einem Projekt benötigen und wie viel Selbstorganisation sie zulassen können. Selbstorganisation geht mit einer horizontalen Kommunikation einher, bei der Entwicklerteams untereinander Absprachen treffen. Bevor diskutiert wird, woraus das Risiko von zu viel und zu wenig Stabilität in dieser Aktivität besteht, wird der Strukturgrad für diese Aktivität beschrieben.

### 7.6.1 Der Strukturgrad für die Koordination

In der Koordination erzeugen die Firmen Strukturen dadurch, dass Entscheidungen und Absprachen in der Entwicklung zentral getroffen werden. Dies ermöglicht einen Überblick über getroffene Entscheidungen und verhindert Chaos bei der Entwicklung. Je mehr Entscheidungen in einem Projekt vom Projektteam zentralisiert werden, desto mehr Struktur und damit Stabilität wird geschaffen. Für den Strukturgrad bei der Koordination wird die Variable $e \in [0, 100]$ definiert. Bei einem Wert von 0 gibt es keine Einschränkungen bei der Selbstorganisation der Teams. Dies bedeutet, dass die Entwicklerteams alle Entscheidungen eigenständig treffen und es keine Projektleitung gibt, die sie mit einbeziehen müssen. Bei einem Wert von 100 wird das gesamte Projekt durch eine zentrale Projektleitung gesteuert. Dies bedeutet, dass die Entwicklerteams keine eigenständigen Entscheidungen und Absprachen treffen dürfen. Stattdessen werden alle Entscheidungen und Koordinationsaufgaben durch eine zentrale Projektsteuerung getroffen und durchgeführt. Zwischen diesen beiden Extrempunkten gibt es weitere Abstufungen. Dafür werden an dieser Stelle basierend auf den Ergebnissen der Interviewstudie die folgenden Stufen definiert:

**Stufe 1:** Auf dieser Stufe müssen die Entwicklerteams die Projektleitung nur einbeziehen, wenn die Projektparameter durch eine Entscheidung angepasst werden müssen.

**Stufe 2:** Auf dieser Stufe müssen die Entwicklerteams nur bei fundamentalen Entscheidungen und Absprachen, die alle Projektteams betreffen, die Projektleitung mit einbeziehen.

**Stufe 3:** Auf dieser Stufe müssen die Entwicklerteams bei allen Entscheidungen und Absprachen, die mindestens ein weiteres Team betreffen, die Projektleitung mit einbeziehen.

**Stufe 4:** Auf dieser Stufe müssen die Entwicklerteams bei allen Entscheidungen und Absprachen, die sie treffen die Projektleitung miteinbeziehen.

Mit einer stärkeren Zentralisierung der Entscheidungen gibt es eine immer zentralere Koordination. Diese bringt immer mehr Stabilität. Abgeleitet aus den beiden allgemeinen Risiken von zu wenig und zu viel Stabilität gibt es daher für die Koordination das Risiko von zu wenig und zu viel zentraler Koordination. Im Folgenden werden die Risikofaktoren beschrieben, aus denen die beiden Risiken bestehen. Zusätzlich werden ebenfalls die Kontextfaktoren beschrieben, die die Kritikalität der Risiken beeinflussen. Eine Liste der Risikofaktoren mit den dazugehörigen Kontextfaktoren befindet sich in Tabelle 7.5.

**Tabelle 7.5:** Liste der Risikofaktoren für zu viel und zu wenig zentraler Koordination und der Kontextfaktoren

**Koordination**

| **Risikofaktoren für zu wenig zentrale Koordination** | | **Kontextfaktoren** |
|---|---|---|
| R5.1 | Auftreten von Kommunikationsfehlern | Größe des Projektteams |
| | | Verteilung der Entwicklung |
| R5.2 | Die Archtitektur bildet sich zufällig | Größe des Projektteams |
| | | Verteilung der Entwicklung |
| | | Nutzung von Anwendungsframeworks |
| **Risikofaktoren für zu viel zentrale Koordination** | | **Kontextfaktoren** |
| R5.3 | Verzögerung bei der Entwicklung | - |

### 7.6.2 Das Risiko von zu wenig zentraler Koordination

In diesem Abschnitt sind die Risikofaktoren für das Risiko von zu wenig zentraler Koordination beschrieben.

**R5.1: Auftreten von Kommunikationsfehlern** Dieses Risiko entsteht, wenn Absprachen und Entscheidungen nicht durch eine zentrale Stelle gelenkt werden. Wenn Entscheidungen und Absprachen nur horizontal zwischen einzelnen Entwicklern oder Teams getroffen werden, kann es passieren, dass Kommunikationsfehler auftreten. Kommunikationsfehler treten zum Beispiel auf, wenn wichtige Entscheidungen und Informationen nicht zu allen Projektteilnehmern gleichermaßen fließen, da die Weitergabe von Informationen nicht zentral gesteuert wird. Zusätzlich kann es durch eine fehlende zentrale Steuerung dazu kommen, dass Entwicklerteams unterschiedliche Informationen besitzen und so möglicherweise widersprüchliche Entscheidungen innerhalb des Projektteams getroffen werden. Diese Kommnuikationsprobleme können dazu führen, dass Fehler bei der Entwicklung auftreten oder Komponenten nicht zueinander passen. Beides führt zu einer Verzögerung im Projekt, da Fehler behoben und Komponenten angepasst werden müssen.

**Kontextfaktoren:** Die Kritikalität dieses Risikos wird durch die Größe und Verteilung des Projektteams beeinflusst. Je mehr Entwickler an einem Projekt beteiligt sind, desto schwieriger ist es sicherzustellen, dass alle Informationen an alle Projektteilnehmer gleichermaßen fließen [32]. Daher ist dieses Risiko in großen Projektteams besonders kritisch. Das gleiche gilt für ein stark verteiltes Entwicklerteam. Auch hier ist es schwierig die Kommunikation ausreichend zwischen den einzelnen Standorten sicherzustellen.

**R5.2: Die Architektur entwickelt sich zufällig** Dieses Risiko tritt ein, wenn es zu wenig zentrale Koordination bei der Entwicklung der Architektur gibt und die Entwicklerteams die Ar-

chitekturentscheidungen eigenständig treffen. Da es dabei keine Steuerung der Entwicklung gibt, kann es sein, dass sich die Architektur auf zufälligem Wege entwickelt ohne einen zentralen Plan. Dies kann dazu führen, dass die Software schlecht wartbar ist. Dies wiederum bedeutet eine Unzufriedenheit des Kunden und kann möglicherweise zu teuren Änderungen der Software führen.

**Kontextfaktoren:** Die Kritikalität dieses Risikos wird von der Größe und Verteilung des Projektteams und der Verwendbarkeit von Anwendungsframeworks beeinflusst. In großen und verteilten Projektteams arbeiten die Entwicklerteams teilweise getrennt voneinander an verschiedenen Teilen der Architektur. Es ist schwierig für die Entwicklerteams sich eigenständig so zu koordinieren, dass die Architektur richtig weiterentwickelt wird. Daher ist dieses Risiko in großen und/oder verteilten Projektteams generell größer.

Die Kritikalität ist ebenfalls vom Einsatz von Anwendungsframeworks abhängig. Anwendungsframeworks geben bereits viel der Architekturdefinitionen vor [39]. Damit sind die Entwicklerteams bei der Entwicklung der Architektur weniger auf eine zentrale Koordination angewiesen. Wenn Anwendungsframeworks in einem Projekt genutzt werden können, ist dieses Risiko daher generell weniger kritisch.

#### Diskussion der Risk Exposure von zu wenig zentraler Koordination

Die Eintrittswahrscheinlichkeit der oben beschriebenen Risiken ist bei einem niedrigen Strukturgrad hoch, da Absprachen und Entscheidungen nur horizontal zwischen Entwicklerteams getroffen werden. Die Interviewstudie zeigt, dass mit der Koordination von Entscheidungen und Absprachen, die das gesamte Projekt betreffen, die horizontale Koordination bereits unterstützt werden kann. Dies trifft auch auf die Entwicklung der Architektur zu. Damit sinkt die Eintrittswahrscheinlichkeit der oben beschriebenen Risiken bereits mit einem geringen Strukturgrad.

Eine immer zentralere Steuerung aller Absprachen und Entscheidungen hat jedoch weniger Einfluss darauf, dass die Eintrittswahrscheinlichkeit gesenkt wird. Die Interviewstudie zeigt, dass Absprachen in Entwicklerteams oder einzelne Absprachen zwischen zwei Entwicklerteams von den Entwicklern selbstständig getroffen werden können. Damit sinkt die Eintrittswahrscheinlichkeit mit einem höheren Strukturgrad immer schwächer.

Dieselbe Argumentation gilt auch für den Schaden. Auf den niedrigen Stufen ist der Schaden sehr hoch, da hier Kommunikationsprobleme zwischen ganzen Entwicklerteams auftreten können oder die Architektur auf Projektebene zufällig wächst. Das kann dazu führen, dass ganze Komponenten nicht zueinander passen und die Software insgesamt schlecht wartbar ist. Dies führt dazu, dass die gesamte Software möglicherweise überarbeitet werden muss. Die Koordination auf Projektebene senkt daher bereits den potenziellen Schaden. Eine stärkere

zentrale Koordination senkt dagegen den potenziellen Schaden weniger, da hier Absprachen auf Teamebene betroffen sind. Wenn dort Probleme auftreten, können die Entwickler, dadurch dass sie eng zusammenarbeiten, diese schnell lösen.

Aus der vorangegangenen Diskussion folgt, dass die Risk Exposure von zu wenig zentraler Koordination bei einem niedrigen Strukturgrad hoch ist und stark fällt. Mit einem steigenden Strukturgrad fällt die Risk Exposure immer schwächer. Daher kann für den Verlauf der Risk Exposure von zu wenig Langzeitplanung die in Abbildung 6.1 eingezeichnete blaue Kurve angenommen werden.

### 7.6.3 Das Risiko von zu viel zentraler Koordination

In diesem Abschnitt wird der Risikofaktor für das Risiko von zu viel zentraler Koordination beschrieben.

**R5.3: Verzögerungen bei der Entwicklung** Dieses Risiko entsteht dadurch, dass bei zu vielen Entscheidungen und Absprachen die Projektleitung oder andere Managementgremien mit einbezogen werden müssen. Dies führt dazu, dass Entwicklerteams häufig auf Entscheidungen warten müssen und Absprachen nicht mehr einfach treffen können. Dadurch verzögert sich die Entwicklung und Deadlines können nicht eingehalten werden. Dies kann zu mehr Kosten und zu einer Unzufriedenheit des Kunden führen.

**Kontextfaktoren:** Für dieses Risiko gibt es keine konkreten Kontextfaktoren, die die Kritikalität beeinflussen.

#### Diskussion der Risk Exposure von zu viel zentraler Koordination

Die Wahrscheinlichkeit, dass Verzögerungen eintreten, ist auf den unteren Stufen bei der Koordination gering, da hier nur Absprachen betroffen sind, die sich auf das gesamte Projektteam beziehen. Diese kommen zum Beispiel im Laufe eines Sprintes weniger häufig vor, da sie vorher bereits getroffen wurden. Das heißt, es muss nur in wenigen Fällen die Projektleitung mit einbezogen werden. Auf den höheren Stufen beziehen die Entwicklerteams bei immer mehr Entscheidungen die Projektleitung mit ein. Damit steigt die Eintrittswahrscheinlichkeit des oben beschriebenen Risikos mit einem hohen Strukturgrad stark an.

Bei einem geringen Strukturgrad sind es wenige Entscheidungen, bei denen die Projektleitung benötigt wird. Die gesamte Verzögerung ist dadurch gering. Mit einem steigendem Strukturgrad wird der Schaden größer, da bei immer mehr Entscheidungen die Projektleitung mit einbezogen werden muss und es zu weiteren Verzögerungen kommt.

Aus der Diskussion folgt, dass die Risk Exposure von zu viel zentraler Koordination mit einem geringen Strukturgrad gering ist und gleichzeitig schwach ansteigt. Mit einem steigenden Strukturgrad steigt die Risk Exposure immer stärker an. Aus diesem Grund kann für die Risk Exposure von zu viel zentraler Koordination die in Abbildung 6.1 eingezeichnete orange Kurve angenommen werden.

### 7.6.4 Diskussion der Balance bei der Koordination

Für die Bestimmung der Balance der Koordination wird das Gesamtrisiko für diese Aktivität betrachtet. Aus der Analyse der Risk Exposure von zu viel und zu wenig zentraler Koordination kann für das Gesamtrisiko die in Abbildung 6.1 eingezeichnete schwarze Kurve angenommen werden. Für die Balance von beiden Risiken wird basierend auf dem Konzept der Tiefpunkt des Gesamtrisikos betrachtet.

Abgeleitet aus dem allgemeinen Konzept ist dieser erreicht, wenn in einem Strukturgrad das Risiko von zu wenig zentraler Koordination genau so stark sinkt, wie das von zu viel zentraler Koordination steigt. Das bedeutet, dass während die Firmen Entscheidungen und Absprachen für eine bessere Koordination zentralisieren, darauf achten müssen die Entwicklerteams nicht zu sehr zu verlangsamen.

Bei einem niedrigen Strukturgrad dominiert das Risiko von zu wenig zentraler Koordination. Dies bedeutet, dass sich die Firmen zuerst darauf fokussieren sollte Entscheidungen und Absprachen ausreichend zu zentralisiert, sodass keine Kommunikationsprobleme auftreten und die Architektur sich nicht zufällig entwickelt. Mit einem steigenden Strukturgrad steigt das Risiko, die Entwicklung zu stark zu verlangsamen.

Um das Risiko von zu wenig zentraler Koordination ausreichend zu senken, zeigt die Interviewstudie, dass es dafür zum einen ausreicht, wenn die Projektleitung bei Entscheidungen eingebunden ist, die mehrere Teams bzw. das gesamte Projektteam betreffen. Zudem sollte die Projektleitung involviert werden, wenn das Projektteam fundamental von der Aufgabenstellung abweichen muss oder die Projektparameter durch eine Entscheidung beeinflusst werden. Eine fundamentale Abweichung von der Aufgabenstellung tritt ein, wenn ein Entwicklungsteam einen anderen technischen Weg einschlagen muss als geplant. Dies ermöglicht dem Projektteam sicherzustellen, dass wichtige Informationen nicht verloren gehen.

Gleichzeitig wird genug Raum für einen schnellen horizontalen Austausch gelassen. Eine stärkere Zentralisierung von Entscheidungen und Absprachen sollte nicht vorgenommen werden. Stattdessen sollte auf eine Selbstorganisation und horizontale Kommunikation der Entwicklerteams gesetzt werden. Aus diesen Zusammenhängen werden die folgenden Prinzipien abgeleitet:

Prinzip 14: Ein Projektteam sollte sich darauf fokussieren, die Gefahr von Kommunikationsproblemen und einer zufälligen Architektur durch eine Zentralisierung von Entscheidungen und Absprachen zu senken.

Prinzip 15: Zur Erfüllung von Prinzip 14 sollte das Projektteam Entscheidungen und Absprachen zentralisieren, die mehrere Teams bzw. das gesamte Projektteam betreffen. Zudem sollte die Projektleitung mit einbezogen werden, wenn ein Projektteam fundamental von ihrer Aufgabenstellung abweichen muss oder durch eine Entscheidung die Projektparameter betroffen sind.

Prinzip 16: Wenn die weitere Zentralisierung von Entscheidungen und Absprachen die Erreichung von Prinzip 14 nicht mehr verbessert, sollten weitere Entscheidungen und Absprachen selbstorganisiert den Entwicklerteams überlassen werden.

# 8

# Ein Entwicklungsframework für APH-Ansätze

Das Ziel dieser Arbeit ist es, ein allgemeines Entwicklungsframework für die Kombination von agiler und plan-basierter Entwicklung zu schaffen. Dafür wird in dieser Arbeit *Design Science* als Forschungsvorgehen genutzt. Das Ziel von Design Science ist es, ein Artefakt zu schaffen, das einen bestimmten Zweck erfüllt [20]. Ein Artefakt kann dabei ein Tool, ein Prozess, eine Methodologie oder Ähnliches sein. In dieser Dissertation ist das Artefakt ein Entwicklungsframework für die Kombination von agiler und plan-basierter Entwicklung (APH-Framework). Da es wichtig ist, dass das Framework an einen speziellen Kontext angepasst werden kann, besteht das Artefakt zudem aus Regeln für die Anpassung des APH-Frameworks an einen speziellen Kontext.

In den Kapiteln 4 und 5 wurden die Anforderungen an ein solches Framework ermittelt und die Wissensbasis für das Framework geschaffen (Relevance und Rigor Cycle). Diese Kapitel haben gezeigt, dass es für die Kombination von agiler und plan-basierter Entwicklung wichtig ist beide Ansätze zu balancieren. In Kapitel 6 und 7 wurde ein Konzept für die Balance von APH-Ansätzen geschaffen, auf denen die Entwicklung des APH-Frameworks aufbaut (Design Cycle).

Ein Framework besteht aus Regeln, Strukturen und Praktiken [14]. Die Regeln stellen Richtlinien bei der Entwicklung dar. In Kapitel 7 wurden die Richtlinien als Prinzipien für eine APH-Entwicklung beschrieben. Dieses Kapitel beschreibt, wie diese Regeln von Projektteams während eines Projektes praktisch umgesetzt werden können. Dafür werden in diesem Kapitel die Strukturen und Praktiken für das APH-Framework beschrieben. Im zweiten Teil des Kapitels sind, basierend auf der Analyse der Risiken aus Kapitel 7, die Regeln für das Anpassen des APH-Frameworks an einen Kontext beschrieben.

## 8.1 Schaffung des APH-Frameworks

Die Ergebnisse der Literaturstudie (Kapitel 4) zeigen, dass für die Kombination von agiler und plan-basierter Entwicklung die Aktivität von Requirements Engineering, Architektur, Planung, Testen und Koordination jeweils balanciert werden müssen. Beim Requirements Engineering besteht die Balance darin, dass das Projektteam darüber entscheidet, wie viele Anforderungen sie bereits zu Beginn des Projektes analysieren und wann sie in ein kontinuierliches Requirements Engineering während der Entwicklung übergehen. Dasselbe gilt für die Definition der Architektur. Bei der Planung entscheidet das Projektteam darüber, wie detailliert ihre Langzeitplanung sein muss und welche Planung sie kontinuierlich durchführen.

Für diese drei Aktivitäten muss es daher in einem Framework für APH-Ansätze sowohl eine Umsetzung zu Beginn des Projektes geben als auch iterativ während des Projektes. In den Daten der Interviewstudie (Kapitel 5) zeigt sich, dass Firmen dafür am häufigsten eine Kombination des Wasserfall-Agil Frameworks mit dem Pipeline Frameworks nutzen. Das Pipeline Framework wird in den Fällen während der iterativen Entwicklung im Wasserfall-Agil Framework genutzt. Das Wasserfall-Agile Framework bietet einem Projektteam eine Anforderungs-, Planungs- und Architekturaktivität zu Beginn des Projektes. Die Implementierung eines Projektes erfolgt anschließend in Iterationen.

Die Verwendung des Pipeline Frameworks für die Iterationen bietet einen Vorteil für die Entwicklung. Bei dem Pipeline Framework können die einzelnen Aktivitäten Anforderungsanalyse, Architekturdesign und Planung gleichzeitig für verschiedene Inkremente während der Entwicklung durchgeführt werden. Dadurch gibt es konstante parallele Prozesse, die an unterschiedlichen Inkrementen arbeiten. Dies ermöglicht die Balance von agiler und plan-basierter Entwicklung zu implementieren. Zudem schafft das Pipeline Framework eine kontinuierliche Wertschöpfungskette von der Aufnahme von Anforderungen bis hin zur Umsetzung und Auslieferung. Diese kontinuierliche Wertschöpfungskette ist dagegen im Wasserfall-Iteration Framework nicht vorhanden, da dort die Aktivitäten nicht gleichzeitig durchgeführt werden. Ein

weiterer Vorteil, der für die Verwendung des Pipeline Frameworks spricht, ist, dass es von Projektteams einfach erweitert werden kann. Eine Schwierigkeit bei der Entwicklung von agilen Projekten ist zum Beispiel die Einbindung von Usabilitydesign [111]. Wenn Projektteams einen Fokus auf Usabilitydesign legen müssen, können sie diesen als parallelen Prozess zur Entwicklung einfügen. Dieser parallele Prozess würde vorgelagert vor der Implementierung das Usabilitykonzept einer Anwendung verwalten und weiterentwickeln.

Aus diesen Gründen bildet die Kombination aus Wasserfall-Agil und Pipeline Framework die Grundlage des APH-Frameworks. Die Wahl des Pipeline Frameworks ermöglicht es zusätzlich die Balance beim Testen in einem Entwicklungsprozess zu implementieren. Das Pipeline Framework bietet ebenfalls einen parallelen Testprozess, in dem die entwickelte Software aus der vorangegangenen Iteration von einem extra Testteam getestet wird.

Die Interviewstudie hat ergeben, dass es für die Koordination von mehreren Entwicklerteams von Vorteil ist, diesen zu Beginn einer Iteration eine Gelegenheit zu geben, sich abzusprechen und Abhängigkeiten zu klären. Das SAFe Essential Framework bieten durch seine Meetings eine Unterstützung für die Koordination von Entwicklerteams [27]. Daher werden die drei Meetings, die eine Iteration in SAFe Essential begleiten, in das APH-Framework integriert. Diese sind das Iteration Planning (PI Planning), Iteration Review (System Demo) und das Scrum of Scrums [27].

## 8.2 Grundlegender Aufbau des APH-Frameworks

In Abbildung 8.1 ist der grundlegende Aufbau des APH-Frameworks dargestellt. Die Rechtecke stehen für die Implementierung einer allgemeinen Aktivität, wie Requirements Engineering. Die Rechtecke werden als Unterscheidung zu der allgemeinen Definition von Aktivität aus Kapitel 2 in dieser Dissertation *Prozessaktivität* genannt. Nach der Definition von Stender [112] führt ein Projektteam sequenziell angeordnete Prozessaktivitäten nacheinander durch. Parallel angeordnete Prozessaktivitäten werden gleichzeitig durchgeführt und stellen parallele Prozesse in einem Projekt dar.

Die Grundlage für das Framework bildet das Wasserfall-Agile Framework. Daher gibt es zu Beginn eines Projektes Aktivitäten für die Anforderungsanalyse, die Architektur und die Planung. Das Ziel dieser Aktivitäten zu Beginn des Projektes ist es, eine ausreichende Stabilität für das Projekt zu schaffen, sodass das Projektteam zuversichtlich ist, dass es das Projekt erfolgreich durchführen kann und gleichzeitig genug Flexibilität für das Projekt behält. Wie die Projektteams dies erreichen können und welche Tätigkeiten dafür notwendig sind, wurde bereits in Kapitel 7 diskutiert.

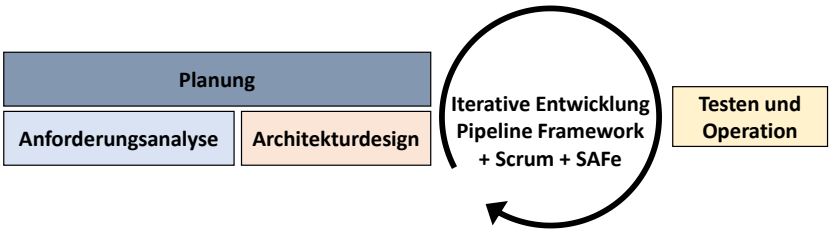


**Abbildung 8.1:** Grundlegender Aufbau des APH-Frameworks bestehend aus dem Wasserfall-Agil Framework

Auf die Aktivitäten zu Beginn des Projektes folgt die iterative Entwicklung. Diese ist ausführlich im folgenden Abschnitt beschrieben. Zur Vollständigkeit des Frameworks gibt es am Ende eines Projektes eine Aktivitäte für die Durchführung der Tests und der Operation der Software. Die Daten der Literatur- und Interviewstudie zeigen, dass diese Aktivität nur für eingebettete Software in einer Hardwareumgebung nötig ist. Diese Aktivität ist daher zur Vollständigkeit vorhanden, wird aber nur in diesen Fällen benötigt.

## 8.3 Aufbau der iterativen Entwicklung

Die Entwicklung im APH-Framework (Abbildung 8.1) ist in Iterationen aufgeteilt. Die Länge dieser Iterationen können zwischen drei und sechs Monaten lang sein. Der Verlauf einer Iteration ist in Abbildung 8.2 abgebildet und folgt dem SAfe Essential Framework [27]. Die gestrichelten Rechtecke stellen Meetings dar. Eine Iteration startet mit einem Iteration Planning Meeting. In diesem besprechen die Entwicklerteams die kommende Iteration und klären Abhängigkeiten. Dieses Meeting unterstützt die horizontale Koordination von Entwicklerteams. Am Ende der Iteration erfolgt ein Iteration Review Meeting. Dabei sammelt das Projektteam Feedback über das Ergebnis der Iteration vom Kunden und Stakeholdern. Dieses Meeting ermöglicht Feedbackschleifen für die integrierten Features, die während einer Iteration entwickelt wurden. Auf diese Weise unterstützt dieses Meeting das kontinuierliche Requirements Engineering, da das Meeting Änderungswünsche und mögliche neue Ideen der Kunden und Stakeholder hervorbringt.

Während einer Iteration arbeiten die Teams in Sprints. Hierbei können sich die einzelnen Entwicklerteams sich ihre eigenen Methoden und Praktiken suchen. Aus der Interviewstudie (Kapitel 5) folgt, dass die Entwicklerteams sich nur an die Strukturen der übergeordneten Iteration halten müssen. Dies bedeutet, dass die Entwicklerteams im gleichen Rhythmus arbeiten, der durch die Meetings Iteration Planning und Iteration Review vorgegeben ist. Scrum ist das am meisten verbreitete Framework für das Vorgehen von Entwicklerteams [2]. Daher wird auch für das APH-Framework das Scrum Vorgehen für die einzelnen Entwicklerteams verwendet.

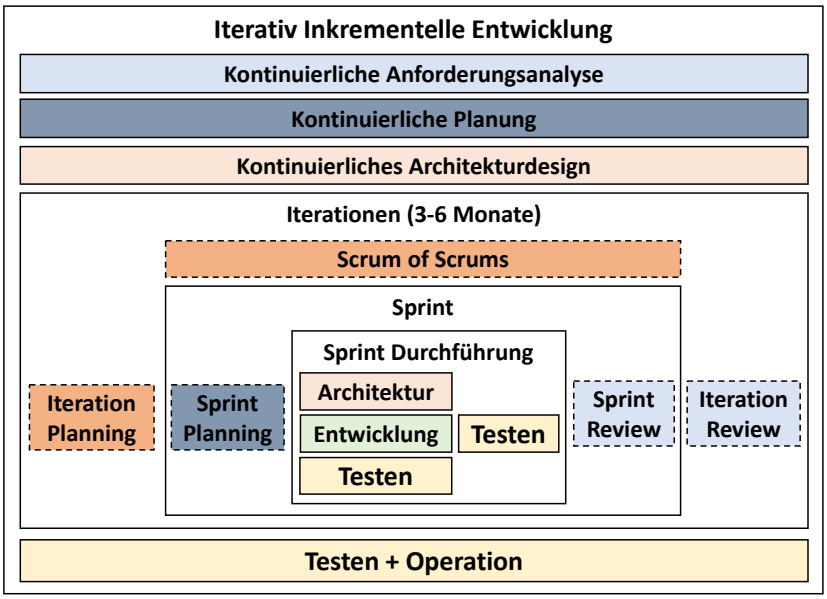


**Abbildung 8.2:** Aufbau der Iterationen im APH-Framework

Wie in Abbildung 8.2 zu sehen ist, besteht ein Sprint aus einem Planning Meeting, der Sprint Ausführung und einem Sprint Review. Während der Sprint Ausführung trifft das Entwicklerteam Architekturentscheidungen, die das eigene Teams betreffen und implementieren die Anforderungen aus dem Sprint Backlog. Beim Testen gibt es eine Erweiterung des Scrum Vorgehens im APH-Framework. Im APH-Framework gibt es die Option, dass das Entwicklerteam für die Intergrations- und Systemtests eine Testaktivität am Ende eines Sprints nutzt. Wann diese genutzt werden sollte, wird bei der Anpassung des Frameworks an einen Kontext diskutiert.

Parallel zu der Arbeit der Entwicklerteams gibt es einen Prozess für ein kontinuierliches Requirements Engineering. Die Tätigkeiten, die die Product Owner in diesem Prozess durchführen, sind in Abschnitt E.1 in Appendix E aufgeführt und wurden aus den Daten der Interviewstudie abgeleitet (siehe Kapitel 5). Die Hauptaufgabe des Product Owner ist es, die nächsten Iterationen inhaltlich zu planen, indem sie ein Inkrement für die Software definieren. Zusätzlich gibt es einen parallelen Prozess für die Planung. In diesem ist eine Projektleitung und/oder das Product Owner Team für die kontinuierliche Planung des Projektes zuständig. Die wichtigsten Aufgaben hierbei sind das Verwalten der Roadmap und deren Weiterführung (Planen von Abhängigkeiten und Meilensteinen), die Zusammenstellung von Iterationen und die Überprüfung des Projektstatus. Der Projektstatus muss beständig mit der groben Schätzung des Projektes verglichen werden. Zusätzlich zu diesen beiden parallelen Prozessen gibt es das kontinuierliche Design der Architektur. In dieser Prozessaktivität behält ein Systemarchitekt (oder ein Team von Architekten) den Überblick über die Gesamtarchitektur der Software und trifft Architekturentschei-

dungen auf Projektebene. Zusätzlich unterstützt der Systemarchitekt die Entwickler bei der Definition der Architektur in den ihren Teams.

Die Prozessaktivität für die Planung und das Requirements Engineering kümmern sich um die Definition und Planung von kommenden Iterationen. Sie schreiten daher der Entwicklung voraus und arbeiten an kommenden Inkrementen. Die Untersuchungen in der Interviewstudie haben ergeben, dass die parallele Prozessaktivität für die Architektur der Entwicklung weniger deutlich vorausschreitet. Stattdessen stellt sie eine Unterstützung für die Entwicklerteams dar. Trotzdem ist es die Aufgabe des Systemarchitekten, kommende Anforderungen zu analysieren und die Architektur für eine Implementierung vorzubereiten.

Der parallele Testprozess läuft der Entwicklung nach. Dieser ist in Abbildung 8.2 als lange gelbe Prozessaktivität dargestellt. In dieser Prozessaktivät werden die Integrations-und Systemtests von einem parallel arbeitenden Testteam nach der Fertigstellung von Features durchgeführt.

## 8.4 Kontextanpassung des APH-Frameworks

Das oben beschriebene APH-Framework ermöglicht es, dass ein Projektteam Aktivitäten sowohl zu Beginn des Projektes als auch während des Projektes ausführen kann. Das Framework ermöglicht es daher einem Projektteam, eine Balance von agiler und plan-basierter Entwicklung zu implementieren.

Wie in Kapitel 6 beschrieben, werden bei der Balance der einzelnen Aktivitäten die beiden Risiken von zu wenig und zu viel Stabilität gegeneinander abgewogen. Der Kontext beeinflusst, wie viel Einfluss das jeweilige Risiko bei der Balance der Aktivitäten hat. Zum Beispiel führt eine geringe Bekanntheit der Anforderungen dazu, dass das Risiko von zu viel Anforderungsanalyse kritisch ist. Die Balance im Requirements Engineering verschiebt sich in diesem Fall tendenziell nach links auf dem Strukturgrad (siehe Abbildung 6.1). Anders verhält es sich, wenn das Risiko von zu wenig Anforderungsanalyse kritisch ist. In diesem Fall verschiebt sich die Balance tendenziell nach rechts auf dem Strukturgrad. Diese Verschiebung bedeutet, dass unterschiedliche Mengen an Anforderungen zu Beginn eines Projektes erhoben werden, und führt dazu, dass das APH-Frameworks angepasst werden muss. Das Aufeinandertreffen von unterschiedlichen Kritikalitäten der beiden Risiken von zu wenig und zu viel Stabilität sorgen für eine Anpassung des APH-Frameworks.

Das Konzept für die Anpassung des APH-Frameworks wird am Beispiel von Requirements Engineering erklärt. Für die Anpassung im Requirements Engineering sind die beiden Risiken von zu wenig und zu viel Anforderungsanalyse relevant. Für die Analyse möglicher Anpassung

werden diese Risiken mit unterschiedlicher Kritikalität in Kombination betrachtet. Eine sehr feine Abstufung der Kritikalität eines Risikos würde dabei zu zu vielen Kombinationsmöglichkeiten führen, die die Komplexität der Anpassung stark steigern würden. Zudem wären die Unterschiede in den Kontexten zu gering, um aussagekräftige Aussagen treffen zu können.

Ob ein Risiko besonders bei der Entwicklung beachtet werden muss, hängt von Kontextfaktoren ab. Das Risiko kritische, nicht-funktionale Anforderungen zu übersehen ist in einem Projekt ohne solche Anforderungen nicht vorhanden, in einem Projekt mit vielen solchen Anforderungen dagegen schon. Das Vorhandensein solcher Anforderungen führt daher dazu, dass das Risiko kritisch ist und für die Anpassung des APH-Frameworks berücksichtigt werden muss. Es wird in dieser Dissertation daher bei der Kritikaltiät eines Risikos nur zwischen zwei Zuständen unterschieden. Entweder ist das Risiko kritisch oder nicht. Ein Risiko ist genau dann kritisch, wenn mindestens einer der zugeordneten Kontextfaktoren (siehe Kapitel 7) einen Schwellenwert erreicht. In Abschnitt E.6 in Appendix E sind die Kontextfaktoren mit ihren möglichen Wertebereichen angegeben. Zum Beispiel gibt es für die Größe des Projektteams die Werte klein (1-2 Teams), groß (3-9 Entwicklerteams) und sehr groß (mehr als 10 Entwicklerteams). Der Schwellenwert liegt hier bei 3-9 Entwicklerteams. Dies wird daraus abgeleitet, dass ein solches Projekt bereits als *large-scale* Projekt gilt [113]. In einem solchen Kontext hat das Risiko von Unsicherheit über das Projektziel und Abhängigkeiten eine Relevanz.

Das Risiko von zu wenig Anforderungsanalyse besteht aus den Risikofaktoren R1.1-R1.3 und das von zu viel Anforderungsanalyse aus dem Risikofaktor R1.4. Um mögliche Anpassungen zu untersuchen, habe ich die Kombinationen von R1.1-R1.3 und R1.4 untersucht. Dafür habe ich eine Matrix geschaffen. Diese ist in Tabelle 8.1 abgebildet.

**Tabelle 8.1:** Kombinationen zwischen den Risikofaktoren von R1.1, R1.2, R1.3 und R1.4

| **Risiko von zu wenig Anforderungsanalyse** | **Risiko von zu viel Anforderungsanalyse** | |
|---|---|---|
| | **R1.4 ist nicht kritisch** | **R1.4 ist kritisch** |
| **R1.1 - R1.3 sind nicht kritisch** | Alternative 1 | Alternative 2 |
| **R1.1 ist kritisch** | Alternative 3 | Alternative 4 |
| **R1.2 ist kritisch** | Alternative 5 | Alternative 6 |
| **R1.3 ist kritisch** | Alternative 7 | Alternative 8 |

In diesem Schritt habe ich die einzelnen Kombinationen der Risiken betrachtet. Dabei habe ich aus den Informationen der Literatur- und Interviewstudie und der Analyse des Konzeptes, Handlungsalternativen für das APH-Framwework für jede Kombination gebildet. In Alternative 3 ist zum Beispiel das Risiko kritisch, Unsicherheit über das Projektziel und Abhängigkeiten zu haben. Die Anforderungen sind weitestgehend bekannt (R1.4 ist nicht kritisch). Dadurch ist einem Projektteam möglich, Anforderungen mehr im Detail aufzunehmen. Dies bedeutet, dass

zum Beispiel die Funktionen in den Anwendungsfällen genauer festgelegt werden können.

In Alternative 2 hingegen ist das Risiko von zu wenig Anforderungsanalyse nicht kritisch, da keiner der Risikofaktoren kritisch ist. Gleichzeitig gibt es eine geringe Bekanntheit der Anforderungen (R1.4 ist kritisch). In diesem Fall sollte das Projektteam daher nur die nötigsten Anforderungen erheben, um mit dem Projekt starten zu können.

Im zweiten Schritt der Analyse habe ich die Alternativen nach Äquivalenzklassen untersucht. Dies bedeutet, dass ich solche Fälle zusammengelegt habe, die ein gleiches Vorgehen für ein Projektteam bedeutet. Das Ergebnis für die Bildung von Äquivalenzklassen für das Requirements Engineering ist in Tabelle 8.2 zu sehen.

**Tabelle 8.2:** Äquivalenzklassen für die Kombination der Risikofaktoren R1.1, R1.2, R1.3 und R1.4

| **Risiko von zu wenig Anforderungsanalyse** | **Risiko von zu viel Anforderungsanalyse** | |
|---|---|---|
| | **R1.4 ist nicht kritisch** | **R1.4 ist kritisch** |
| **R1.1 - R1.3 sind nicht kritisch** | Äquivalenzklasse 1 | Äquivalenzklasse 2 |
| **R1.1 ist kritisch** | Äquivalenzklasse 3 | Äquivalenzklasse 4 |
| **R1.2 ist kritisch** | | |
| **R1.3 ist kritisch** | | |

Die Alternativen in diesem Beispiel konnten zu vier Äquivalenzklassen zusammengefasst werden. Zum Beispiel ist das Vorgehen in Alternative 3, 5 und 7 ähnlich. In all diesen Alternativen kann das Projektteam, dadurch dass das Risiko von zu viel Anforderungsanalyse nicht kritisch ist, die Anforderungen soweit erheben, dass die Risikofaktoren R1.1 - R1.3 ausreichend gesenkt sind. Dies gilt für alle diese Alternativen, auch wenn R1.1 - R1.3 gleichzeitig auftreten. Daher können diese Alternativen zu einer Äquivalenzklasse zusammengefügt werden.

Die Analyse von Alternativen und die anschließende Bildung von Äquivalenzklassen habe ich für jede der Aktivitäten durchgeführt. Im Folgenden sind die einzelnen Äquivalenzklassen für die Aktivitäten beschrieben. Die Äquivalenzklassen mit Aussagen aus der Interviewstudie und der Literaturstudie sind in Appendix D aufgelistet.

### 8.4.1 Requirements Engineering

Die Äquivalenzklassen im Requirements Engineering bestimmen das Vorgehen in der Anforderungsanalyse zu Beginn eines Projektes und während der kontinuierlichen Anforderungsanalyse.

**Äquivalenzklasse A1: R1.1, R1.2, R1.3 und R1.4 sind nicht kritisch** In diesem Kontext sind die Risikofaktoren R1.1 - R1.3 nicht kritisch. Dadurch ist das Risiko von zu wenig Anforderungsanalyse insgesamt nicht kritisch. Gleichzeit sind die Anforderungen gut bekannt, wodurch

R1.4 nicht kritisch ist. Dies bedeutet, dass die Anforderungen bereits soweit erhoben werden können, wie sie bekannt sind. Bei der Erhebung der Anforderungen sollte das Projektteam, die in Kapitel 7 beschriebene Balance beachten. Um die Anforderungsanalyse zu Beginn des Projektes nicht unnötig zu verlängern, reicht es jedoch aus, die Anforderungen soweit zu erheben, bis das Projektteam ein gemeinsames Verständnis für die Anforderungen erreicht hat. Dies ist erreicht, wenn das Projektteam gemeinsame eine grobe Schätzung für den geplanten Anforderungsumfang abgeben kann und sie gemeinsam beschließen, dass sie genug Informationen gewonnen haben, um mit dem Projekt beginnen zu können.

**Äquivalenzklasse A2: R1.1, R1.2, R1.3 sind nicht kritisch und R1.4 ist kritisch** Da die Risiken R1.1 - R1.3 in diesem Kontext nicht kritisch sind, ist das Projektteam in der Lage, mit einem minimalen Verständnis der Anforderungen in ein Projekt zu starten. Das Projektteam ist klein und nicht verteilt. Dadurch können diese schnell kommunizieren und benötigen zu Beginn keine gemeinsame Ausrichtung, die die Koordination unterstützt. Es reicht aus, dass sich das Projektteam zu Beginn auf die Bildung einer Vision und der Erhebung von wenigen Anwendungsfällen auf der Ebene von Epics oder auf die genaue Beschreibung eines Anwendungsfalls beschränkt. Es reicht aus, wenn das Projektteam eine sehr grobe Schätzung für das Projekt abgeben kann. Dadurch dass R1.4 kritisch ist, sollte das Projektteam nur die wichtigsten Anwendungsfälle identifizieren und diese nur grob analysieren.

Da nur wenige Anforderungen zu Beginn erhoben werden, liegt der Fokus beim kontinuierlichen Requirements Engineering auf der Erhebung der weiteren Anforderungen. Dafür ist ein Fokus auf Feedbackzyklen von Vorteil. Über das Einholen von Feedback von Kunden und Stakeholdern ergeben sich neue Anforderungen und Änderungswünsche. Auf diese Weise steuert das Projektteam das Projekt in die richtige Richtung.

**Äquivalenzklasse A3: Mindestens ein Risiko von R1.1, R1.2 und R1.3 ist kritisch und R1.4 ist nicht kritisch** Mindestens eins der Risiken von R1.1 - R1.3 ist in diesem Kontext kritisch, während R1.4 nicht kritisch ist. Dies bedeutet, dass das Risiko von zu wenig Anforderungsanalyse überwiegt und sich die Balance auf dem Strukturgrad nach rechts bewegt. Daraus folgt, dass mehr Anforderungen aufgenommen werden sollten als in Abschnitt 7.2.4 beschrieben ist.

Wenn R1.1 kritisch ist, sollte sich das Projektteam darum bemühen, eine komplette Übersicht über die Anwendung zu verschaffen. Dies bedeutet, dass das Projektteam eine klare Vision erstellt und die Anwendungsfälle und wie diese zusammenhängen analysiert. Zudem ist es wichtig Abhängigkeiten zu anderen Projekten zu analysieren. Zur Minimierung von R1.1 muss das Projektteam nicht die genauen Nutzerschritte und Systemfunktionen der Anwendungsfälle

analysieren. Diese betreffen die Arbeit in den einzelnen Entwicklerteams und unterstützen nicht die Koordination des gesamten Projektes. Hierbei ist die Gefahr größer, dass sich das Projektteam zu sehr im Detail verliert.

Das Projektteam sollte die Funktionen von Anwendungsfällen genauer analysieren, wenn R1.2 kritisch ist. Wenn R1.2 kritisch ist, muss das Projektteam ein ausreichendes Verständnis über die Umsetzung bekommen, um genug Informationen für eine Schätzung haben. Daher ist es in einem Kontext, in dem R1.2 kritisch ist, von Vorteil, dass das Projektteam die Funktionen der Anwendungsfälle näher betrachtet. Zur Unterstützung können auch teilweise die Nutzer- und Systemschritte analysiert werden. Hierbei muss darauf geachtet werden, dass dies nur soweit betrieben wird, wie es es das Risiko R1.2 nötig macht. Die genauen technischen Anforderungen müssen nicht betrachtet werden, da diese in der Verantwortung der Entwickler liegt. Wenn R1.3 kritisch ist, sollte sich das Projektteam darauf fokussieren, die kritischen, nichtfunktionalen Anforderungen zu erheben.

Dadurch dass R1.4 in diesem Kontext nicht kritisch ist, ist es dem Projektteam möglich die nötigen Anforderungen, die zur Senkung von R1.1-R1.3 nötig sind, festzulegen. Um die Anforderungsanalyse zu Beginn des Projektes jedoch nicht unnötig zu verlängern, sollte das Projektteams keine unnötigen Details während der Anforderungsanalyse erheben. Details, die auch während des Projektes erhoben werden können, sollten während der kontinuierlichen Anforderungsanalyse analysiert werden.

**Äquivalenzklasse A4: Mindestens ein Risiko von R1.1, R1.2 und R1.3 ist kritisch und R1.4 ist kritisch** In diesem Kontext ist sowohl das Risiko von zu viel und zu wenig Anforderungsanalyse kritisch. Um das Gesamtrisiko zu senken, sollte das Projekt eine höhere Bekanntheit der Anforderungen erreichen. Dies ermöglicht, dass die Risiken von R1.1-R1.3 gesenkt werden können. Um gleichzeitig das Risiko von R1.4 nicht zu erhöhen, sollten für die größere Bekanntheit Negotiationstechniken und Prototyping genutzt werden. Durch Negotiationstechniken kann ein Projektteam zwischen Stakeholdern eine größere Einigkeit über Anforderungen erreichen. Protoyping hilft Stakeholdern dabei größere Klarheit zu gewinnen, was sie an Anforderungen möchten und hilft dem Projektteam die Umsetzung von Anforderungen besser zu verstehen. Dadurch dass die Anforderungen weiterhin unsicher sind, sollten diese Tätigkeiten nur so lange betrieben werden, bis die Risiken von R1.1-R1.3 so weit gesenkt sind, dass das Projektteam zuversichtlich ist, mit dem Projekt starten zu können. Hierbei ist es wichtig, dass das Projektteam die Anwendungsfälle, ähnlich zu Äquivalenzklasse A3, nicht unnötig detailliert betrachtet.

Das Projektteam sollte während des kontinuierlichen Requirements Engineering einen Fo-

kus auf Feedbackschleifen legen. Dadurch dass R1.4 kritisch ist, kann es passiert sein, dass während der Anforderungsanalyse zu Beginn des Projektes falsche Anforderungen aufgenommen wurden. Die Feedbackzyklen und das Aufnehmen von zusätzlichen Anforderungen stellt sicher, dass sich das Projekt in die richtige Richtung bewegt.

**Kontext: Viele Stakeholder** Bei einer großen Menge an Stakeholdern, die während der Projektlaufzeit verwaltet werden müssen, kann das Projektteam zur Unterstützung der Product Owner ein zusätzliches Team von Business Analysten nutzen. Dieses Team unterstützt die Product Owner bei der Kommunikation mit den Stakeholdern und dem Erheben weiterer Anforderungen.

### 8.4.2 Architektur

Die Äquivalenzklassen in der Architektur betreffen Anpassungen für da Architekturdesign zu Beginn des Projektes, die kontinuierliche Architekturdesign und die Definition der Architektur in den einzelnen Entwicklerteams während der Sprintdurchführung.

**Äquivalenzklasse D1: R2.4 ist nicht kritisch** In diesem Kontext ist das Risiko von zu viel Architekturdefinition nicht kritisch. Dies bedeutet, dass das Projektteam die Architektur so weit entwickeln kann, dass alle möglichen Risiken von R2.1 - R2.3 ausreichend gesenkt werden. Wenn R2.1-R2.3 nicht kritisch sind, reicht es aus, dass für die Architektur die generellen Architekturpattern und die verwendeten Technologien (Technologiestack) definiert werden. Andernfalls sollte das Projektteam einen deutlichen Fokus darauf legen, dass es in diesem Kontext die nötigen Architekturentscheidungen zu Beginn trifft, sodass das Projektteam zuversichtlich ist, mit dem erreichten Stand an Architekturdefinition die Software entwickeln zu können. Wenn R2.1 kritisch ist, sollte das Projektteam die Architektur soweit definieren, dass sie zu Beginn des Projektes einen Plan hat, wie sich die Architektur weiterentwickeln soll. Wenn R2.2 kritisch ist, sollte das Projektteam die Abhängigkeiten zu anderen Projekten und innerhalb des Teams analysieren und diese in der Architektur berücksichtigen. Wenn R2.3 kritisch ist, sollten alle kritischen, nicht-funktionalen Anforderungen soweit umgesetzt werden, dass das Projektteam zuversichtlich ist, dass sie diese auf diese Weise umsetzen können.

Um trotzdem genug Flexibilität bei der Definition der Architektur zu behalten und das Architekturdesign zu Beginn des Projektes nicht unnötig zu verlängern, sollte das Projektteam nur die nötigen Entscheidungen treffen, die für die Risiken R2.1-R2.3 nötig sind. Alle weiteren Entscheidungen sollten zurückgehalten werden.

Wenn eins der Risiken von R2.1-R2.3 kritisch ist, sollte für die Weiterentwicklung der Architektur die Prozessaktivität für das kontinuierliche Architekturdesign genutzt werden. Dies unterstützt dabei, dass sich die Architektur in die richtige Richtung entwickelt.

**Äquivalenzklasse D2: R2.1, R2.2 und R2.3 sind nicht kritisch und R2.4 ist kritisch** In diesem Kontext überwiegt das Risiko von zu viel Architekturdefinition. Dies bedeutet, dass sich die Balance bei der Architektur nach links auf dem Strukturgrad verschiebt. In diesem Kontext sollten daher zu Beginn des Projektes nur die minimalen Architekturentscheidungen getroffen werden. Diese betreffen den Technologiestack und die grundlegenden Architekturpattern. Alle weiteren Entscheidungen sollten soweit wie möglich zurückgehalten werden. Dadurch dass es nur ein oder zwei Teams gibt und diese nicht verteilt sind, kann die Architektur direkt in den Entwicklerteams im Lauf des Projektes weiterentwickelt werden, sodass der parallele Architekturprozess weggelassen werden kann.

**Äquivalenzklasse D3: Mindestens eines der Risiken R2.1, R2.2 oder R2.3 ist kritisch und R2.4 ist kritisch** In diesem Kontext ist sowohl das Risiko von zu wenig und zu viel Architekturdefinition kritisch. In diesem Kontext sollte sich das Projektteam zu Beginn des Projektes darauf konzentrieren die grundlegenden Architekturpattern zu definieren, Schnittstellen für die Koordination zu definieren und die kritischen, nicht-funktionalen Anforderungen zu berücksichtigen. Wichtig in diesem Kontext ist es, die Arbeit von unterschiedlichen Entwicklerteams in der Architektur durch Schnittstellen zu kapseln, damit diese unabhängig arbeiten können. Da R2.4 ebenfalls kritisch ist, ist es wichtig, dass das Entwicklerteam nicht über diese Entscheidungen hinausgeht. Stattdessen sollten alle weiteren Architekturentscheidungen soweit wie möglich zurückgehalten werden. Im Gegensatz zu Äquivalenzklasse D1 sollte das Projektteam in dieser Klasse auch keinen Plan über die Weiterentwicklung der Software machen. Dieser verursacht sonst zu viel Aufwand durch Änderungen.

Da während der Entwicklung wichtige Architekturentscheidungen getroffen werden müssen, sollte ein Fokus darauf gelegt werden, dass die Entwicklung durch den parallelen Prozess gesteuert wird.

### 8.4.3 Planung

Diese Alternativen betreffen die Planung zu Beginn des Projektes und die kontinuierliche Prozessaktivität für die Planung parallel zur Entwicklung.

**Äquivalenzklasse P1: R3.1 und R3.2 sind nicht kritisch** In diesem Kontext ist das Risiko von zu wenig Langzeitplanung nicht kritisch. Dies bedeutet, dass der Kunde keine genaue Bindung von Zeit, Umfang und Budget benötigt. Für die Verwaltung des Umfanges kann daher der Flexibilitätsmodus verwendet werden. Wenn in diesem Kontext R3.3 kritisch ist, sollte das Projektteam stark darauf achten, dass es für die Schätzung des geplanten Umfangs des Projektes einen ausreichend großen Toleranzbereich lässt. Wenn R3.3 nicht kritisch ist, kann dieser Toleranzbereich kleiner ausfallen. Da R3.2 in diesem Kontext nicht kritisch ist, sollte die Verwendung einer Roadmap lediglich dafür genutzt werden, um die Entwicklungsreihenfolge von Features zu planen und so die Vision zu transportieren. Um die Entwicklung zu strukturieren, kann diese in größere Iterationen eingeteilt werden [16].

Wenn das Projektteam nicht groß und/oder verteilt ist, kann das Projektteam einen Fokus auf Kurzzeitplanung legen. Die Planung ist dadurch darauf fokussiert, dass die Entwicklerteams kontinuierlich die nächsten zwei oder drei Sprints planen, indem sie die Anforderungen für die Sprints zusammenstellen. Die Roadmap sollte beständig zentral weitergeführt und verwaltet werden.

**Äquivalenzklasse P2: Das Risiko 3.1 ist kritisch** In diesem Kontext ist das Risiko R3.2 nicht kritisch, da der Kunde keine genaue Bindung von Zeit, Umfang und Budget benötigt. Für die Verwaltung des Umfanges kann daher der Flexibilitätsmodus verwendet werden. Wenn R3.3 in diesem Kontext kritisch ist, sollte das Projektteam stark darauf achten, dass sie für die Schätzung zu Beginn des Projektes einen ausreichend großen Toleranzbereich lässt. Wenn R3.3 nicht kritisch ist, kann dieser Toleranzbereich kleiner ausfallen.

Da in diesem Kontext 3.1 kritisch ist, ist es wichtig, dass das Projektteams Abhängigkeiten zu anderen Projekten und innerhalb des Projekts in einer Roadmap plant. Während des Projektes ist es wichtig, dass diese beständig weitergeführt wird. Darauf sollte der Fokus während der kontinuierlichen Planung liegen. Es sollten jedoch nur Meilensteine in die Roadmap aufgenommen werden, die bei der Koordination von Abhängigkeiten helfen. Feinere Meilensteine, zum Beispiel über die interne Arbeit von Entwicklerteams, sollten nicht erstellt werden. Die Entwickler planen die Arbeit in ihren Sprints selbständig. Dafür befüllen sie die nötige Anzahl von Sprints (zwei bis drei Sprints) im voraus. Dies stellt sicher, dass die Entwickler nicht unnötig verlangsamt werden. Die Roadmap sollte während der Entwicklung beständig zentral weitergeführt und verwaltet werden.

**Äquivalenzklasse P3: Das Risiko 3.2 ist kritisch und R3.3 ist nicht kritisch** Der Kunde in diesem Projekt möchte eine möglichst genaue Bindung von Umfang, Zeit und Budget ha-

ben. Da R3.3 nicht kritisch ist, ist es für das Projektteam möglich, eine Schätzung mit einem möglichst kleinen Toleranzbereich abzugeben. Das Projektteam sollte sich darauf fokussieren Deadlines zu analysieren. Deadlines können zum Beispiel wichtige Releases oder Integrationspunkte mit anderen Projekten sein. Das Projektteam sollte diese Deadlines in einer Langzeitplanung berücksichtigen und die Features analysieren, die zu einer Deadline fertig sein sollen. Zusätzlich ist es wichtig, dass das Projektteam analysiert, wie viel Zeit sie für die einzelnen Features benötigen. Dies stellt sicher, dass die Deadlines eingehalten werden können. In dieser Planung können gleichzeitig Abhängigkeiten zu anderen Projekten und innerhalb des Projektes berücksichtigt werden. Da R3.3 in diesem Kontext nicht kritisch ist, ist das Projektteam in der Lage diese Planung durchzuführen. Das Projektteam sollte keine detaillierteren Meilensteine verwenden, als für die Risiken von R3.1 und R3.2 nötig sind. Während der Entwicklung sollte das Projektteam diese Roadmap zentral beständig weiterführen und verwalten. Die Arbeit der einzelnen Entwicklerteams in ihren Sprints wird nicht durch die zentrale Roadmap durchgeführt. Die Entwickler planen die Arbeit in ihren Sprints selbständig. Dafür befüllen sie die nötige Anzahl von Sprints (zwei bis drei Sprints) im Voraus.

Besonders wichtig ist, dass das Projektteam flexible Umfänge mit in ihre Planung aufnimmt. Flexible Umfänge können im Notfall verschoben oder sogar komplett fallen gelassen werden. Dies ermöglicht es einem Projektteam, wenn es in Schwierigkeiten gerät, die Anforderungen umzusetzen, die auf jeden Fall umgesetzt werden müssen.

Während der kontinuierlichen Planung sollte der Fokus zum einen auf der Weiterführung der Langzeitplanung für Deadlines liegen und zum anderen auf der Überprüfung des Status. Es ist wichtig, den Status zu überprüfen, um den Stand des Projektes mit der gemachten Planung zu vergleichen. Hier ist es wichtig, früh reagieren zu können, wenn das Projekt Deadlines oder die gemachte Schätzung nicht einhalten kann. Basierend auf der Statusprüfung kann das Projektteam rechtzeitig entscheiden, ob einige der flexiblen Anforderungen verschoben oder fallen gelassen werden können.

**Äquivalenzklasse P4: Das Risiko R3.2 ist kritisch (Existenz von Deadlines) und R3.3 ist kritisch** In diesem Kontext benötigt der Kunde eine Bindung von Zeit und Umfang, da es Deadlines gibt, zu denen einige Anforderungen umgesetzt sein müssen. Gleichzeitig ist das Risiko R3.3 in diesem Kontext kritisch. Dies bedeutet, dass zu Beginn des Projektes eine Unsicherheit über die Anforderungen und/oder verwendete Technologie herrscht.

Das Vorgehen in dieser Äquivalenzklasse ist mit dem aus Äquivalenzklasse P3 bis auf eine Änderung identisch. Das Projektteam kann die beschriebene Langzeitplanung aus Äquivalenzklasse P3 in diesem Kontext nicht verwenden, da die Bekanntheit der Anforderungen und/oder

Technologie zu gering ist. Die Wahrscheinlichkeit ist groß, dass sich Pläne ändern. Um eine größere Sicherheit herzustellen, dass das Projektteam die Deadlines einhalten kann, sollte es Negotiationstechniken und Prototyping während der Anforderungsanalyse zu Beginn des Projektes verwenden. Diese sorgen dafür, dass es eine größere Sicherheit bei den Anforderungen gibt und das Projektteam eine größeres Verständnis für die Umsetzung von Anforderungen gewinnt. Dadurch ist das Projektteam besser in der Lage eine Entwicklung mit Deadlines zu planen. Ein besonderer Fokus, sollte darauf liegen, dass das Projekt flexible Anforderungen enthält.

**Äquivalenzklasse P5: Das Risiko R3.2 ist kritisch (Bindung von Zeit, Umfang und Budget) und R3.3 ist kritisch** In diesem Kontext benötigt der Kunde zu der Bindung von Zeit und Umfang zusätzlich die Bindung des Budgets. Gleichzeitig ist das Risiko R3.3 kritisch. Dies bedeutet, dass die Anforderung und/oder die Technologie unsicher sind. In diesem Kontext sollte das Projektteam nur eine grobe Schätzung des Projektes abgeben und für den Rest des Projektes den Iterationsmodus verwenden. In diesem legt das Projektteam für jeweils eine Iteration den genauen Umfang fest und kann mit dem Kunden zusätzlich einen Vertrag über das Budget für jeweils eine Iteration abschließen. Die kürzere Dauer einer Iteration im Vergleich zum gesamten Projekt ermöglicht es dem Projektteam, verbindliche Aussagen zu treffen. Die grobe Schätzung am Anfang sorgt sowohl für die Softwarefirma als auch für den Kunden für Sicherheit. Zum Beispiel entsteht für den Kunden daraus die Sicherheit, dass die Softwarefirma Ressourcen nicht nur für eine Iteration bereitstellt, sondern für mehrere.

Eine wichtige Voraussetzung für den Iterationsmodus ist, dass die Prozessaktivität für die kontinuierliche Anforderungsanalyse einen kontinuierlichen Schätzprozess für neue Anforderungen enthält. Dieser besteht daraus, dass die Entwicklerteams neue Anforderungen analysieren und eine Schätzung über die Dauer und das Budget für die Umsetzung dieser Anforderungen abgeben. Dies kann zum Beispiel dadurch passieren, dass Teams Lösungskonzepte zu Anforderungen entwickeln. Diese Analyse ist wichtig, damit ein Projektteam verbindliche Aussagen darüber treffen kann, wie viele Anforderungen in einer Iteration umgesetzt werden können und wie teuer einzelne Features sind.

Die Entwicklung in den Iterationen sollte zusätzlich durch eine Roadmap begleitet werden, um das Projekt insgesamt steuern zu können. Zusätzlich sollten Deadlines und andere Integrationspunkte nach der Iterationslänge ausgerichtet werden. Auf diese Weise helfen die Iterationen auch bei der Einhaltung von Deadlines. Die Arbeit der Entwickler in ihren Sprints wird nicht durch die zentrale Roadmap durchgeführt. Die Entwickler planen die Arbeit in ihren Sprints selbstständig. Dafür befüllen sie die nötige Anzahl von Sprints (zwei bis drei Sprints) im

Voraus.

### 8.4.4 Testen

Die Äquivalenzklassen beim Testen entscheiden über die Durchführung der Tests im Sprint, im parallelen Testprozess und die Testaktivität am Ende des Projektes. Wie bereits beschrieben werden in jeder Alternative die Unit Tests von den Entwicklern durchgeführt.

**Äquivalenzklasse T1: R4.1 ist nicht kritisch** In diesem Kontext ist R4.1 nicht kritisch. Das bedeutet, dass weder das Risiko von Fehlern bei der Integration kritisch ist, noch ein großes und/oder verteiltes Team existiert. Dies bedeutet, dass die Entwickler die Integrations- und Systemtests direkt bei der Fertigstellung neuer Features durchführen können.

**Äquivalenzklasse T2: R4.1 ist kritisch und es gibt nur ein oder zwei Entwicklerteams und keine verteilte Entwicklung** Da R4.1 in diesem Kontext kritisch ist, sollte das Projektteam Sorgfalt bei den Integrations- und Systemtests walten lassen. Da gleichzeitig das Projektteam nur aus einem oder zwei Entwicklerteams besteht, ist kein extra Testteam nötig. Stattdessen sollten die Tester mit in das Entwicklerteam integriert werden. Das Projektteam kann eine ausreichende Struktur für diese Tests erreichen, indem sie diese gesammelt am Ende eines Sprints durchführen. Dies bedeutet, dass das Projektteam in diesem Kontext die Testaktivität am Ende des Sprints nutzen sollte (Abbildung 8.2).

**Äquivalenzklasse T3: R4.1 ist kritisch und es gibt ein großes Projektteam und/oder verteilte Entwicklung** Da R4.1 in diesem Kontext kritisch ist, sollte das Projektteam Sorgfalt bei den Integrations- und Systemtests walten lassen. Da es in diesem Kontext viele Teams gibt, die möglicherweise zusätzlich verteilt arbeiten, sollten die Integrations- und Systemtests nicht in den einzelne Entwicklerteams durchgeführt werden. Dies kann sonst zu Chaos bei der Integration führen. Stattdessen sollte das Projektteam ein extra Testteam besitzen, das parallel zur Entwicklung arbeitet. Dies bedeutet, dass das Projektteam die parallele Prozessaktivität fürs Testen nutzen sollte.

**Kontext: Eingebete Software** Wenn eine Software in Verbindung mit einer Hardware entwickelt wird, muss diese auch zusammen mit der Hardware getestet werden. Da Hardware meist nicht iterativ entwickelt wird, kann diese erst am Ende eines Projektes zusammen mit der Software getestet werden. In einem solchen Kontext sollte das Projektteam in den Äquivalenzklassen T1-T3 die Testaktivität am Ende des Projektes nutzen (Abbildung 8.1).

Wenn das Projektteam keine eingebettete Software entwickelt, kann diese Aktivität weggelassen werden.

## 8.5 Koordination

Die Alternativen bei der Koordination entscheiden darüber, welches Framework während der Entwicklung genutzt wird und welche Entscheidungen, die Entwicklerteams treffen dürfen.

**Äquivalenzklasse K1: Projektteam besteht nur aus einem oder zwei Entwicklerteams und ist nicht verteilt** In diesem Kontext besteht das Projektteam nur aus ein oder zwei Entwicklungsteams. Daher benötigt das Projektteam die SAFe Essential Erweiterung im APH-Framework nicht. Absprachen können direkt im oder zwischen den beiden Entwicklerteams getroffen werden. Zudem benötigt das Projekt dadurch keine Projektleitung. Stattdessen können die Planungstätigkeiten vom Product Owner übernommen werden. Das Projektteam kann eigenständige Entscheidungen treffen, so lange nicht die Projektparameter betroffen sind.

**Äquivalenzklasse K2: Großes und/oder verteiltes Projektteam** In diesem Kontext gibt es ein großes und/oder verteiltes Projektteam. Zur Unterstützung der Koordination dieser Teams, sollte das Projekt die Essential SAFe Anteile im Framework nutzen (Abbildung 8.2). Die Entwicklerteams sollten Entscheidungen und Absprachen eigenständig treffen, solange nicht das gesamte Projekt betroffen ist, sie von ihrer Aufgabenstellung abweichen müssen oder die Projektparameter betroffen sind. Darüber hinaus sollte das Projektteam eine horizontale Kommunikation zwischen den Entwicklerteams nutzen, in der sie Absprachen und Entscheidungen eigenständig treffen [92]. Damit zentrale Entscheidungen die Entwicklung nicht zu stark verlangsamen, sollte sich die Projektleitung darauf fokussieren, Entscheidungen schnell treffen zu können. Damit die Planung auf Projektebene und Entwicklerteamebene zusammen greift, sollte die Kommunikation schnell in beide Richtungen funktionieren [32]. Dies bedeutet, dass die Entwicklerteams einen leichten Kontakt zur Projektleitung haben und die Projektleitung schnell antwortet.

Bei einer verteilten Entwicklung sollte es eine zusätzliche Rolle geben, die die Kommunikation zwischen verschiedenen Standorten lenkt und Kommunikationsfehler verhindert.

## 8.6 Unterstützung der Anpassung des APH-Frameworks

Um die Anpassung des APH-Frameworks zu vereinfachen und zu unterstützen, habe ich ein Tool entwickelt, das automatisch bei der Eingabe von Kontextfaktoren die jeweilige Äquivalenzklasse für die Aktivitäten auswählt und das APH-Framework dementsprechend anpasst und darstellt. In Abhängigkeit der Äquivalenzklasse gibt das Tool Handlungsempfehlungen für die jeweilige Prozessaktivität aus. Zusätzlich gibt das Tool dem Nutzer Erklärungen für diese Handlungsempfehlungen aus. Die Entwicklung des Tools wurde von Daniel Morbach [114] und Ole Stender [112] unterstützt.

Dieses Tool kann zum Beispiel von Projektleitern oder Consultants genutzt werden, um einen APH-Entwicklungsprozess für einen Projektkontext entwerfen zu lassen. Für die Bestimmung des Kontextes muss der Nutzer eine Reihe von Fragen beantworten. Diese bestimmen die Werte für die in Kapitel 7 identifizierten Kontextfaktoren, die die Anpassung des APH-Frameworks beeinflussen. In Appendix E sind die Fragen mit den Antwortmöglichkeiten aufgelistet.

Nach der Beantwortung der Fragen berechnet das Tool, welche Risiken in dem eingestellten Kontext kritisch sind und wählt die entsprechenden Äquivalenzklassen für die Aktivitäten aus. Basierend auf diesen Äquivalenzklassen passt das Tool das APH-Framework an und wählt die Aussagen und Erklärungen aus.

Für die Definition der Handlungsempfehlungen habe ich die allgemeinen Beschreibungen für die jeweilige Äquivalenzklasse aus diesem Kapitel, in einzelne Aussagen aufgeteilt. In Appendix E sind die Handlungsempfehlungen und die dazugehörigen Erklärungen für die jeweilige Äquivalenzklasse aufgelistet.

# 9

# Validierung

Das Ziel dieser Dissertation ist es, ein Framework zu entwickeln, dass es Projektteams ermöglicht, eine Balance von agiler und plan-basierter Entwicklung zu erreichen. Zusätzlich werden Empfehlungen bereitgestellt, damit das Framework an einen Projektkontext angepasst werden kann. In den vorherigen Kapiteln wurden die Regeln, Strukturen und Praktiken des APH-Frameworks und die Regeln und Handlungsalternativen für die Anpassung des APH-Frameworks an einen Projektkontext beschrieben.

Für die Erstellung des APH-Frameworks wurde Design Science als Forschungsmethode verwendet. Bei der Verwendung von Design Science muss das entstandene Artefakt evaluiert werden, um zu überprüfen, ob es das Ziel erfüllt. In dieser Arbeit besteht das Artefakt aus dem APH-Framework und den Empfehlungen, die für eine Anpassung des Frameworks an einen Projektkontext getroffen werden. Die Anpassungen bestehen aus den Aussagen und Erklärungen, die in den jeweiligen Äquivalenzklassen getroffen werden (siehe Appendix E). Für die Validierung des Frameworks muss überprüft werden, ob dies zu einem bestimmten Kontext einen passenden Entwicklungsprozess ausgibt, der es einem Projektteam ermöglicht mit den getätigten Aussagen ein Projekt zu entwickeln.

Experten Evaluationen sind ein gängiges Vorgehen in Design Science, um ein Artefakt zu evaluieren [115]. Der Vorteil einer Experten Evaluation ist, dass diese nicht nur das Artefakt bewerten können, sondern auch Feedback zu möglichen Verbesserungen geben können. Unter

Berücksichtigung der Verbesserungen kann die Validität des APH-Frameworks noch gesteigert werden.

## 9.1 Durchführung von Experteninterviews

In dieser Evaluation sollen besonders die Aussagen im Framework in Abhängigkeit eines bestimmten Kontextes evaluiert werden. Um Empfehlungen zu evaluieren, ist die Verwendung von Testfällen eine gängige Methode [116]. Die Testfälle bestehen aus Szenarien für die das entwickelte Tool einen Entwicklungsprozess zusammen mit den entsprechenden Empfehlungen ausgibt. Für die Evaluation wird eine Vergleichsfallstudie mit drei Testszenarien und vier Experten durchgeführt, um das Feedback der Experten miteinander vergleichen zu können [104]. In dieser Fallstudie bewerten die Experten den jeweils erstellten Entwicklungsprozess und die dazugehörigen Empfehlungen und Erklärungen. Zudem geben sie allgemeines Feedback zu dem APH-Framework.

### 9.1.1 Definition von Testfällen

Für die Bewertung der Anpassungen des APH-Frameworks müssen Testfälle definiert werden, anhand derer die Experten den erstellten Entwicklungsprozess bewerten können [117]. Diese Testfälle sind Beispiele für die Anwendungsdomäne von APH-Ansätzen [117]. Um verschiedene Beispiele abzudecken, habe ich drei Szenarien erstellt, die die Testfälle darstellen. Bei der Erstellung der Szenarien wurde darauf geachtet, dass diese möglichst unterschiedliche Kontexte abdecken, um den Einsatz des APH-Frameworks für unterschiedliche Kontexte validieren zu können. Die erstellten Testszenarien für die Experteninterviews sind in Appendix F zu finden.

### 9.1.2 Auswahl von Experten

Als Experten wurden Personen ausgewählt, die Firmen über das Entwicklungsvorgehen in Projekten beraten. Solche Consultants müssen ein Verständnis für den gesamten Entwicklungsprozess haben. Diese bezieht zum Beispiel sowohl Kenntnisse über Requirements Engineering, als auch Architektur oder Testen mit ein. Zudem müssen Consultants ein Verständnis dafür haben, wie sich der Kontext eines Projektes auf den Entwicklungsansatz auswirkt. Diese Eigenschaften machen Consultants, die im Bereich der Prozessanpassung Beratungen durchführen, zu geeigneten Kandidaten für die Expertenevaluation. Bei der Auswahl der Exper-

ten wurde zudem darauf geachtet, dass diese Erfahrungen mit APH-Ansätzen und ein Wissen über sowohl agile als auch plan-basierte Entwicklung haben.

Experte 1 hat seit sechs Jahren Berufserfahrung und berät Zulieferfirmen im Automobilbereich, die mit hybriden Prozessen arbeiten. Der Experte führt zudem Forschungen im Bereich von hybrider Entwicklung durch. Experte 2 hat fünf Jahre Berufserfahrung, ist Product Owner und Agile Coach und berät Firmen darin, agile Entwicklung in traditionellen Projekten zu integrieren. Experte 2 forscht darüber hinaus, wie Firmen mehr Agilität in ihre Prozesse einführen können. Experte 3 hat mehr als drei Jahre Berufserfahrung und berät Firmen im Banken- und Versicherungsbereich. Hier ist der Experte mit der Einführung von Agilität betraut. Durch die Regulierungen in diesen Bereichen werden hier häufig APH-Ansätze verwendet. Experte 4 hat seit 11 Jahren Berufserfahrung als Consultant und berät Firmen in unterschiedlichen Geschäftsbereichen bei der Einführung von Agilität und zur Anpassung an den jeweiligen Kontext.

### 9.1.3 Ausführung

Die Experten haben in getrennten Sitzungen das Framework und die drei Testszenarien bewertet. Jede Sitzung dauerte zwei Stunden und wurde aufgezeichnet. Die Sitzungen waren jeweils in drei Teile aufgeteilt. Im ersten Teil habe ich den Experten erklärt, dass diese Sitzung dazu dient mein Framework und die Anpassungen daran zu evaluieren. Dafür sollen die Experten eine Bewertung abgeben. Dies geschieht anonym und es geht nicht darum die Experten zu bewerten. Im zweiten Teil habe ich den Experten jeweils das Framework selbst vorgestellt. Dabei habe ich basierend auf den Abbildungen 8.1 und 8.2 den Aufbau und die Struktur des Frameworks erklärt. Zusätzlich habe ich beschrieben, welche Tätigkeiten die Projektteams in den jeweiligen Prozessaktivitäten durchführen. Während dieses Teils konnten die Experten Fragen stellen und bereits Anmerkungen machen.

Für den dritten Teil bekamen die Experten die Szenarien und haben die Ausgabe des Tools bewertet. Das Tool gibt zu den Prozessaktivitäten Empfehlungen zum Vorgehen aus und gibt Erklärungen, warum das Projektteam diese Empfehlung umsetzen sollte. Für jedes Szenario haben die Experten die jeweiligen Empfehlungen und die Erklärungen gelesen und bewertet. Dafür haben sie angegeben, bei welchen Aussagen sie zustimmen beziehungsweise ob sie diese Empfehlung zusammen mit dem jeweiligen Kontext auch geben würden. Zusätzlich haben die Experten angegeben, welche Aussagen sie ablehnen und warum sie diese ablehnen. Die Experten konnten zudem Ergänzungen für die Empfehlungen und Erklärungen bei den einzelnen Prozessaktivitäten vornehmen. Diese Ergänzungen stellen wichtige zusätzliche Informationen dar, die ein Projektteam aus der Sicht des Experten für den jeweiligen Kontext

benötigt, um erfolgreich entwickeln zu können. Zusätzlich wurden die Experten aufgefordert zu erklären, warum sie diese Ergänzungen als wichtig erachten. Nachdem die Experten die einzelnen Aussagen bewerten haben, haben sie den gesamten dargestellten Entwicklungsprozess für den jeweiligen Kontext bewertet. Hierbei wurde gefragt, ob sie den dargestellten Entwicklungsprozess für ein Projektteam in dem jeweiligen Kontext als sinnvoll betrachten und mit dem Entwicklungsprozess die agile und plan-basierte Entwicklung passend balanciert wird. Auch hier konnten die Experten Aussagen dazu tätigen, ob einzelne Prozessaktivitäten nicht nötig sind, andere hinzugefügt werden sollten oder Aussagen fehlen. Zu all diesen Anmerkungen wurde auch immer nach Erklärungen der Experten gefragt.

### 9.1.4 Analyse

Das Ziel der Validierung ist es, zu überprüfen, ob das APH-Framework zu einem Kontext eine passende Balance für agile und plan-basierte Entwicklung ausgibt und diese in einen passenden Entwicklungsprozess überführt. Dafür muss überprüft werden, ob die getroffenen Empfehlungen und Erklärungen richtig sind und ob keine Informationen übersehen wurden, die Projektteams gegeben werden müssen. Um die Performance des APH-Frameworks zu überprüfen und gleichzeitig Verbesserungen an dem APH-Framework vorzunehmen, wird überprüft bei wie vielen Aussagen die Experten zustimmen, welche sie ablehnen und welche Informationen die Experten zusätzlich angeben. Dafür wurden die Aussagen der Experten aus den Sitzungen transkribiert. Die Aussagen wurden dann eingeteilt in Aussagen, die die Balance von agiler und plan-basierter Entwicklung betreffen und wichtig für die einzelnen Szenarien sind und in solche, die eine mögliche allgemeine Erweiterung des Frameworks darstellen.

Um bewerten zu können, wie geeignet das APH-Framework ist, wurde in jedem Szenario und zu jedem Experten die Precision, der Recall und der F1-Score für das APH-Framework bestimmt. Eine hohe Eignung des APH-Frameworks ist gegeben, wenn jeweils die Precision und der Recall hoch sind. Dies bedeutet, dass das Tool richtige Aussagen trifft und gleichzeitig keine wichtigen Aussagen übersehen werden, die hätten getroffen werden müssen. Aus der Menge der Aussagen des Frameworks und der Menge der abgelehnten Aussagen des Experten wurde die Precision des APH-Frameworks berechnet. Aus der Menge der richtig getroffenen Aussagen und der zusätzlich gemachten Aussagen des Experten wurde der Recall berechnet. Zusätzlich wurde der F1-Score in jedem Fall berechnet. Der F1-Score setzt die Precision und den Recall in ein Verhältnis. Über den F1-Score kann damit die gesamte Güte des APH-Frameworks bestimmt werden. Je näher der F1-Score an der eins liegt, desto besser ist die Güte des APH-Frameworks.

## 9.2 Resultate

In den folgenden Abschnitten werden die einzelnen Ablehnungen und Ergänzungen von den Experten beschrieben und Verbesserungen am APH-Framework diskutiert. In den Abschnitten für die einzelnen Szenarien werden die Anmerkungen der Experten beschrieben, die sich auf das jeweilige Szenario beziehen. Aussagen, die nicht nur das jeweilige Szenario betreffen, sondern auch andere Kontexte werden am Ende in einem gesonderten Abschnitt diskutiert. In diesem Abschnitt werden zudem die allgemeinen Verbesserungsvorschläge der Experten, die das gesamte Framework betreffen, besprochen.

### 9.2.1 Szenario 1

Für den Entwicklungsprozess in Szenario 1 wurden die Äquivalenzklassen A3, D1, P3, T3 und K2 verwendet (siehe Appendix E). Insgesamt tätigt das Tool dabei 64 Aussagen und Erklärungen. In Tabelle F.1 in Appendix F sind die Aussagen aufgelistet, die die Experten als zusätzliche Informationen gegeben haben und welche Aussagen sie abgelehnt haben.

Die Experten 1-3 haben jeweils die gleiche Kritik zu Aussage A3.1 geäußert. In dieser Aussage wird empfohlen, dass sich ein Projektteam einen *kompletten* Überblick über die zu entwickelnde Anwendung verschaffen sollte. Die Experten stimmen grundlegend zu, dass das Projektteam in Szenario 1 einen möglichst guten Überblick über die Anforderungen gewinnen sollte. Sie kritisieren hier die Verwendung des Wortes *komplett*, da dies suggeriert, dass die Anforderungserhebung zu Beginn des Projektes abgeschlossen ist.

Die Aussage soll ein Projektteam darauf hinweisen, dass es sich einen möglichst vollständigen Überblick über die Anforderungen verschaffen sollte, ohne dabei zu sehr ins Detail zu gehen. Die detaillierten Anforderungen zu genauen Nutzer- und Systemschritten der Anwendungsfälle sollten während der kontinuierlichen Anforderungsanalyse definiert werden. Die Experten stimmen zu, dass es in dem Szenario möglich ist, einen möglichst vollständigen Überblick darüber zu bekommen, welche Anwendungsfälle es gibt. Die Experten lehnen den Kern der Aussage nicht ab, trotzdem wird dies als Ablehnung gezählt. Um die Aussage zu verbessern, wurde sie folgend angepasst:

> A3.1: Das Projektteam sollte sich einen möglichst vollständigen Überblick über die Anforderungen verschaffen. Dies beinhaltet die Definition einer Vision, die Analyse der Anwendungsfällen und deren groben Funktionen auf der Ebene von Epics und die Feststellung und Analyse von Abhängigkeiten. Für einen Überblick der Anforderungen ist es nicht notwendig die genauen Nutzer- und Systemschritte zu untersuchen.

Der Hinweis, dass es bei der Analyse der Anwendungsfälle nur um die groben Funktionen, ohne die detaillierten Nutzer- und Systemschritte, geht, stellt eine Präzisierung bei der Anforderungsanalyse dar und wird daher auch für die anderen Äquivalenzklassen übernommen. Zusätzlich hat Experte 1 für die Anforderungsanalyse zu Beginn des Projektes ergänzt, dass es für eine verteilte Entwicklung von Vorteil ist, ein Kick-off Meeting mit allen Projektteilnehmern zu Beginn des Projektes durchzuführen. Experte 3 hat angegeben, dass die Rolle für die Kommunikation zwischen verschiedenen Standorten bereits zu Beginn eng in die Anforderungsanalyse und Planung involviert werden sollte. Beide Empfehlungen werden für den Fall einer verteilten Entwicklung aufgenommen, da sie unterstützend für die Koordination einer verteilten Entwicklung wirken.

Experte 1 hat ausgesagt, dass die Architekturdefinition zu Beginn eines Projektes die Fähigkeit des Projektteams verbessert, eine Schätzung des Projektes durchzuführen, da das Projektteam mehr Informationen über die Durchführung des Projektes gewinnt. Zudem hat Experte 1 ausgesagt, dass mehr Architekturdefinition zu Beginn den Gesamtaufwand für ein Projekt reduzieren kann. Diese Aussage wird gestützt von Waterman et al. [39] und Islam und Storer [90]. In der verwendeten Äquivalenzklasse D1 kann ein Plan für die Architektur zu Beginn des Projektes erstellt werden. Beide Aussagen von Experte 1 unterstreichen, dass die Erstellung eines Plans einen Vorteil darstellt. Beide Aussagen werden daher als ergänzende Erklärungen für das Bilden eines Architekturplans aufgenommen.

Experte 3 gibt an, dass dem Projektteam eine Frequenz für die Statusüberprüfung des Projektes gegeben werden sollte. Der Experte sagt, dass eine Statusüberprüfung jeweils zu den Sprint Reviews erfolgen sollte, denn eine tägliche Statusüberprüfung wäre eine zu feingranulare Abtastung. Es wird daher die Empfehlung im APH-Framework aufgenommen, dass eine Statusprüfung zu den Sprint Reviews erfolgen sollte.

In Tabelle 9.1 sind die Anzahl der Ablehnungen und Ergänzungen für jeden Experten und die resultierenden Werte für die Precision, den Recall und den F1-Score für das APH-Framework eingetragen. Dadurch dass es nur eine Ablehnung gab und die Experten nur wenige Informationen hinzufügen mussten, ergeben sich in diesem Szenario hohe Werte sowohl für die Precision (min=0,98; max=1) als auch für den Recall (min=0,91; max=0,98). Daraus ergeben sich auch hohe Werte für den F1-Score (min=0,94; max=0,99).

### 9.2.2 Szenario 2

Für den Entwicklungsprozess in Szenario 2 wurden die Äquivalenzklassen A4, D3, P1, T1 und K1 verwendet (siehe Appendix E). Insgesamt besteht das Framework in Szenario 2 aus 51 Aussagen und Erklärungen. In Tabelle F.2 sind die Aussagen aufgelistet, die die Experten als

**Tabelle 9.1:** Anzahl der Ablehnungen und Ergänzungen der Experten und die resultierenden Werte für Precision, Recall und den F1-Score für das APH-Framework in Szenario 1

| | Ablehnungen | Ergänzungen | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| **Experte 1** | 1 | 6 | 0,98 | 0,91 | 0,94 |
| **Experte 2** | 1 | 3 | 0,98 | 0,95 | 0,96 |
| **Experte 3** | 1 | 5 | 0,98 | 0,92 | 0,95 |
| **Experte 4** | 0 | 1 | 1 | 0,98 | 0,99 |
| | | **Durchschnitt** | 0,98 | 0,91 | 0,94 |

zusätzliche Informationen gegeben haben und welche Aussagen sie abgelehnt haben.

In Szenario 2 gab es nur eine Ablehnung. Experte 1 hat für die kontinuierliche Architekturdefinition angegeben, dass keine Rolle benötigt wird, die die Weiterentwicklung auf Projektebene steuert. In diesem Szenario gibt es nur zwei Entwicklerteams. Diese können sich auch direkt über Architekturentscheidungen abstimmen, ohne dass es eine zentrale Rolle gibt. Daher wird für die kontinuierliche Architekturdefinition die folgende Verbesserung vorgenommen:

Wenn ein Risiko von R1.1 - R1.3 kritisch ist, das Projektteam jedoch nur aus ein oder zwei Entwicklerteams besteht, dann sollten die Architekturexperten direkt in die Entwicklerteams integriert werden. Diese können sich direkt abstimmen und der parallele Prozess für die Architekturdefinition kann weggelassen werden. Dies unterstützt eine engere Kommunikation zwischen Entwicklern und Architekten.

Experte 1 und 3 geben an, dass sich das Projektteam bei einer geringen Bekanntheit der Anforderungen (wie in Szenario 2) auf die Endnutzer fokussieren sollte. Experte 3 gibt an, dass das Projektteam Personas einsetzen sollte, um die Erhebung der Anforderungen zu unterstützen. Experte 1 gibt zusätzlich an, dass Feedback während der Entwicklung auch von den Endnutzern eingeholt werden sollte. Dies kann zum Beispiel dadurch geschehen, dass Endnutzer bei den Reviews anwesend sind. Beides sind unterstützende Maßnahmen in Fällen, bei denen die Anforderungen wenig bekannt sind. Daher werden beide Empfehlungen mit aufgenommen. Für eine weitere Unterstützung während der Anforderungsanalyse zu Beginn eines Projektes schlägt Experte 1 vor, dass dort bereits Business Experten eingesetzt werden sollten, um das Erheben von Anforderungen zu unterstützen. Da in diesem Szenario bereits Business Experten eingesetzt werden, wird diese Ergänzung übernommen. In anderen Fällen muss dagegen abgewogen werden, ob der Nutzen den finanziellen Aufwand für eine Team von Business Experten rechtfertigt. Daher wird nur eine Empfehlung ausgesprochen, ein Team von Business Experten zu nutzen, wenn es viele verschiedene Stakeholder beziehungsweise Endnutzer gibt.

Experte 2 gibt an, dass es in dem Szenario wichtig ist, klare Absprachen mit dem Kunden über den Toleranzbereich der Schätzung zu treffen. Dadurch dass die Anforderungen und

die Technologie eine geringe Bekanntheit haben, muss der Toleranzbereich klar abgesprochen werden, damit beim Kunden keine falsche Erwartungshaltung entsteht. Diese falsche Erwartungshaltung kann auch in anderen Kontexten auftreten. Daher wird diese Aussage für alle Äquivalenzklassen als zusätzlicher Hinweis mit eingefügt.

In Tabelle 9.2 sind die Anzahl der Ablehnungen und Ergänzungen für jeden Experten und die resultierenden Werte für die Precision, den Recall und den F1-Score für das APH-Framework eingetragen. Dadurch dass es nur eine Ablehnung gab und die Experten nur wenige Informationen hinzufügen mussten, ergeben sich hohe Werte sowohl für die Precision (min=0,98; max=1) als auch für den Recall (min=0,94; max=0,98). Daraus ergeben sich auch hohe Werte für den F1-Score (min=0,96; max=0,99).

**Tabelle 9.2:** Anzahl der Ablehnungen und Ergänzungen der Experten und die resultierenden Werte für Precision, Recall und den F1-Score für das APH-Framawork in Szenario 2

| | **Ablehnungen** | **Ergänzungen** | **Precision** | **Recall** | **F1 Score** |
|---|---|---|---|---|---|
| **Experte 1** | 1 | 3 | 0,98 | 0,94 | 0,96 |
| **Experte 2** | 0 | 2 | 1 | 0,96 | 0,98 |
| **Experte 3** | 0 | 1 | 1 | 0,98 | 0,99 |
| **Experte 4** | 0 | 1 | 1 | 0,98 | 0,99 |
| | | **Durchschnitt** | 0,99 | 0,96 | 0,98 |

### 9.2.3 Szenario 3

Für den Entwicklungsprozess in Szenario 3 wurden die Äquivalenzklassen A4, D3, P5, T3 und K2 verwendet (siehe Appendix E). Insgesamt besteht das Framework in Szenario 3 aus 59 Aussagen und Erklärungen. In Tabelle F.3 sind die Aussagen aufgelistet, die die Experten als zusätzliche Informationen gegeben haben und welche Aussagen sie abgelehnt haben. In Szenario 3 wurde keine Aussage oder Erklärung von den Experten abgelehnt. Die Experten haben jedoch auch hier wieder Ergänzungen getätigt.

Die Stakeholder sind in dem Szenario vorwiegend zu Beginn des Projektes verfügbar, was die Durchführung von Feedbackschleifen schwierig macht. Gleichzeitig sind die Anforderungen wenig bekannt und die Experten stimmen zu, dass Feedbackzyklen in diesem Szenario wichtig sind. Die Experten 1, 2 und 3 geben daher an, dass das Projektteam am Anfang des Projektes zusammen mit dem Kunden besprechen muss, wie Feedback von Stakeholdern eingesammelt werden kann. Experte 2 gibt an, dass das Projektteam den Kunden und die Stakeholder von der Wichtigkeit von Feedback überzeugen sollte, damit diese Raum dafür schaffen. Nach Experte 3 kann sich das Projektteam auch abschnittsweise auf die relevanten Stakeholder beschränken. Feedback ist in einem Umfeld, in dem die Anforderungen wenig bekannt sind,

besonders wichtig. Daher wird die Aussage hinzugefügt, dass das Projektteam, bei einer geringen Verfügbarkeit der Stakeholder, zusammen mit dem Kunden und den Stakeholdern eine Lösung für die Implementierung von Feedbackzyklen finden sollte.

Experte 2 gibt an, dass das Projektteam in diesem Szenario zusammen mit dem Kunden flexible Anforderungen bei der Definition des Anforderungsumfanges für eine Iteration verwenden sollte. Dies ermöglicht dem Projektteam, auf Probleme während einer Iteration zu reagieren. Zusätzlich gibt der Experte an, dass das Projektteam bei der kontinuierlichen Schätzung jedes Mal eine gewisse Pauschale dazu addieren sollte, damit es Unsicherheiten bei der Entwicklung berücksichtigen kann. Beide beschriebenen Maßnahmen ermöglichen einem Projektteam mehr Flexibilität während den definierten Iterationen in Äquivalenzklasse P5, daher werden beide Empfehlungen für das APH-Framework übernommen.

Die Entwickler arbeiten in diesem Szenario nicht immer zu 100% an dem Projekt. Experte 3 gibt an, dass das Projektteam in diesem Fall zumindest für die einzelnen Iterationen festlegen sollte, wie viele Entwickler zur Verfügung stehen, damit eine genaue Planung möglich ist. Experte 2 gibt zudem an, dass das Projektteam gewarnt werden sollte, dass durch den Wechsel von Entwicklern zwischen unterschiedlichen Projekten die Performance möglicherweise geringer ist. Beide Ergänzungen unterstützen ein Projektteam dabei, erfolgreich einzelne Iterationen zu sammeln und mit einer wechselnden Verfügbarkeit der Entwickler zu planen. Daher werden beide Aussagen als Ergänzungen aufgenommen.

Experte 3 gibt zusätzlich an, dass das Projektteam, durch die geringe Bekanntheit der Anforderungen, der Technologie und der gleichzeitigen Kritikaliät der Software, einen starken Fokus auf das Risikomanagement legen sollte. Risikomanagement und Analyse tragen beide dazu bei, dass Projektteams frühzeitig potenzielle Probleme erkennen und Gegenmaßnahmen einleiten können. Aus diesem Grund wird diese Empfehlung ebenfalls übernommen.

In Tabelle 9.3 sind die Anzahl der Ablehnungen und Ergänzungen für jeden Experten und die resultierenden Werte für die Precision, den Recall und den F1-Score für das APH-Framework in diesem Szenario eingetragen. Dadurch dass es keine Ablehnung gab, liegt die Precision des APH-Frameworks bei jedem Experten bei 1. Auch in diesem Szenario mussten die Experten nur wenige Informationen hinzufügen. Dies sorgt für einen hohen Recall (min=0,94;max=0,98) und insgesamt hohen F1-Score (min=0,96;max=0,99).

### 9.2.4 Allgemeine Anmerkungen

In diesem Abschnitt werden die Ergänzungen der Experten beschrieben und diskutiert, die entweder eine Verbesserung für mehrere Äquivalenzklassen darstellen oder allgemeines Feedback sind.

**Tabelle 9.3:** Anzahl der Ablehnungen und Ergänzungen der Experten und die resultierenden Werte für Precision, Recall und den F1-Score für das APH-Framawork in Szenario 3

| | Ablehnungen | Ergänzungen | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| **Experte 1** | 0 | 2 | 1 | 0,96 | 0,98 |
| **Experte 2** | 0 | 3 | 1 | 0,95 | 0,97 |
| **Experte 3** | 0 | 5 | 1 | 0,92 | 0,95 |
| **Experte 4** | 0 | 1 | 1 | 0,98 | 0,99 |
| | | **Durchschnitt** | 1 | 0,95 | 0,97 |

In Kontexten mit mehreren Entwicklerteams gibt es auch mehrere Product Owner. Die Experten 1 und 3 geben an, dass es in solchen Fällen einen Chief Product Owner geben sollte, der die Gesamtverantwortung für ein Produkt besitzt. Die Rolle eines Chief Product Owners stellt eine Unterstützung bei der Koordination eines Projektes dar. Daher wird diese Rolle für die entsprechenden Kontexte mit in das APH-Framework aufgenommen.

Für die Aussagen zur Anforderungsanalyse zu Beginn eines Projektes hat Experte 4 hinzugefügt, dass es zu Beginn eines Projektes um das *Was* und nicht um das *Wie* gehen sollte. Das Wie sollen die Entwicklerteams während des Projektes selbst bestimmen. Dies stellt eine ergänzende Information dar, die Projektteams dafür sensibilisiert zu Beginn nicht zu detailliert die Anforderungen zu definieren. Daher wurde die Aussage in das APH-Framework eingebaut und für alle Äquivalenzklassen bei der Anforderungsanalyse aufgenommen.

Die Experten 1 und 2 geben an, dass bei der Schätzung zu Beginn des Projektes Entwickler und Architekten mit einbezogen werden sollten und diese nicht allein von der Projektleitung durchgeführt werden sollte. Die Aussagen zu der Schätzung geben an, dass das gesamte Projektteam für die Schätzung verantwortlich ist. Dies bezieht auch die Entwickler und Architekten mit ein. Diese sind besonders wichtig für die Schätzung, da sie die Entwicklungsarbeit leisten. Die Ergänzung der beiden Experten stellt eine Präzisierung der getroffenen Aussagen im APH-Frameworks dar und wird daher mit übernommen.

Experte 4 gibt an, dass bei der Schätzung eines Projektes nicht nur das Budget, die Dauer und der Anforderungsumfang für das Projekt geplant werden sollten. Zusätzlich sollte immer die benötigte Qualität der Software mit berücksichtigt werden. Es ist wichtig, dass ein Projektteam Software in ausreichender Qualität liefert. Daher sollte das Projektteam auch immer gleich die benötigte Qualität der Software zu Beginn des Projektes mit dem Kunden absprechen, um diese mit einzuplanen. Diese Ergänzung wird daher in das APH-Framework mit aufgenommen.

Die Experten 2 und 3 geben an, dass die Durchführung von Feedbackschleifen bei den Tätigkeiten der kontinuierlichen Anforderungsanalyse aufgenommen werden sollte. Experte 3 hat zusätzlich ergänzt, dass die Kommunikation mit Stakeholdern aufgenommen werden sollte. Experte 4 hat angegeben, dass bei der Statusüberprüfung während der kontinuierlichen

Planung zusätzlich explizit mit angegeben werden sollte, dass das Projekt entsprechend dem Projektstatus angepasst werden sollte. Dies beinhaltet zum Beispiel das Fallenlassen von Features. Alle drei Aussagen stellen Präzisierungen von bereits existierenden Aussagen im APH-Framework dar und werden daher übernommen.

Die Experten 2 und 4 geben an, dass das Framework Retrospektiven für die Sprints und Iterationen enthalten sollte. Retrospektiven wurden in der vorgelegten Version des Frameworks weggelassen, um das Wesentliche darzustellen. Trotzdem sind sie ein wichtiger Bestandteil von agiler Entwicklung und können auch in APH-Frameworks eingesetzt werden. Basierend auf dem Feedback der Experten, wurden diese daher zum Framework hinzugefügt.

Experte 1 gibt an, dass bei den parallelen Architekturdefinitionen explizit mit genannt werden sollte, dass die Entwicklerteams eigenständige Archtitekturentscheidungen treffen können, solange diese nicht die Gesamtarchitektur des Projektes betreffen. Dies ist eine Präzisierung der Aussage, dass Entwicklerteams eigenständig Entscheidungen treffen dürfen, solange diese nicht mehrere Teams oder das gesamte Projekt betreffen. Daher wird diese Aussage mit hinzugefügt.

Die Experten 1 und 4 geben als Feedback an, dass das Framework ebenfalls Empfehlungen über die Verwendung von Anforderungsartefakte tätigen könnte. In der Interviewstudie (Kapitel 5) hat sich gezeigt, dass Epics und Storycards geeignete Artefakte in APH-Ansätzen sind. Epics ermöglichen einen Überblick über die Anforderungen ohne bereits zu sehr ins Detail zu gehen. Das Projektteam kann im Verlauf des Projektes die Epics mit Storycards präzisieren und somit die detaillierteren Anforderungen aufnehmen. Die Kombination von Epics und Storycards unterstützt daher die Balance beim Requirements Engineering in APH-Ansätzen. Auch die beiden Experten stimmen dieser Einschätzung zu. Aus diesem Grund wird die Verwendung dieser beiden Anforderungsartefakte in das Framework aufgenommen. Für genauere Empfehlung zur Nutzung auch anderer Artefakte ist jedoch mehr Forschung nötig.

Experte 1 gibt als Feedback an, dass das Framework auch eine Empfehlung zu Vertragsarten zwischen Softwarefirma und Kunde geben könnte. In der Softwareentwicklung sind zwei Vertragsarten vertreten: Dienstverträge und Werkverträge. Bei Dienstverträgen bezahlt ein Kunde die Entwicklung von Software, ohne dass dabei Ergebnisse festgelegt werden müssen. Dadurch ist die Softwarefirma nur zur Erbringung der Entwicklungsarbeit verpflichtet. Bei Werkverträgen hingegen wird das Ergebnis vertraglich festgelegt. Die Softwarefirma ist somit verpflichtet ein bestimmtes Produkt oder Feature zu liefern. Das APH-Framework berücksichtigt bereits im Kontext, ob der Kunde eine Bindung von Anforderungsumfang, Budget, Dauer und Qualität benötigt. In Fällen, in denen der Kunde diese Bindung nicht verlangt, können Firmen Dienstverträge mit dem Kunden nutzen. Diese können abschnittsweise über eine Laufzeit ei-

nes Projektes abgeschlossen werden. Die grobe Schätzung des Projektes zu Beginn stellt eine Sicherheit sowohl für den Kunden als auch die Softwarefirma her. Durch die Schätzung verpflichten sich beide zu einem Projekt, das über mehrere Iterationen geht. Dienstverträge sind besonders in Fällen, in denen die Anforderungen und/oder die Technologie wenig bekannt sind, geeignet, da sie einen leichten Austausch von Anforderungen ermöglichen.

Werkverträge sind dagegen geeignet, wenn der Kunde eine starke Bindung von Anforderungsumfang, Dauer und Budget haben möchte. Hier ist es wichtig, dass das Projekt in mehrere Verträge eingeteilt werden kann (Iterationsmodus bei der Planung). Hoda et al. [38] beschreiben, dass es möglich ist, auch bei Werkverträgen Anforderungen zu definieren, die ausgetauscht oder weggelassen werden können. Dies ist wichtig, um genug Flexibilität auch in Iterationen und bei der Erfüllung von Deadlines zu haben (siehe oben). Ein Projektteam sollte diese Praktik beim Abschluss eines Vertrages berücksichtigen. Die beschriebenen Empfehlungen werden von dem Experten unterstützt und daher für das Framework übernommen. Um detailliertere Empfehlungen zu tätigen oder andere Vertragsmodelle zu empfehlen, bedarf es jedoch weiterer Forschung.

Experte 2 gibt an, dass die Struktur des Projektteams einen Einfluss auf deren Performance und deren Fähigkeit, eine Schätzung abzugeben, hat. Ein Projektteam, das bereits gemeinsame Erfahrung hat, ist performanter und kann eine bessere Schätzung abgeben. Dieser Kontextfaktor wird daher für das APH-Framework mit aufgenommen. Bei einem neu zusammengestellten Projektteam wird die Warnung ausgegeben, dass dies bei der Schätzung zu Beginn des Projektes berücksichtigt werden sollte. In diesem Zusammenhang empfiehlt Experte 3, dass das das Projektteam zu Beginn eines Projektes ein gemeinsames Teamgefühl und einen gemeinsamen Arbeitsmodus herstellt. Dies erweitert die Tätigkeit zur Zusammenstellung des Projektteams bei der Planung. Daher wird diese Empfehlung mit aufgenommen.

## 9.3 Bedrohungen der Validität

Nach Runeson und Höst [104] sollten bei der Durchführung von Fallstudien Konstruktvalidität, Interne und Externe Validität und Verlässlichkeit betrachtet werden.

**Konstruktvalidität** Die Konstruktvalidität berücksichtigt, in wieweit die verwendeten Variablen zu dem passen, was untersucht werden soll. Die Variablen in dieser Fallstudie sind die erstellten Szenarien und die Verwendung von Precision, Recall und des F1-Scores zur Messung der Performance des APH-Frameworks. Die Erstellung der Szenarien stellt hier eine Bedrohung dar. Um diese Bedrohung zu reduzieren, wurden drei Szenarien erstellt, um möglichst viele

unterschiedliche Fälle bei der Analyse des APH-Frameworks abzudecken. Zusätzlich wurde darauf geachtet, dass die Szenarien nicht zu stark in einen klaren agilen oder plan-basierten Kontext liegen. Stattdessen wurde darauf geachtet, dass die Kontexte einen schwierigen Fall für die Balance darstellen.

**Interne Validität** Die Interne Validität berücksichtigt, ob ein beobachteter Faktor von einem weiteren unbekannten Faktor beeinflusst wird. In diesem Fall stellt die Wahl der Experten eine Bedrohung dar. Diese haben insgesamt eine geringe Dauer von Berufserfahrung, haben dafür aber ein hohes Wissen in agiler und hybrider Entwicklung, sowohl in der Theorie (Forschungsergebnisse etc.) als auch in der Praxis.

**Externe Validität** Die Externe Validität berücksichtigt in wieweit die Ergebnisse generalisiert werden können und die Ergebnisse für andere von Interesse sind. Um die Generalisierbarkeit zu erhöhen, wurden die Szenarien von vier Experten bewertet. Durch die hohe Zustimmung der Experten zu dem Framework, betrachte ich dieses als generell valide. Mit dem Framework wurde auch die Wissensbasis validiert. Dadurch sind die erbrachten Ergebnisse auch interessant für andere Personen.

**Verlässlichkeit** Die Verlässlichkeit berücksichtigt in wieweit die Ergebnisse auf der Durchführung der Fallstudie durch einen bestimmten Forscher abhängen. Alle Experteninterviews wurden von mir durchgeführt. Dies stellt eine Bedrohung dar. Um diese Bedrohung zu minimieren, wurde ein genauer Ablauf für alle Interviews entworfen, der sicherstellt, dass alle Experten die gleichen Informationen erhalten haben.

## 9.4 Fazit

Die Experten haben in Szenario 1 und 2 jeweils nur eine Aussage abgelehnt. Zusätzlich haben die Experten die Empfehlung in Szenario 1 nicht grundsätzlich abgelehnt, sondern eine Verbesserung bei der Formulierung vorgeschlagen. Zu der Ablehnung in Szenario 2 hat Experte 2 zusätzlich angegeben, dass es ein Grenzfall ist und das Projektteam auch auf beide Arten die Architektur entwickeln könnte. Aus diesem Grund stufe ich beide Ablehnungen nicht als gravierend ein. Dies bedeutet eine insgesamt hohe Precision (mindestens 0,98) für das APH-Framework.

Neben den beiden Ablehnungen haben die Experten eine Reihe von Vorschlägen für Ergänzungen gemacht. Das APH-Framework ist fokussiert auf die Balance in APH-Ansätzen und

wie diese an einen bestimmten Kontext angepasst werden sollten. Die Ergänzungen sind meist Aussagen oder weitere Erklärungen, die bereits bestehende Aussagen des APH-Frameworks verstärken, präzisieren oder weitere Unterstützung darstellen. Auch die Experten haben für keine der Ergänzungen angegeben, dass diese eine kritische Information für Projektteams darstellt. Daher werden auch die Ergänzungen als nicht gravierend betrachtet, sondern nur unterstützend. Die Experten geben zudem an, dass das Framework, mit den jeweiligen Anpassungen, Aussagen und Erklärungen, für die jeweiligen Kontexte geeignet ist. Die berechneten Werte für den Recall zeigen zudem, dass die Experten jeweils nur wenige Ergänzungen vornehmen mussten. Das Framework hat in jedem Szenario mindestens einen Recall von 0,9. Der F1-Score, der aus der Precision und dem Recall berechnet wird, hat in jedem Szenrario mindestens einen Wert von 0,92. Dies bedeutet eine insgesamt hohe Güte des APH-Frameworks.

Aus diesen Werten und dem Feedback der Experten schließe ich, dass das APH-Framework zum einen valide ist und zum anderen Firmen ermöglicht eine Balance von agiler und planbasierter Entwicklung passend zu ihrem Kontext zu nutzen. Die eingebrachten Ergänzungen der Experten verbessern die Validität des APH-Frameworks zusätzlich.

# 10

# Zusammenfassung und Ausblick

In diesem Abschnitt werden abschließend die Beiträge der Arbeit, die Grenzen der Arbeit und der Ausblick diskutiert.

## 10.1 Beitrag dieser Dissertation

Zu Beginn der Arbeit wurden zwei Forschungslücken identifiziert. (1) Die Erforschung von APH-Ansätzen ist stark auf Fallstudien und Erfahrungsberichte beschränkt und es existiert wenig fallübergreifende Forschung. Fallübergreifende Forschung ist zudem häufig beschränkt auf die Frage, ob bestimmte Methoden angewendet werden, aber nicht wie diese angewendet werden. Entwicklungsprozesse müssen an den jeweiligen Kontext eines Projektes angepasst werden. Es existiert in der Forschung jedoch eine Forschungslücke darüber, wie sich Kontextfaktoren auf einen Entwicklungsprozess auswirken.

(2) Es fehlt ein strukturiertes Vorgehen für die Bildung von APH-Entwicklungsprozessen, durch das Firmen agile und plan-basierte Entwicklung kombinieren und die Vorteile beider Vorgehen nutzen können. Firmen nutzen stattdessen ein evolutionäres Vorgehen für die Erstellung von APH-Entwicklungsprozessen.

Im Folgenden werden die Beiträge dieser Arbeit zur Adressierung dieser beiden Forschungslücken diskutiert.

### 10.1.1 Erforschung der Nutzung von APH-Ansätzen

Die erste Forschungslücke wurde durch eine Mappingstudie und eine Interviewstudie adressiert. In der Mappingstudie wurden Fallstudien und Erfahrungsberichte untersucht, die die Verwendung eines APH-Ansatzes in einer Firma oder Projekt beschreiben.

**Verbindung zwischen Kontext und Entwicklungsprozess** In diesen Fallstudien und Erfahrungsberichten wurde untersucht, welche agilen und plan-basierten Ziele Firmen in APH-Ansätzen verfolgen und durch welche Maßnahmen sie diese Ziele adressieren. Hierbei hat sich gezeigt, dass die Firmen agile und plan-basierte Entwicklung einsetzen, um mit verschiedenen Kontexten umzugehen. Plan-basierte Entwicklung wird zum Beispiel verwendet, um ein großes Projektteam zu koordinieren. Agile Methoden hingegen werden eingesetzt, um mit Änderungen während des Projektes umzugehen. Die Untersuchung der Maßnahmen hat gezeigt, dass diese zu Konflikten im Entwicklungsprozess führen. Das Aufeinandertreffen von unterschiedlichen Zielen sorgt dafür, dass ein Projektteam sowohl plan-basierte als auch agile Methoden für eine Aktivität anwendet. Diese Aktivitäten sind das Requirements Engineering, Architekturdefinition, Planung, Testen und Koordination. Agile und plan-basierte Entwicklung verfolgt jedoch meist gegensätzliche Konzepte bei der Ausführung einer Aktivität. Um die Konflikte zu lösen, müssen die Anwendung von agiler und plan-basierter Entwicklung balanciert werden.

Mit der Analyse der Ziele und Maßnahmen in APH-Ansätzen adressiert diese Arbeit die oben beschriebene Forschungslücke und trägt zum Wissen der Forschungsgemeinde über APH-Ansätze bei. Besonders die identifizierten Konflikte bereiten die Grundlage für weitere Forschung. Sie sensibilisieren Forscher und Praktiker dafür, welche Bereiche von APH-Ansätzen für eine erfolgreiche Entwicklung besonders beachtet werden müssen. Zudem legt diese Forschung die Grundlage für das APH-Framework. Die Erkenntnisse wurden in [3] veröffentlicht und wurden durch das Durchführen einer Interviewstudie mit Praktikern validiert.

**Untersuchung von Entwicklungsprozessen** Die in den Fallstudien und Erfahrungsberichten beschriebenen Entwicklungsprozesse wurden analysiert. Dabei wurden drei grundlegende Modell identifiziert nach denen Firmen in APH-Ansätzen entwickeln: Das Wasserfall-Agil, das Wasserfall-Iteration und das Pipeline Framework. Es wurde untersucht, wie Firmen diese Frameworks anwenden. Dabei wurde beobachtet, dass Firmen diese miteinander kombinieren. Die Kombination dieser Frameworks ermöglicht es Firmen die Aktivitäten sowohl zu Beginn des Projektes als auch während des Projektes durchzuführen (zum Beispiel Anforderungsanalyse).

Die Untersuchung der Entwicklungsprozesse und die drei identifizierten Frameworks liefern

eine Grundlage für die Entwicklung eines allgemeinen Frameworks. Zudem adressiert sie die Forschungslücke über die inneren Strukturen von APH-Ansätzen. Weitere Erforschung der inneren Strukturen von APH-Ansätzen kann darauf aufbauen. Die Kenntnis dieser Frameworks unterstützt Praktiker dabei, ihren eigenen Entwicklungsprozess zu bilden. Die Ergebnisse aus dieser Forschung wurden in [48] veröffentlicht und durch das Durchführen einer Interviewstudie validiert.

**Schwierigkeiten bei der Entwicklung von APH-Ansätzen** In der Mappingstudie wurden die Schwierigkeiten identifiziert, die Firmen bei der Bildung von APH-Ansätzen haben. Es zeigt sich, dass diese besonders Schwierigkeiten haben, die agilen und plan-basierten Ansätze zu balancieren. Die Schwierigkeiten liegen bei der Balance der Anforderungsanalyse, der Architekturdefinition, der Planung, der Dokumentation und der Koordination.

Diese Erkenntnisse erweitern das Wissen der Forschungsgemeinde über die Schwierigkeiten in APH-Ansätzen. Die Schwierigkeit liegt weniger in der Frage welche Methoden ausgewählt werden sollten. Stattdessen liegt sie darin, wie diese richtig angewendet werden müssen, damit eine Firma ihre Ziele erreichen. Diese Arbeit setzt damit weitere Impulse, die Forschung in diese Richtung zu lenken. Weitere Forschung im Bereich von APH-Ansätzen sollte sich weiter mit der Erforschung und Verbesserung der Balance beschäftigen und nicht nur darauf fokussieren, ob bestimmte Methoden angewendet werden. Die Ergebnisse dieser Forschung wurden in [3] veröffentlicht.

**Untersuchung der Balance von agiler und plan-basierter Entwicklung** Im Rahmen einer Interviewstudie wurde untersucht, wie die Firmen in den identifizierten Aktivitäten eine Balance von agiler und plan-basierter Entwicklung erreichen. Diese Untersuchung hat wichtige Indikatoren erbracht nach denen Firmen entscheiden, wie viel plan-basierte Entwicklung sie anwenden müssen und wie viel agile Entwicklung sie anwenden können. Diese Abwägung ist in Projekten wichtig, damit das Projektteam eine ausreichende Stabilität hat und gleichzeitig so viel Flexibilität wie möglich erreicht.

Diese Forschung leistet einen Beitrag zur Unterstützung von Firmen, die mit der Balance von agiler und plan-basierter Entwicklung Schwierigkeiten haben. Sie liefert zudem die Grundlage für weitere Forschung im Bereich der Balance von APH-Ansätzen. Die Erkenntnisse aus der Interviewstudie wurden in [4] veröffentlicht.

### 10.1.2 Entwicklung eines allgemeinen Frameworks für APH-Ansätze

Zur Adressierung der zweiten Forschungslücke wurde ein allgmeines Konzept für die Balance von agiler und plan-basierter Entwicklung geschaffen. Aus dem Konzept wurde anschließend Prinzipien und ein allgemeines Framework für APH-Ansätze entwickelt.

**Konzept für die Balance von agiler und plan-basierter Entwicklung** Die Erforschung der Balance von APH-Ansätzen hat gezeigt, dass dabei Firmen Stabilität und Flexibilität gegeneinander abwägen. Ich habe daher ein risikobasiertes Konzept zur Abwägung der Risiken von zu wenig Stabilität und zu wenig Flexibilität entwickelt. Wie kritisch diese Risiken sind, wird bestimmt durch den Stand an Struktur, den ein Projektteam erreicht, und dem Kontext, in dem sich ein Projekt befindet. Der erreichte Stand der Struktur ist wichtiger Teil des Konzeptes, da es nicht mit abstrakten Größen arbeitet, wie zum Beispiel der aufgewendete Aufwand für eine Aktivität. Aus dem Stand der Struktur lassen sich konkrete Handlungsempfehlungen für Projektteams ableiten. Dies ist bei einer abstrakten Größe, wie dem allgemeinen Aufwand, schwieriger.

Das Konzept bietet eine Grundlage für die Erstellung eines allgemeinen Frameworks, in dem eine passende Balance zu einem Kontext erreicht wird. Das Konzeptionieren der Balance unterstützt Praktiker dabei die Kräfte, die in APH-Ansätzen wirken, besser zu verstehen.

**Prinzipien für die Balance von agiler und plan-basierter Entwicklung** Auf Basis der Daten aus der Mapping- und Interviewstudie habe ich die Risiken identifiziert, die bei der Balance von zu wenig Stabilität und zu wenig Flexibilität in den einzelnen Aktivitäten abgewogen werden müssen. Durch die Anwendung des Konzeptes habe ich daraus Prinzipien für die Balance von agiler und plan-basierter Entwicklung abgeleitet.

Diese Prinzipien leisten einen Beitrag für die Unterstützung von Firmen bei der Balance von agiler und plan-basierter Entwicklung. Firmen können zum Beispiel anhand der Prinzipien entscheiden, wann sie die Erhebung der Anforderungen zu Beginn des Projektes abbrechen und in eine kontinuierliche Anforderungsanalyse übergehen. Auf diese Weise geben sie Projektteams eine Richtlinie bei der Anwendung von APH-Ansätzen.

**Entwicklung und Validierung eines allgemeinen Frameworks für APH-Ansätze** Aus den gesammelten Erkenntnissen wurde ein allgemeines Framework für APH-Ansätze entwickelt. Dieses Framework ermöglicht es einem Projektteam eine Balance von sowohl Stabilität als auch Flexibilität in ihrem Entwicklungsprozess zu erreichen. Zusätzlich zu dem allgemeinen

Framework wurden anhand des Konzeptes und der identifizierten Risiken Handlungsalternativen definiert, anhand derer das APH-Framework an einen Projektkontext angepasst werden kann. Das Framework wurde anschließend durch Experteninterviews validiert.

Dieses Framework adressiert, dass es kein strukturiertes Vorgehen für die Kombination von agiler und plan-basierter Entwicklung gibt und stellt auf diese Weise eine Unterstützung für Projektteams dar. Durch die Handlungsempfehlungen im Framework unterstützen Firmen dabei ausreichend Stabilität in ihrem Entwicklungsprozess zu erreichen und gleichzeitig so flexibel, wie möglich zu sein. Das Framework unterstützt Firmen auf diese Weise die Firmen bei den Schwierigkeiten, die sie bei der Kombination von agiler und plan-basierter Entwicklung haben.

## 10.2 Grenzen der Arbeit und Ausblick

Im Folgenden werden Grenzen dieser Arbeit und weitere Forschungsrichtungen diskutiert.

**Anwendung des APH-Frameworks in der Industrie** Das APH-Framework wurde von Experten im Bereich von hybrider Entwicklung validiert. Es wurde jedoch noch nicht überprüft, ob die Anwendung des Frameworks eine Verbesserung in einem Projekt herbeiführt beziehungsweise einen Vorteil gegenüber einem evolutionär gebildeten Entwicklungsprozess hat.

Daher sollte das Framework in zukünftiger Forschung in verschiedenen Entwicklungsprojekten angewendet werden, um einen tatsächlichen Nutzen zu untersuchen und zu identifizieren.

**Erweiterung der Wissensbasis** Diese Arbeit leistet einen Beitrag zur Erforschung der Nutzung von APH-Ansätzen. Die Erkenntnisse stammen aus einer Literatur- und einer Interviewstudie. Weitere Analysen, von unterschiedlichen APH-Ansätzen, sollten durchgeführt werden, um genauere Anpassungen des APH-Frameworks an einen Projektkontext zu ermöglichen. Die Wichtigkeit der Erforschung von hybriden Entwicklungsansätzen hat durch die Helena Studie [7] eine neue Bedeutung bekommen. Hier ist zu erwarten, dass sich in Zukunft weitere Erkenntnisse ergeben. Zukünftige Forschung sollte immer mit einbeziehen, wie Methoden und Praktiken angewendet werden und nicht nur ob sie angewendet werden. Um die Ergebnisse weiter zu generalisieren, könnte eine Studie, ähnlich der Helena Studie, über die erreichten Balancen in Projekten durchgeführt werden. Dies würde möglicherweise weitere Möglichkeiten der Balance aufdecken.

Ein weiterer Schritt für die Erweiterung der Wissensbasis ist die Verbindung der Helena Daten und des APH-Frameworks. Aus den Helena Daten kann entnommen werden, welche Methoden und Praktiken Firmen anwenden, bei einer unterschiedlichen Ausprägung von agiler

und plan-basierter Entwicklung anwenden [31]. Daraus können die Handlungsempfehlungen des APH-Frameworks verbessert werden, indem weitere unterstützende Praktiken bei einer Balance vorgeschlagen werden. Dies ist Bestandteil aktueller Forschung.

Diese Arbeit hat untersucht, welche Balance von Firmen erreicht werden sollte und wie sie diese bestimmen können. Zur weiteren Unterstützung von Projektteams, sollte untersucht werden, welche Werkzeuge sie beim Erreichen einer Balance unterstützen würden. Dies können zum Beispiel bestimmte Checklisten oder ähnliches sein.

**Verbesserungen des APH-Frameworks** Das APH-Framework ist fokussiert auf die wesentlichen Punkte, die für eine Balance von agiler und plan-basierter Entwicklung notwendig sind. Daher gibt es Raum für Erweiterungen, die unter anderem während der Validierung identifiziert wurden.

Eine Erweiterung des APH-Frameworks betrifft Empfehlungen darüber, welche Anforderungsartefakte ein Projektteam für die Anforderungsanalyse verwenden sollte. Ein erster Schritt dafür wäre die Verbindung der beschriebenen Risiken aus dieser Arbeit mit den beschriebenen Risiken von Boruszewski [107]. Sie beschreibt die Risiken, die bei der Verwendung von unterschiedlichen Anforderungsartefakten abgewogen werden sollten. Sie berücksichtigt dabei jedoch nur die Anwendung oder das Weglassen von Artefakten und nicht wie diese Artefakte angewendet werden sollten. Daher ist dort mehr Forschung nötig, um zu einer Balance passende Empfehlungen auszusprechen.

Die Handlungsempfehlungen des APH-Frameworks gehen nicht darauf ein, wie das APH-Framework am besten in einem Projektteam eingeführt werden kann. Softwareentwicklung hängt stark von den Personen ab, die sie durchführen. Daher sollte als Erweiterung des Frameworks die Perspektive von Entwicklern miteinbezogen werden und Schwierigkeiten bei der Einführung des APH-Frameworks untersucht werden. Es sollte untersucht werden, auf welche Weise das APH-Framework möglicherweise angepasst werden muss, um mit den Personen und den Werten in einem Projektteam kompatibel zu sein.

## 10.3 Fazit

Die Nutzung von APH-Ansätzen ist in der Industrie weit verbreitet. Die Forschung in dieser Arbeit hat gezeigt, dass Firmen agile und plan-basierte Entwicklung kombinieren, da sie sowohl Flexibilität als auch Stabilität in ihrem Entwicklungsprozess benötigen. Die Stabilität ermöglicht es Firmen, große Entwicklungsprojekte zu koordinieren oder sicherheitskritische Anwendungen zu entwickeln. Die Flexibilität ermöglicht es Firmen auf Änderungen zu reagieren. Die

Firmen haben bei der Bildung von APH-Entwicklungsprozessen besonders Schwierigkeiten dabei, eine Balance von agiler und plan-basierter Entwicklung zu erreichen. Gleichzeitig fehlt ein strukturiertes Vorgehen, wie sie diese Balance erreichen und einen Entwicklungsprozess bilden können.

In dieser Arbeit wurde daher zum einen untersucht, wie Firmen eine Balance von agiler und plan-basierter Entwicklung erreichen und wie diese in einen Entwicklungsprozess überführt werden kann. Zum anderen wurde ein allgemeines Entwicklungsframework für APH-Ansätze geschaffen, das zusätzlich von Projektteams an einen Projektkontext angepasst werden kann und dafür Empfehlungen ausspricht. Dieses Framework wurde durch Experten validiert und weiter verbessert. Diese Arbeit leistet durch die Entwicklung des Frameworks eine direkte Unterstützung für Firmen, da das Framework genaue Handlungsempfehlungen ausspricht. Zusätzlich liefert diese Arbeit durch das Konzept und die Prinzipien für die Balance die Grundlage für Firmen, damit diese die Balance von agiler und plan-basierter Entwicklung eigenständig für ihre Firmen weiterentwickeln können.

# A

# Die Daten der Systematischen Mappingstudie

## A.1 Daten des ersten Suchdurchlaufes

### A.1.1 Suchstring

*“hybrid software development”* ***OR*** *(agile* ***AND*** *(((“traditional software development”* ***OR*** *plan-based)* ***AND*** *(integrate* ***OR*** *combine* ***OR*** *balance))* ***OR*** *large-scale* ***OR*** *“safety critical”))*

### A.1.2 Der Filterungsprozess während des ersten Suchdurchlaufes

Der Filterungsprozess für den ersten Suchdurchlauf ist in Abbildung A.1 dargestellt.

### A.1.3 Referenzliste des ersten Suchdurchlaufes

#### Startset

1. Abdelaziz, A. A., El-Tahir, Y., Osman, R. (2015, September). Adaptive Software Development for developing safety critical software. In 2015 International Conference on Compu-

ting, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE) (pp. 41-46). IEEE.

2. Adelakun, O., Garcia, R., Tabaka, T., Ismail, R. (2017). Hybrid project management: Agile with discipline. In International Conference on Information Resources Management (CONF-IRM). Association For Information Systems.

3. Anitha, P. C., Savio, D., Mani, V. S. (2013, July). Managing requirements volatility while "Scrumming" within the V-Model. In 2013 3rd International Workshop on Empirical Requirements Engineering (EmpiRE) (pp. 17-23). IEEE.

4. Batra, D., Xia, W., VanderMeer, D., Dutta, K. (2010). Balancing agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry. Communications of the Association for Information Systems, 27(1), 21.

5. Bick, S., Spohrer, K., Hoda, R., Scheerer, A., Heinzl, A. (2017). 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.

6. Binder, J., Aillaud, L. I., Schilli, L. (2014). The project management cocktail model: An approach for balancing agile and ISO 21500. Procedia-Social and Behavioral Sciences, 119, 182-191.

7. Cao, L., Mohan, K., Xu, P., Ramesh, B. (2004, January). How extreme does extreme programming have to be? Adapting XP practices to large-scale projects. In 37th Annual Hawaii International Conference on System Sciences, 2004. (pp. 10-pp). IEEE.

8. Hayata, T., Han, J. (2011, July). A hybrid model for IT project with Scrum. In Proceedings of 2011 IEEE International Conference on Service Operations, Logistics and Informatics (pp. 285-290). IEEE.

9. Heeager, L. T., Nielsen, P. A. (2009). Agile software development and its compatibility with a document-driven approach? A case study. In 20th Australasian Conference on Information Systems Compatibility of Agile and Document-Driven Approaches 205-214.

10. Heeager, L. T. (2012). Introducing agile practices in a documentation-driven software development practice: a case study. Journal of Information Technology Case and Application Research, 14(1), 3-24.

11. Heeager, L. T., Nielsen, P. A. (2020). Meshing agile and plan-driven development in safety-critical software: a case study. Empirical Software Engineering, 25(2), 1035-1062.

12. Heidenberg, J., Matinlassi, M., Pikkarainen, M., Hirkman, P., Partanen, J. (2010, June). Systematic piloting of agile methods in the large: two cases in embedded systems development. In International Conference on Product Focused Software Process Improvement (pp. 47-61). Springer, Berlin, Heidelberg.

13. Heikkilä, V. T., Paasivaara, M., Lasssenius, C., Damian, D., Engblom, C. (2017). Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. Empirical Software Engineering, 22(6), 2892-2936.

14. Imani, T., Nakano, M., Anantatmula, V. (2017). Does a hybrid approach of agile and plan-driven methods work better for IT system development projects. International journal of engineering research and applications, 1(2), 3.

15. Kautz, K., Madsen, S. (2010). Understanding Agile Software Development in Practice. International Conference on Information Resources Management.

16. Laux, I., Kranz, J. (2019). Coexisting Plan-driven and Agile Methods: How Tensions Emerge and Are Resolved. International Conference on Information Systems (ICIS).

17. McHugh, M., McCaffery, F., Casey, V. (2014). Adopting agile practices when developing software for use in the medical domain. Journal of Software: Evolution and Process, 26(5), 504-512.

18. Matkovic, P., Maric, M., Tumbas, P., Sakal, M. (2018). Traditionalisation of agile processes: architectural aspects. Computer Science and Information Systems, 15(1), 79-109.

19. Neto, F. G. D. O., Horkoff, J., Knauss, E., Kasauli, R., Liebel, G. (2017, September). 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) (pp. 315-322). IEEE.

20. Pechau, J. (2011, July). Rafting the agile waterfall: value based conflicts of agile software development. In Proceedings of the 16th European Conference on Pattern Languages of Programs (pp. 1-15).

21. Portela, L. T., Borrego, G. (2016, August). Scrumconix: Agile and documented method to AGSD. In 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE) (pp. 195-196). IEEE.

22. Qureshi, M. R. J. (2012). Agile software development methodology for medium and large projects. IET software, 6(4), 358-363.

23. Ramesh, B., Mohan, K., Cao, L. (2012). Ambidexterity in agile distributed development: An empirical investigation. Information systems research, 23(2), 323-339.

24. Read, A., Briggs, R. O. (2012, January). The many lives of an agile story: Design processes, design products, and understandings in a large-scale agile development project. In 2012 45th Hawaii International Conference on System Sciences (pp. 5319-5328). IEEE.

25. Paige, R. F., Charalambous, R., Ge, X., Brooke, P. J. (2008, September). Towards agile engineering of high-integrity systems. In International Conference on Computer Safety, Reliability, and Security (pp. 30-43). Springer, Berlin, Heidelberg.

26. Tjørnehøj, A. (2018). The role of the IT-Project Manager in Organizations that Balance Agile and Traditional Software Development. In IRIS: Selected Papers of the Information Systems Research Seminar in Scandinavia (No. 9). Scandinavian Chapter of the Association for Information Systems (AIS)-Scandinavian IRIS.

27. Uludağ, Ö., Kleehaus, M., Xu, X., Matthes, F. (2017, October). Investigating the role of architects in scaling agile frameworks. In 2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC) (pp. 123-132). IEEE.

28. Van Waardenburg, G., Van Vliet, H. (2013). When agile meets the enterprise. Information and software technology, 55(12), 2154-2171.

29. Waterman, M., Noble, J., Allan, G. (2015, May). How much up-front? A grounded theory of agile architecture. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (Vol. 1, pp. 347-357). IEEE.

30. Xu, P. (2009). Coordination in large agile projects. Review of Business Information Systems (RBIS), 13(4).

31. Islam, G., Storer, T. (2020). A case study of agile software development for safety-Critical systems projects. Reliability Engineering and System Safety, 200, 106954.

**1. Iteration**

1. Scheerer, A., Hildenbrand, T., Kude, T. (2014, January). Coordination in large-scale agile software development: A multiteam systems perspective. In 2014 47th Hawaii international conference on system sciences (pp. 4780-4788). IEEE.

2. Dingsøyr, T., Moe, N. B., Fægri, T. E., Seim, E. A. (2018). Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. Empirical Software Engineering, 23(1), 490-520.

3. Waterman, M., Noble, J., Allan, G. (2013, June). The effect of complexity and value on architecture planning in agile software development. In International Conference on Agile Software Development (pp. 238-252). Springer, Berlin, Heidelberg.

4. Rottier, P. A., Rodrigues, V. (2008, August). Agile development in a medical device company. In Agile 2008 Conference (pp. 218-223). IEEE.

## A.2 Suchdurchlauf mit dem zweiten Suchstring

### A.2.1 Suchstring des zweiten Suchdurchlaufes

*("hybrid software development"* ***OR*** *(agile* ***AND*** *(((“traditional software development"* ***OR*** *plan-based)* ***AND*** *(integrate* ***OR*** *combine* ***OR*** *balance))* ***OR*** *large-scale* ***OR*** *“safety critical"))) **AND** (approach **OR** framework **OR** process)*

### A.2.2 Der Filterungsprozess während des zweiten Suchdurchlaufes

Der Filterungsprozess für den zweiten Suchdurchlauf ist in Abbildung A.2 dargestellt.

### A.2.3 Referenzliste des zweiten Suchdurchlaufes

#### Startset

1. McHugh, M., McCaffery, F., Coady, G. (2014, November). An agile implementation within a medical device software organisation. In International Conference on Software Process Improvement and Capability Determination (pp. 190-201). Springer, Cham.

2. Shimoda, A., Yaguchi, K. (2017, July). A method of setting the order of user story development of an agile-waterfall hybrid method by focusing on common objects. In 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI) (pp. 301-306). IEEE.

3. Rong, G., Shao, D., Zhang, H. (2010, November). Scrum-PSP: Embracing process agility and discipline. In 2010 Asia Pacific Software Engineering Conference (pp. 316-325). IEEE.

4. Heikkilä, V. T., Paasivaara, M., Lasssenius, C., Damian, D., Engblom, C. (2017). Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. Empirical Software Engineering, 22(6), 2892-2936.

5. Heikkilä, V. T., Paasivaara, M., Lassenius, C., Engblom, C. (2013, June). Continuous release planning in a large-scale scrum development organization at ericsson. In International conference on agile software development (pp. 195-209). Springer, Berlin, Heidelberg.

6. Millard, W. David, Johnson, Daniel M., Henderson, John M., Lombardo, Nicholas J., Bass, Robert B., and Smith, Jason E. (2014). Embedding Agile Practices within a Plan-Driven Hierarchical Project Life Cycle. In INCOSE International Symposium (Vol. 24, No. 1, pp. 745-758).

7. Takahira, R. Y., Laraia, L. R., Dias, F. A., Abraham, S. Y., Nascimento, P. T., Camargo, A. S. (2014, July). Scrum and Embedded Software development for the automotive industry. In Proceedings of PICMET'14 Conference: Portland International Center for Management of Engineering and Technology; Infrastructure and Service Integration (pp. 2664-2672). IEEE.

8. Paige, R. F., Charalambous, R., Ge, X., Brooke, P. J. (2008, September). Towards agile engineering of high-integrity systems. In International Conference on Computer Safety, Reliability, and Security (pp. 30-43). Springer, Berlin, Heidelberg.

9. Read, A., Briggs, R. O. (2012, January). The many lives of an agile story: Design processes, design products, and understandings in a large-scale agile development project. In 2012 45th Hawaii International Conference on System Sciences (pp. 5319-5328). IEEE.

10. Taxén, L., Pettersson, U. (2010). Agile and Incremental Development of Large Systems. In 7th European Systems Engineering Conference (EuSEC 2010), Stockholm, Sweden, May 23–26, 2010.

11. Tanveer, M. (2015, December). Agile for large scale projects—A hybrid approach. In 2015 National Software Engineering Conference (NSEC) (pp. 14-18). IEEE.

12. Smojver, K., Belani, H., Car, Z. (2009, June). Building a hybrid process model for a complex software system integration. In 2009 10th International Conference on Telecommunications (pp. 147-153). IEEE.

13. Ahmad, G., Soomro, T. R., Brohi, M. N. (2014). XSR: novel hybrid software development model (integrating XP, scrum and RUP). International Journal of Soft Computing and Engineering (IJSCE), 2(3), 126-130.

14. Cho, J. (2009). A hybrid software development method for large-scale projects: rational unified process with scrum. Issues in Information Systems, 10(2), 340-348.

15. Zaki, K. M., Moawad, R. (2010, March). A hybrid disciplined Agile software process model. In 2010 The 7th International Conference on Informatics and Systems (INFOS) (pp. 1-8). IEEE.

16. Sharma, N., Wadhwa, M. (2015). exsrup: Hybrid software development model integrating extreme programing, scrum and rational unified process. TELKOMNIKA Indonesian Journal of Electrical Engineering, 16(2), 377-388.

17. Sultana, S., Motla, Y. H., Asghar, S., Jamal, M., Azad, R. (2014, February). A hybrid model by integrating agile practices for Pakistani software industry. In 2014 International Conference on Electronics, Communications and Computers (CONIELECOMP) (pp. 256-262). IEEE.

18. Hayata, T., Han, J. (2011, July). A hybrid model for IT project with Scrum. In Proceedings of 2011 IEEE International Conference on Service Operations, Logistics and Informatics (pp. 285-290). IEEE.

**1. Iteration**

1. Bose, L., Thakur, S. (2013). Introducing Agile into a Non Agile Project: Analysis of Agile Methodology with its Issues and Challenges. International Journal of Advanced Research in Computer Science, 4(1).

2. Heeager, L. T., Nielsen, P. A. (2009). Agile software development and its compatibility with a document-driven approach? A case study. In 20th Australasian Conference on Information Systems Compatibility of Agile and Document-Driven Approaches, Melbourne.

3. Lesmana, I. P. D., Karimah, R. N., Widiawan, B. (2016, November). Agile-Waterfall hybrid for prevention information system of dengue viral infections: A case study in Health Department of Jember, East Java, Indonesia. In 2016 14th International Conference on ICT and Knowledge Engineering (ICT&KE) (pp. 1-6). IEEE.

4. Lozo, G., Jovanovic, S. (2012). A flexible hybrid method for IT project management. Journal of Emerging Trends in Computing and Information Sciences, 3(7), 1027-1036.

5. Portela, L. T., Borrego, G. (2016, August). Scrumconix: Agile and documented method to AGSD. In 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE) (pp. 195-196). IEEE.

6. Nisa, S. U., Qureshi, M. R. J. (2012). Empirical estimation of hybrid model: A controlled case study. IJ Information Technology and Computer Science, 4(8), 43-50.

## A.3 Auswertung der Daten aus der Literaturstudie

Die Auswertung der Daten aus der Literaturstudie finden sich in den Tabellen A.1, A.2, A.3, A.4 und A.5.

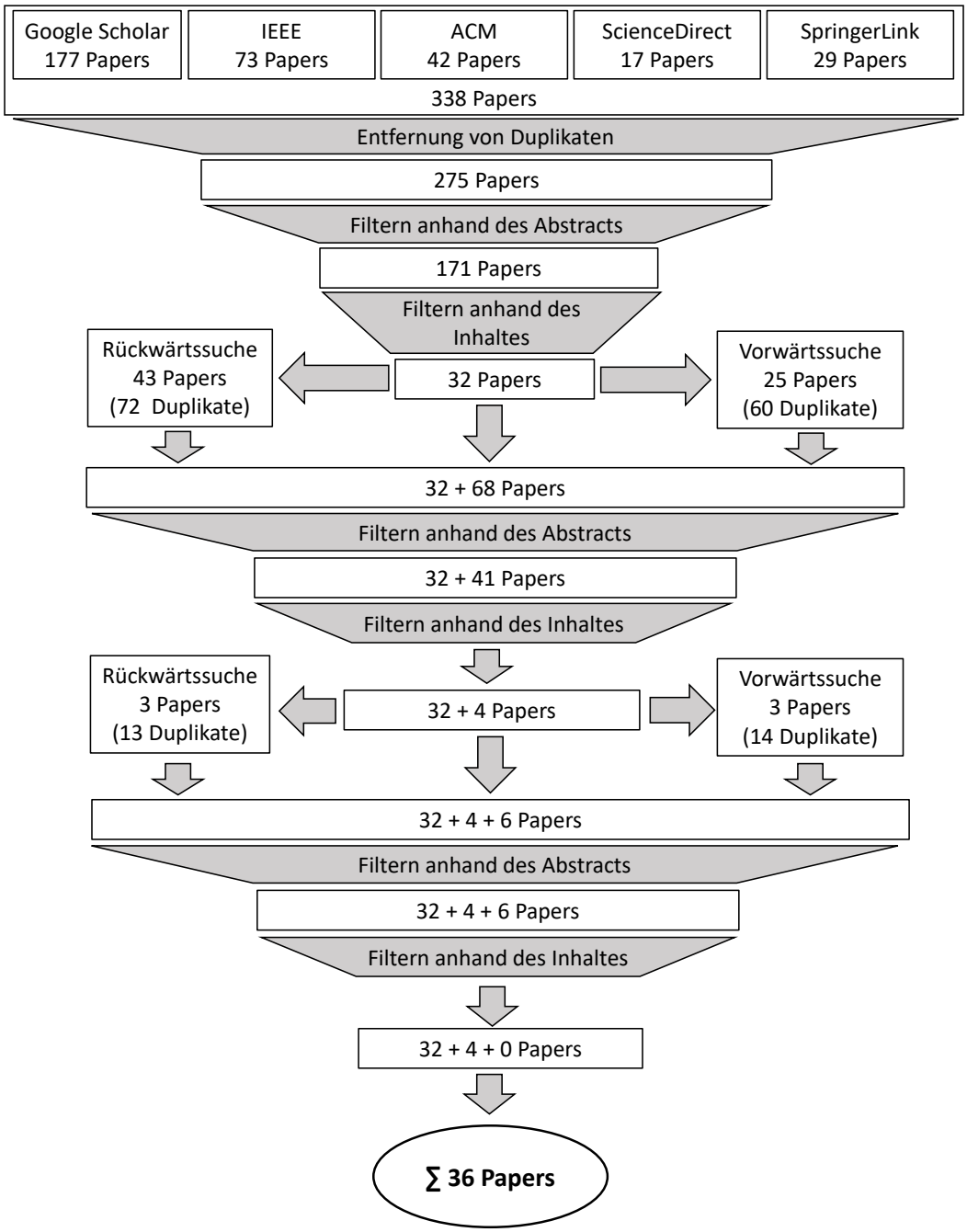


**Abbildung A.1:** Repräsentation des Such- und Filterungsprozesses mit dem ersten Suchstring nach einer Darstellung von Klünder et al. [118]

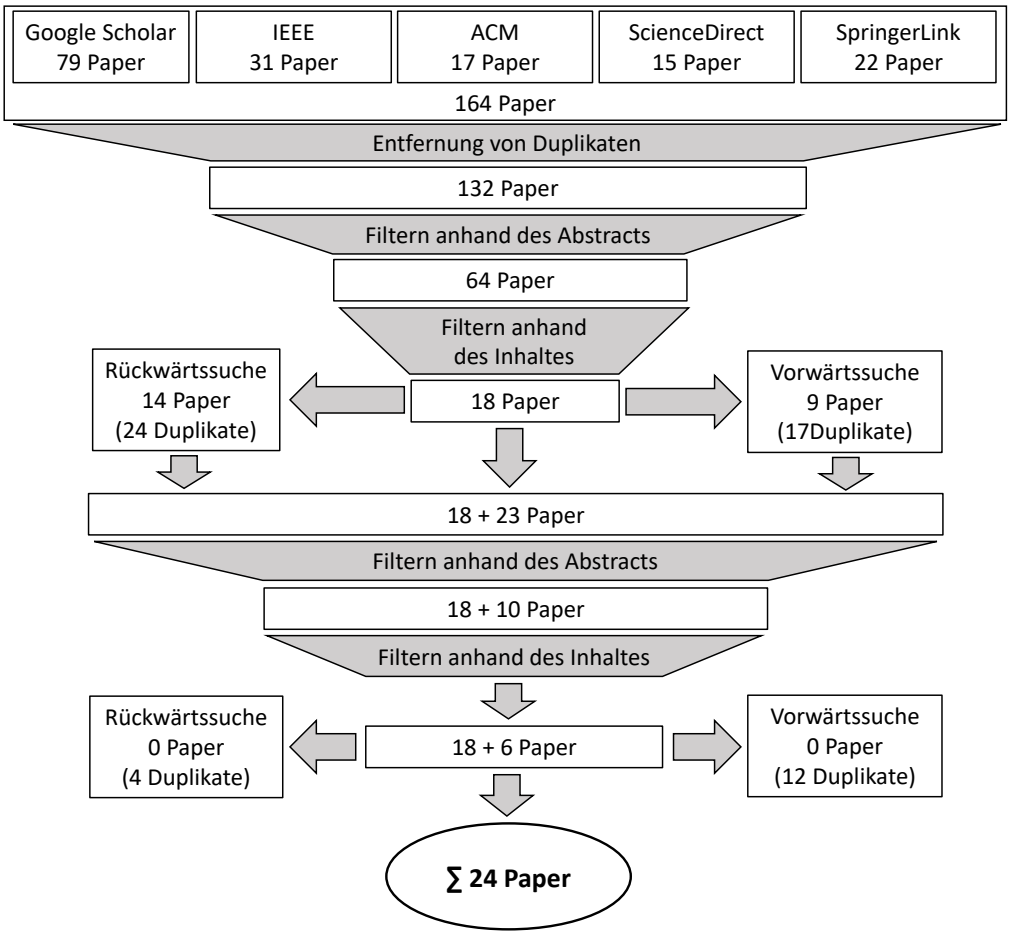


**Abbildung A.2:** Repräsentation des Such- und Filterungsprozesses mit dem zweiten Suchstring nach einer Darstellung von Klünder et al. [118]

| Publikation | Geschäftsbereich | Kontext | Nutzung der Frameworks und deren Aktivitäten | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | WAF | | | | | | WIF | | | | | PF | | | | | |
| | | | Anf. | Arch. | Back. | Entw. | Test. | Oper. | Anf. | Arch. | Entw. | Test. | Oper. | Anf. | Arch. | Entw. | Test. | Oper. |
| Zaki and Mohawad [59] | N/A | N/A | X | X | | X (I) | X | X | | | | | | | | | | | |
| Millard et al. [60] | Verteidigungssysteme | N/A | X | X | | X | X (S) | X | | | | | | | | | | | |
| Ahmad et al [74] | N/A | N/A | X | X | | X | N/A | X | | | | | | | | | | | |
| Cho [42] | N/A | N/A | X | X | | X (T) | | X | | | | | | | | | | | |
| Lesmana et al. [61] | Medizin | N/A | X | X | X | X (I) | X (A) | X | | | | | | | | | | | |
| Lozo et al. [63] | Finanzsysteme | N/A | X | X | | X (U) | X (I, S) | X | | | | | | | | | | | |
| Portela and Borego [70] | N/A | Verteilte Entwicklung | X | | | X (T) | | N/A | | | | | | | | | | | |
| Hayata and Han [87] | N/A | N/A | X | X | | X (U) | X (S) | N/A | | | | | | | | | | | |
| Shimoda and Yanguchi [71] | Informationssysteme | N/A | X | X | | X | X (S) | N/A | | | | | | | | | | | |
| Takahira et al. [119] | Automobil | Eingebettete Entwicklung | X | | | X | X (U, I, S) | N/A | | | | | | | | | | | |
| McHugh et al. [44] | Medizin | N/A | X | X | | X (U) | X (I, S) | X | | | | | | | | | | | |
| Bose und Takhur [65] | Webanwendung | N/A | | | | | | | X | X | X | X (S) | X | | | | | | |
| Smojver et al. [64] | N/A | Eingebettete Entwicklung | X | | | WIF | | X | X | X | X (U) | X (A) | | | | | | | |
| Rong et al. [75] | Webanwendung | N/A | X | | | WIF | | | X | X | X (U) | N/A | N/A | | | | | | |
| Heeager und Nielsen [66] | Medizin | N/A | X | | | WIF | | | X | | X (U) | X (I) | N/A | | | | | | |
| Tanveer [120] | N/A | Large-scale Entwicklung | X | X | | WIF | | | X | | X | X | X | | | | | | |
| Sharma and Wadhwa [72] | N/A | N/A | X | X | | WIF | X (B) | X | X | X | X (U | X (I) | | | | | | | |
| Sultana et al. [62] | N/A | N/A | X | X | X | WIF | X (S, B) | X | X | X | X | X (U,I) | | | | | | | |
| Nisa und Quershi [73] | Informationssysteme | N/A | X | | | WIF | | X | X | | X | X | | | | | | | |
| Taxen and Petterson [40] | N/A | Large-scale Entwicklung | | | | | | | | | | | | X | X | X (U) | X(S) | X | |
| Read und Briggs [68] | IDE Entwicklung | Verteilte Entwicklung | | | | | | | | | | | | X | X | X (A) | N/A | N/A | |
| Paige et al. [67] | Luftfahrt | Eingebettete Entwicklung | X | X | | PF | | | | | | | | X | | X | X | N/A | |
| Heikkilä et al. [41,69] | Telekommunikation | Large-scale Entwicklung | | | | | | | | | | | | X | X | X | N/A | X | |

**Tabelle A.1:** In dieser Übersicht ist dargestellt, welche Frameworks in den jeweiligen Fällen genannt werden und welche Aktivitäten der Frameworks genutzt werden. Die Nutzung einer Aktivität aus einem der Frameworks ist jeweils mit einem X gekennzeichnet. Bei den Fällen, bei denen das Wasserfall-Iteration Framework (WIF) oder Pipeline Framework (PF) mit dem Wasserfall-Agil Framework (WAF) während der Entwicklung kombiniert werden, ist dies in der Entwicklungsaktivität des Wasserfall-Agil Frameworks festgehalten. Die eingeklammerten Buchstaben geben an, welche Tests in dieser Aktivität durchgeführt werden (U: Unit Tests, I: Integrationstests, S: Systemtests, A: Aktzeptanztests, B: Betatests).

**Tabelle A.2:** Liste der identifizierten Tätigkeiten in in der initialen Anforderungsanalyse

| Aktivität | # Nennungen | Publikationen |
|---|---|---|
| Sammeln der Anforderungen | 7 | [60, 63, 64, 70–72, 87] |
| Definition des Umfanges | 7 | [59, 63, 64, 71, 72, 87, 120] |
| Festlegung des Budgets | 5 | [62, 64, 72, 74, 87] |
| Zeit- und Meilensteinplan | 5 | [62–64, 119, 120] |
| Untersuchung des Projektgrundes | 5 | [59, 62, 67, 72, 120] |
| Erstellung einer Vision | 4 | [59, 72, 74, 119] |
| Festlegung des zeitlichen Umfangs | 4 | [62, 64, 70, 72] |
| Zusammenstellung des Projektteams | 4 | [63, 64, 74, 119] |
| Analyse der Stakeholder | 1 | [60] |
| Festlegung der benötigten Ressourcen | 1 | [64] |
| Analyse von Restriktionen | 1 | [59] |

**Tabelle A.3:** Liste der identifizierten Tätigkeiten während des initialen Architekturdesigns

| Aktivitäten | # Nennungen | Publikationen |
|---|---|---|
| Analyse und Definition von Schnittstellen | 3 | [60, 72, 119] |
| Analyse der verwendeten Technologie und Tools | 3 | [59, 60, 87] |
| Analyse der benötigten Standards | 1 | [59] |
| Festlegung der Komponenten | 1 | [59] |

**Tabelle A.4:** Plan-basierte Ziele und Maßnahmen in APH-Ansätzen

| **Ziele / Gründe** | **Publikationen** | **Maßnahmen** | **Publikationen** |
|---|---|---|---|
| Umgang mit Sicherheits-, Security- und Regulationsanforderungen (nicht-funktionalen Anforderungen) | [43, 78–80, 83] [34, 67, 81, 90, 121] | Up-front Anforderungsanalyse | [34, 78, 80, 81, 90] |
| | | Up-front Architekturdesign | [79, 83] |
| | | Wasserfall-Iteration Framework | [43, 80] |
| | | Testphase am Ende des Projektes | [43] |
| | | Up-front Gefahrenanalyse | [34] |
| | | V-Modell | [121] |
| | | Pipeline Framework | [67] |
| Koordination von großen Projektteams | [77, 82, 84, 85, 91] [**?**, 92, 96, 97] | Zentrales Team für Koordination und Planung | [82, 92, 97] |
| | | Up-front Architekturdesign (Hohe Abstraktion) | [84, 91] |
| | | Langzeit Planung | [82, 85] |
| | | Pipeline Framework | [69, 96] |
| | | Up-front Anforderungsanalyse | [85] |
| | | Verträge | [85] |
| | | Up-front Definition des Umfanges | [85] |
| | | Risikoanalyse | [85] |
| | | Formale Kommunikation und Kontrolle | [85] |
| Bedürfnis nach Planungssicherheit und fixierten Ressourcen | [32, 43, 69, 77, 93] [90] | Fokus auf Planung und Kontrolle | [32] |
| | | Up-front Definition von Rahmenbedingungen | [77] |
| | | Up-front Anforderungsanalyse | [90] |
| Verwaltung des Überblicks und Abhängigkeiten | [70, 83, 90, 94, 95] | Up-front Architekturdesign | [83, 94, 95] |
| | | Wasserfall-Iteration Framework | [70] |
| | | Up-front Anforderungsanalyse | [90] |
| Koordination von verteilter Entwicklung | [70, 89] | Up-front Anforderungsanalyse (Sichtbarkeit des Projektes) | [70, 89] |
| | | Up-front Architekturdesign | [89] |
| | | Formale Kommunikationspraktiken (Koordinationsrolle) | [89] |
| Klärung des Projektziels und Leistungen | [79, 82] | Up-front Anforderungsanalyse | [79, 82] |
| | | Up-front Planung | [79, 82] |
| Kontrolle von Änderungen | [80, 85] | Formale Änderungskontrolle | [80, 85] |
| Senkung von Kosten | [39, 90] | Up-front Architekturdesign | [39, 90] |
| Umgang mit komplexer Entwicklung | [32, 39, 77] | Pipeline Framework | [32] |
| Traceability | [43] | N/A | N/A |
| Verbesserung von Risikomanagement | [70] | Wasserfall-Iteration Framework | [70] |
| Testen von embedded Software | [67] | Testphase am Ende des Projektes | [67] |

**Tabelle A.5:** Agile Ziele und Maßnahmen in APH-Ansätzen

| **Ziele / Gründe** | **Publikationen** | **Maßnahmen** | **Publikationen** |
|---|---|---|---|
| Flexibilität bei Änderungen (Anforderungen, Technologie, etc.) | [33, 77–81] [34, 69, 70, 76, 85, 96, 121] [67, 68, 89, 90, 92, 95] | Iterative Entwicklung (Scrum Framework) | [34, 79, 90] |
| | | Ganzheitliche und co-located Teams | [89] |
| | | Dezentrale Entscheidungen | [92] |
| | | Agile Architektur | [76] |
| Umgang mit unsicheren Anforderungen | [34, 76, 81–86] | Agile Architektur | [76] |
| Schnelle Entwicklung und kurze Time-to-market | [43, 77, 81, 87–90] | Iterative Entwicklung (Scrum Framework oder XP) | [81, 87, 90] |
| | | Autonome Teams | [88] |
| Verbesserung der Kommunikation | [33, 90, 96] | Scrum Framework | [90] |
| Verbesserung der Transparenz des Projektstatus | [70, 96] | Scrum Framework | [70] |
| Erhöhung der Produktivität | [70, 83] | Scrum Framework | [70] |
| Verringerung der Kosten | [43, 81] | N/A | N/A |
| Verbesserung der Businessausrichtung und Kundenzufriedenheit | [77, 78] | Zusammenarbeit mit dem Kunden | [78] |
| Verbesserung der Motivation der Mitarbeiter | [69, 90] | Scrum Framework | [90] |
| Adaptivität von Plänen | [82] | Schneller Entscheidungsprozess | [82] |
| Verbesserung der Planbarkeit von Ressourcen | [96] | Kontinuierliche Planung | [96] |
| Frühe Schaffung von Wert | [86] | N/A | N/A |
| Frühe Entdeckung von Problemen | [90] | Scrum Framework | [90] |

# B

# Daten der Interviewstudie

## B.1 Interviewfragen

1. **Hintergrund**
   - Was ist Ihre Rolle, Hintergrund und Erfahrung in der Firma?
   - Was ist der Hauptgeschäftsbereich Ihres Unternehmens?
   - Welches Produkt entwickeln Sie in Ihrem Projekt?
   - Wie viele Personen sind an Ihrem Projekt beteiligt?
2. **Organisation des Entwicklungsansatzes**
   - Wie verläuft der Start eines neuen Projektes?
   - Werden Phasen für einzelne Aktivitäten (RE, Architektur, Planung) genutzt?
   - Welches Vorgehen wird während der Entwicklung genutzt?
   - Wie werden die Anforderungen während der Entwicklung gesammelt und verwaltet?
   - Welche Langzeitplanung wird im Projekt durchgeführt?
   - Wie wird die Architektur zu Beginn und während des Projektes entwickelt?
   - Welche Struktur wird genutzt, um die Teams zu koordinieren?

- Welcher Managementstil wird für die Teams genutzt?
- Gibt es eine separate Test- und Operationsphase?

3. **Vorteile und Nachteile von agilen und plan-basierten Praktiken**
   - Welche Vorteile und Nachteile bringen die agilen Praktiken?
   - Welche Vorteile und Nachteile bringen die plan-basierten Praktiken?
4. **Schwierigkeiten bei der Nutzung von agilen und plan-basierten Praktiken**
   - Welche Konflikte entstehen durch die Nutzung von agilen und plan-basierten Praktiken?
   - Welche Aktivitäten sind davon betroffen?
5. **Kombination von agilen und plan-basierten Praktiken**
   - Wie wird ein Kompromiss zwischen den agilen und plan-basierten Praktiken gefunden?
   - Was muss mindestens up-front in einem Projekt durchgeführt werden?
   - Wann werden up-front Tätigkeiten schädlich in einem Projekt?
   - Wie erkennt ein Projektteam wann es die up-front Aktivitäten abbrechen sollte?
   - Welche zusätzlichen Praktiken werden genutzt, um die agilen und plan-basierten Praktiken zu verbinden?
6. **Herausforderung bei der Kombination**
   - Was erschwert das Finden eines Kompromisses?
   - Was fehlt für das erfolgreiche Verbinden von agilen und plan-basierten Praktiken?

## B.2 Ziele und Maßnahmen in APH-Ansätzen

**Tabelle B.1:** Identifizierte agile Ziele und Maßnahmen in APH-Ansätzen in der Interviewstudie

| **Agile Ziele** | **Anzahl an Nennungen** | **Maßnahmen** | **Anzahl an Nennungen** |
|---|---|---|---|
| Flexibilität | 7 | Variabler Umfang | 2 |
| | | Umpriorisierung von Meilensteinen | 2 |
| | | Kontinuierliche Reviews / Feedbackzyklen | 2 |
| | | Stetiger Austausch | 1 |
| | | Anforderungen offen lassen | 1 |
| | | Schnelle Analyse und Schätzung von neuen Anforderungen | 1 |
| Umgang mit unbekannten Anforderungen / Technologien | 7 | Kontinuierliche Reviews / Feedbackzyklen | 3 |
| | | Lernzyklen | 1 |
| | | Umpriorisierung von Meilensteinen | 1 |
| | | Iteratives Vorgehen | 1 |
| | | Variabler Umfang | 1 |
| Steigerung von Kundenzufriedenheit / Einbindung des Kunden | 3 | Kontinuierliche Reviews / Feedbackzyklen | 2 |
| Steuerung des Projektes / Überprüfung des Projektstatus | 3 | Nutzung der Struktur von Sprints (Dailies, Planning Meetings etc.) | 3 |
| Schnelle Entwicklung / Schaffung von frühem Wert | 2 | Up-front Aufwand reduzieren | 1 |
| | | Tägliche Auslieferung | 1 |
| | | Test Automatisierung | 1 |
| Verbesserung der Kommunikation | 1 | Selbstorganisation | 1 |
| Schnelle Fehlerbehebung | 1 | Automatisiertes Testen | 1 |
| | | Test Driven Development | 1 |
| | | DevOps Verantwortung | 1 |

**Tabelle B.2:** Identifizierte plan-basierte Ziele und Maßnahmen in APH-Ansätzen in der Interviewstudie

| **Plan-basierte Ziele** | **Anzahl an Nennungen** | **Maßnahmen** | **Anzahl an Nennungen** |
|---|---|---|---|
| Vision und gemeinsames Verständnis für das Projekt schaffen | 6 | Up-front Anforderungsanalyse | 5 |
| | | Nutzen einer Roadmap | 1 |
| Schaffung einer Struktur für die Architektur | 5 | Up-front Architekturphase | 4 |
| | | Aufsicht über Architektur während der Entwicklung | 1 |
| Grobe Schätzung des Projektes | 4 | Up-front Anforderungsanalyse | 3 |
| Umgang mit konkreten Preisen zu konkreten Features | 4 | Festlegung des Umfanges für eine Iteration | 4 |
| | | Genaue Planung einer Iteration | 4 |
| | | Grobkonzept für Features | 1 |
| | | Nutzung von Change Requests | 1 |
| | | Nutzung einer Roadmap | 1 |
| Klare Testverantwortung / Strukturierter Testprozess | 4 | Extra Testteam für Integrationstests | 4 |
| Umgang mit kritischen Systemen | 4 | Up-front Anforderungsphase | 3 |
| | | Auslieferung der Software am Ende des Projektes | 1 |
| | | Extra Testteam | 1 |
| Planungsstruktur für das Projekt | 3 | Teamleitung mit Roadmap | 2 |
| | | Up-front Definition des Umfanges | 1 |
| Abstimmung zwischen Projekten | 2 | Umfang für einen begrenzten Zeitraum festlegen | 2 |
| | | Umfang für einen begrenzten Zeitraum festlegen | 1 |
| Koordination bei der Integration | 1 | Integrationsteam | 1 |

# C

# Interviewaussagen und Literaturzitate für die Ableitung von Risiken

## C.1 Requirements Engineering

### C.1.1 Das Risiko von zu viel Anforderungsanalyse

#### C.1.1.1 R1.1: Unsicherheit über die Projektrichtung und Abhängigkeiten

**Interviewer:** ”Wofür benötigt ihr eine Anforderungsanalyse zu Beginn des Projektes?”
**Interviewteilnehmer 04:** ”Um den Überblick zu behalten und zu schauen, in welche Richtung die Reise geht. Wenn du von außen erstmal gucken kannst, ok, grob gesehen brauche ich das und das und das, dann kann ich das leichter aufdröseln und vielleicht die Aufgaben leichter verteilen.”

**Hayata et al. [87, S.4]:** ”First, the project team and the customer can apply Waterfall-UpFront to specify requirements, formulate documents, and bind them together as a contract. This will reduce the risk of upfront ambiguities in terms of the project goals and deliverables.”

**Interviewteilnehmer 01:** ”Die Anforderungen werden [...] in Form von Epics, beschrieben. Sie müssen erstmal so die groben Pfeiler einrammen und sagen in diese Richtung soll es gehen. Ansonsten bewegen Sie sich ja im chaotischen Bereich und da wollen wir ja nicht hin.”

**Interviewer:** ”Wofür habt ihr zu Beginn eine Anforderungsanalysephase durchgeführt?”
**Interviewteilnehmer 08:** Sonst hätten wir wahrscheinlich irgendetwas gebastelt oder gebaut, was dann niemandem etwas gebracht hätte. Das war auf jeden Fall sehr wichtig, dass wir uns darüber informiert haben, was wir eigentlich machen sollen.

#### C.1.1.2 Kontextfaktoren für R1.1

**Anzahl und Verfügbarkeit der Stakeholder**

**Interviewer:** ”Warum habt ihr bereits so viele Anforderungen zu Beginn erhoben?”
**Interviewteilnehmer 04:** ”Es würde die [Stakeholder] tatsächlich auch ein bisschen überfordern, wenn man das [Anforderungen] jetzt regelmäßig mit denen abstimmen würde. Das wäre glaube ich ein bisschen schwierig. Wenn man einmal die Woche oder alle zwei Wochen sich mit denen hinsetzt und das nochmal komplett neu aufdröseln würde, wäre glaube ich ein bisschen tricky.”
**Interviewer:** ”Würdet ihr denn überhaupt die Stakeholder wieder zusammenbekommen?”
**Interviewteilnehmer 04:** ”Ne”

**Größe und Verteilung des Projektteams**

**Qureshi [85, S.3]:** ”Stable requirements facilitate a development team [mittleres bis großes Entwicklungsteam] to achieve strong architectural design [...].”

**Art der Abhängigkeiten**

**Interviewer:** ”Warum ist hier mehr Anforderungsanalyse nötig?”
**Interviewteilnehmer 12:** ”Da sind ganz viele Abteilungen dran, die sich daraus bedienen. Da ist es wichtig, die Anforderungen erstmal sauber aufsammeln zu können. [...] Da hängt natürlich ein ganzer Rattenschwanz dran. Ich kann nicht einfach meine API ändern und erwarten, dass der andere auch direkt mitzieht. Sondern da ist halt mehr Abstimmung nötig.”

#### C.1.1.3 R1.2: Projektteam gibt falsche Schätzung ab

**Interviewteilnehmer 13:** ”Wir haben den Anspruch die [Anforderungen] zu verstehen und zwar so tief, dass das Team am Ende sagen kann, ok, wir haben jetzt eine Vorstellung, was da gebraucht wird. Und wir sind auch in der Lage, so einen Daumenwert abzugeben, wie viel Entwicklungsaufwand darin steckt.”

**Interviewteilnehmer 01:** ”So einen gewissen Detailierungsgrad benötigen Sie [bei der Anforderungsanalyse] und der muss so gut sein, dass Sie in der Lage sind, auf der Basis eine initiale Aufwandsschätzung abzugeben, um dann quasi das Projekt zu beauftragen und anzugehen. [...] Mein Ziel ist es am Anfang eines Projektes, die Anforderungen zumindest auf einem Detaillevel zu verstehen, sodass ich sie A verstanden habe und B, dass ich in der Lage bin, zumindest mal eine grobe Indikation zu geben, wie aufwändig die Umsetzung wäre. Das heißt, ich habe mal so eine grobe Idee, was das denn IT technisch bedeutet, was ich machen muss. Dann kann ich das auch entsprechend abschätzen.”

#### C.1.1.4 Kontextfaktoren für R1.2

**Bedürfnis nach fixierten Ressourcen / Existenz von Deadlines**

**Interviewteilnehmer 01:** ”Man bindet ja interne Ressourcen und gegebenenfalls auch noch externe, die dann richtig Cash-out bedeuten. Das muss man sich dann genehmigen lassen und dann geht es los und dann wird das Grobkonzept weiter detailliert. [...] Das heißt, Sie müssen schon gucken, bin ich personell richtig besetzt, welche Leute brauche ich. Häufig haben Sie in Unternehmen ja auch so ein Ressourcenanforderungsprozess, den man dann auch im Sinne der Projektintialisierung starten muss, damit Sie überhaupt erstmal ein Team haben, was zusammenkommt. Das heißt, Sie brauchen die Ressourcen. Sie brauchen irgendwie ein Budget, dafür brauchen Sie eine Aufwandschätzung.”

**Interviewteilnehmer 03:** ”Wenn ich am Ende dieser Punkte oder nach dieser Zeit sage, ok, ich habe diese Punkte nicht geschafft, wir schieben das ins nächste Release, dann widerspricht das dem klassischen Einkaufsverhalten [unserer Kunden], weil dann eine Leistung nicht erfolgt ist, die festgeschrieben ist.”

#### C.1.1.5 R1.3: Übersehen von kritischen nicht-funktionalen Anforderungen

**Interviewteilnehmer 01:** ”Wenn ich solch eine Anforderung [nicht-funktionale Anforderung] erst sehr spät im Projekt bekomme, dann laufe ich Gefahr, dass ich das, was ich entwickelt habe, einmal kräftig in die Tonne treten kann. Und muss es komplett neu entwickeln.”

**Batra et al. [82, S.4]:** ”Some of the user requirements were not defined clearly with the needed specificity in the beginning. [. . . ] They had also underestimated the security and privacy needs of the system, such as protecting customer credit-card information.”

**Interviewteilnehmer 11:** ”Die Systeme werden ja heute, wenn du sie kaufst, konfiguriert. Da kann ich nicht sagen, ich fange mal an und setze was auf und dann mache ich nochmal eine Schleife und nochmal eine Schleife. Und dann komme ich zu einem Punkt, wo ich feststelle, verdammt, vier Schleifen vorher falsch konfiguriert. Erstmal alles auf null und alles von vorne.”

#### C.1.1.6 Kontextfaktoren für R1.3

**Art der nicht-funktionalen Anforderungen**

**Interviewer:** ”Wann ist mehr up-front Anforderungsanalyse nötig?”
**Interviewteilnehmer 12:** ”Ja, wenn man viel mit Compliance zu tun hat, man also viele Gesetzesauflagen erfüllen muss. Viele Rahmenbedingen, die man quasi nicht umgehen kann. Man kann keine Anwendung entwickeln und die dann auf den Markt bringen und die erfüllt nicht mal die minimalen Complianceanforderungen.”

### C.1.2 Risiko von zu viel Anforderungsanalyse

#### C.1.2.1 R1.4: Aufnahme von falschen Anforderungen

**Interviewer:** ”Was ist der Vorteil von agiler Entwicklung für euch?”
**Interviewteilnehmer 02:** ”Erlebt der Kunde am Ende dann das Ergebnis und hat dann entweder ein Wohlfühlerlebnis oder einen Schrecken ohne Ende. Und das ist halt bei dem Agilen anders.”

**Heeagar and Nielsen [34, S.14]:** ”The software team especially struggled with slow changeability of the software requirements specification. This problem was exacerbated by the fact that several documents had been written in the early stages of the

project, both due to requirements by the FDA standards that certain documents were to be produced before the product was developed (e.g. a safety hazard analysis) but also due to the overall project model which focused on milestones and written evidence for progress."

**Interviewer:** "Warum verwendet ihr ein agiles Vorgehen?"
**Interviewteilnehmer 09:** "Weil wir festgestellt haben, dass es keinen Sinn macht, sich zwei Jahre lang einzusperren und mit dem Kunden über Anforderungen zu sprechen. Dann sich zwei Jahre lang hinzusetzen und das Ganze zu entwickeln, dem Kunden zu zeigen und er hinterher sagt, das wollte ich gar nicht so haben."

**Interviewteilnehmer 14:** "Der Nachteil [von up-front Anforderungsanalyse] ist ganz einfach. Das ist eine komplexe Materie. Je weiter ich spezifiziere in die Zukunft, desto mehr Fehler mache ich."

#### C.1.2.2 Kontextfaktoren für R1.4

**Bekanntheit der Anforderungen**

**Interviewteilnehmer 13:** "Tatsächlich stellen wir aber in der Entwicklungsphase fest, dass sich 60 – 70 Prozent der Anforderungen nochmal ändern. Ein Learning ist, dass du in der Softwareentwicklung nicht in der Lage bist, die Anforderungen in dieser frühen Phase zu machen und zu sagen, das setzte ich jetzt um."

**Interviewteilnehmer 01:** "Je unklarer das [Bekanntheit der Anforderungen] ist, umso eher ist die Wahrscheinlichkeit, dass ich ins Agile reingehe."

## C.2 Architektur

### C.2.1 Das Risiko von zu wenig Architekturdesign

#### C.2.1.1 R2.1: Architektur bildet sich zufällig

**Waterman et al. [39, S.1]:** "Too little planning however may lead to an accidental architecture that has not been carefully thought through, and may lead to the team spending a lot of time fixing architecture problems and not enough time delivering functionality (value). An accidental architecture can potentially lead to gradual failure of the project."

**Cao et al. [33, S.3]:** ”It is found that without an overall design, it is difficult to maintain a big picture of the project as the systems grow.”

**Interviewteilnehmer 09:** ”Wenn die Leute auf Tools, Entwicklungsumgebung oder Architekturen festgepinnt sind, [..]. Aber auf der anderen Seite erhöht es die Wartbarkeit des Systems ungemein, weil die Architektur, die wir vorgeben, dafür sorgt, dass das Produkt langlebiger ist.”

#### C.2.1.2 Kontextfaktoren für R2.1

**Größe und Verteilung des Projektteams**

**Interviewteilnehmer 09:** ”Die Entwickler werden eingeschränkt durch die Vorgaben, die global getroffen werden. Das halte ich durchaus für eine Problem. Auf der einen Seite erzeugt es eine gleichmäßige Architektur über die gesamte Firma. [. . . ]. Je größer ich bin, desto mehr Vorgaben benötige ich auch, damit nicht jeder sein eigenes Süppchen kocht.”

**Nutzung von Anwendungsframeworks**

**Waterman et al. [39, S.8]:** ”Frameworks – particularly those that follow the ‘convention over configuration’ paradigm – also greatly reduce the complexity of the architecture because many of the architectural decisions are embedded in the framework, and hence architectural changes can be made with a lot less effort.”

#### C.2.1.3 R2.2: Auftreten von Koordinationsproblemen

**Islam et al. [90, S.11]:** ”Large systems engineering projects often depend on significant upfront design as a means of coordinating effort between different subteams working on software, firmware and hardware element.”

#### C.2.1.4 Kontextfaktoren für R2.2

**Größe und Verteilung der Entwicklung**

**Interviewteilnehmer 09:** ”Die Entwickler werden eingeschränkt durch die Vorgaben, die global getroffen werden. Das halte ich durchaus für ein Problem. Auf der einen Seite erzeugt es eine gleichmäßige Architektur über die gesamte Firma.[. . . ]

Je größer ich bin, desto mehr Vorgaben benötige ich auch, damit nicht jeder sein eigenes Süppchen kocht."

**Islam et al. [90, S.11]:** "Large systems engineering projects often depend on significant upfront design as a means of coordinating effort between different subteams working on software, firmware and hardware elements."

#### Art der Abhängigkeiten

**Waterman et al. [39, S.10]:** "Integration with other systems require data and communications to be mapped between the systems, which adds to the up-front effort to ensure integration is possible with the technologies being used."

#### C.2.1.5 R2.3: Nichterfüllung kritischer nicht-funktionaler Anforderungen

**Interviewteilnehmer 01:** "Wie ich schon sagte, so Kapazität, Verfügbarkeit, Performance ist etwas, das muss ich sehr früh im Design berücksichtigen, sonst laufe ich da gegen eine Wand. Wenn ich auch sehr strenge Sicherheitsanforderungen habe, dann muss ich die auch schon von Anfang an mit ins Design einbauen."

**Interviewteilnehmer 06:** "Das bezieht sich bei uns halt so auf Sensorsignale. Wie müssen die aufbereitet werden? Welche Abtastzeit braucht unsere Komponente überhaupt? Wie schnell müssen wir da reagieren? In welcher Tasklaufzeit muss sie laufen? Das muss natürlich alles vorher stehen. Genau diese Architekturentscheidungen, die müssen vorher gefallen sein."

### C.2.2 Das Risiko von zu viel Architekturdefinition

#### C.2.2.1 R2.4: Überarbeitung der Architektur

**Interviewteilnehmer 08:** "Hätte man gewusst, dass es Entscheidungen auf bestimmten Ebenen geben wird, die manche Architektursachen verhindern, dann hätte man das gar nicht so feingranular aufgebaut. Das hat dann natürlich ein bisschen mehr Arbeit gebracht, die vernichtet werden musste und nicht mehr relevant war im Nachhinein."

**Interviewteilnehmer 12:** "Je feiner man die Architektur macht, desto häufiger muss man sie danach auch revidieren, weil dann die Erfahrung oder die Wahrheit mit der Entwicklungszeit kommt und man sieht, welche Sachen gut oder schlecht laufen."

### C.2.3 Kontextfaktoren für R2.4

#### Bekanntheit der Anforderungen

**Waterman et al. [39, S.7]:** ”Delaying decision-making means not making architecture decisions too early; waiting until sufficient information on the requirements is known so that there is less likelihood of the decisions needing to be changed.”

## C.3 Planung

### C.3.1 Das Risiko von zu wenig Langzeitplanung

### C.3.2 R3.1: Auftreten von Koordinationsproblemen

**Interviewteilnehmer 11:** ”Und dann ich würde mal sagen, so eine medium size Roadmap erstellen, damit man so eine gewisse grobe Vorplanung hat, wann was passiert. Die wird sich natürlich auch verändern, das ist auch klar. Da gibt es auch Verzögerungen. Da reicht es eigentlich, wenn man so grob weiß, das sind die Blöcke. Das muss da fertig sein. So, damit kann man eigentlich arbeiten. So bekommt man dann die ganzen Gewerke und Personen kanalisiert. Damit die auch zielgerichtet auf die Vision hinarbeiten können.”

**Interviewteilnehmer 12:** ”Man muss diese Langzeitplanung auf jeden Fall haben, damit man nicht nur von Tag zu Tag entwickelt, sondern auf Sichtweite. Du musst ja auch ungefähr wissen, wo es hingeht, damit du bestimmte Architekturentscheidungen jetzt schon abschätzen kannst.”

**Interviewteilnehmer 11:** ”Besonders, wenn du Dinge hast aus Projektmanagementsicht, die ineinandergreifen, gibt es Abhängigkeiten von User Stories von Team 1 und User Stories von Team 2. [. . . ]. Da den Überblick besonders in großen Projekten zu behalten, sind diese Roadmaps super hilfreich.”

#### Kontextfaktoren für R3.1

#### Größe und Verteilung des Projektteams

**Batra et al. [82, S.11]:** ”In addition, the distributed nature of the project required the discipline and control provided by the structured approach that was necessary

for planning and coordination of the various aspects of the project across different locations."

**Art der Abhängigkeiten**

**Interviewteilnehmer 01:** "Wenn ich das jetzt projekthaft nur mache, dann ist meine Roadmap, dass ich sage, welche Abhängigkeiten habe ich zu Backend Systemen, die eben nicht agil arbeiten und wo sind meine Synchronisationspunkte. Das ergibt dann meine Roadmap."

#### C.3.2.1 R3.2: Nichteinhaltung von Projektparametern

**Pechai [93, S.4]:** "Project management depends on long-term planing to predict the time and cost need to execute a project."

**Waterman et al. [39, S.6]:** "Many non-agile customers prefer the accountability of a fixed price contract with a fixed scope and fixed delivery dates – perhaps with penalties if milestones are not met.[...] For a team, these contracts mean investing significant amounts of time up-front to map out in detail how much work is required to deliver the customer's list of requirements."

**Interviewteilnehmer 06**: "Der Zeitraum kam tatsächlich vom Kunden. Sie wollen bis dann und dann ein Ergebnis haben, wo es funktioniert. [...] Und dann kann man halt zurückrechnen, ok, ein Jahr vorher sollen die Feldtests starten und dann weiß man, bis dahin muss das auf jeden Fall fertig sein."

#### C.3.2.2 Kontextfaktoren für R3.2

**Bedürfnis nach fixierten Ressourcen und die Existenz von Deadlines**

**Interviewteilnehmer 01:** "Also ich habe bestimmte Verpflichtungen, weil die meisten Banken immer noch feste Releasetermine haben."

**Interviewteilnehmer 03:** "Wenn ich am Ende oder nach dieser Zeit sage, ok, ich habe diese Punkte nicht geschafft, wir schieben das ins nächste Release, dann widerspricht das dem klassischen Einkaufsverhalten, weil dann eine Leistung nicht erfolgt ist, die festgeschrieben ist."

### C.3.3 Das Risiko von zu viel Langzeitplanung

### C.3.4 R3.3: Verminderung der Flexibilität

**Interviewteilnehmer 12:** ”Je enger man das Raster für die Planung legt, desto eher wird man auch daneben liegen und sie häufig revidieren müssen.”

**Interviewer:** ”Welchen Vorteil hat agiles Arbeiten für euch?”
**Interviewteilnehmer 08:** ”Der Vorteil ist ganz klar, dass wir schnell auf irgendwelche Priorisierungsänderungen reagieren können. Wir müssen nicht irgendwelche Dokumente umschreiben, sondern machen in Jira bei den entsprechenden Karten die Priorisierung anders.”

**Interviewteilnehmer 11:** ”Je feiner du [die Planung] ausgliederst, desto mehr totdokumentierst du dich, weil es ist ja ein lebendes Dokument. Das muss ja kontinuierlich gepflegt und gewartet werden. Dann erzeugst du einfach einen Rattenschwanz an Veränderungsarbeit.”

#### C.3.4.1 Kontextfakoren für R3.3

**Bekanntheit der Anforderungen und Technologien**

**Interviewer:** ”Wonach entscheiden Sie, ob sie agil oder traditionell entwickeln?”
**Interviewteilnehmer 01:** ”Klarheit der Anforderungen, Bekanntheit der Technologie. Da gibt es so eine klassische Matrix, wo Sie sagen können, bin ich in einem einfachen, in einem komplexen Umfeld oder in einem komplizierten Umfeld.”

**Interviewteilnehmer 02:** ”Ich kann diesen Fachkonzeptansatz gar nicht fahren, sondern da ist dann diese agile Vorgehensweise auf jeden Fall deutlich besser, weil man dann bestimmte Erkenntnisse da hat [...].”

**Verfügbarkeit des Projektteams**

**Interviewteilnehmer 06:** ”Wir werden halt immer wieder abgezogen, wenn halt irgendwo was ist. Dann kommt wieder ein Kundenworkshop, dann ist irgendwo wieder ein Claim und dann hadert das. Dann ist irgendwas wieder höher priorisiert. Der Kunde ist wichtiger als der andere Kunde.”

## C.4 Testen

### C.4.1 Das Risiko von zu wenig Trennung beim Testen

### C.4.2 R4.1: Qualitätsverlust bei der Integration

**Interviewer:** ”Was ist der Vorteil eines extra Testteams?”
**Interviewteilnehmer 12:** ”Ein Vorteil ist, du weißt ja nicht, wenn du es ausrollst, ob es noch mit den anderen funktioniert. Wenn du da nicht ordentlich getestet hast oder mit den anderen kommunizierst, dann ist natürlich so ein Testteam Gold wert, das das Gesamte so ein bisschen kanalisiert.”

### C.4.3 Kontextfaktoren für R4.1

#### Größe und Verteilung des Projektteams

**Interviewteilnehmer 01:** ”Stellen Sie sich mal vor, Sie haben einen N-to-N Prozess der 7 oder 8 Softwareprodukte beinhaltet. Wem wollen Sie da die Verantwortung geben, können Sie gar nicht.”

#### Existenz von Qualitätsstandards

**Interviewteilnehmer 03:** ”Wir haben eine Testdoku, die wir dann auch wasserfallmäßig den Kunden nachweisen müssen. Die verlangen zum Teil, dass sie Testdoku sehen wollen, die aus dem Systemtest gekommen ist. Das machen wir dann am Ende.”

#### Kritikalität der Integration

**Interviewteilnehmer 11:** ”Es kommt auch auf die Resilienz deines Systems an. Bei Applikationen wo es egal ist, ob ein Service mal funktioniert oder nicht, weil das Gesamtsystem so modular ist, dass es nicht in die Knie geht [wird kein extra Testteam benötigt]. Bei Applikationen, wo eine Sache nicht mehr funktioniert und dadurch das gesamte System nicht mehr funktioniert, [wird ein extra Testteam benötigt]. Da kommt es auf die Projektart drauf an. ”

**Aufwand bei der Integration**

**Interviewteilnehmer 11:** ”Integrationstests sind immer schwieriger.[...] Je mehr Teilsysteme du hast, je komplexer dein Umfeld ist, desto sinnvoller ist es mit Staging Ansätzen zu arbeiten und mit losgelösten Teams das Testing durchzuführen.”

**Eingebettete Software**

**Interviewteilnehmer 06:** ”Wir testen dann immer zu den jeweiligen Quartalsreleases und auch am Ende nochmal sehr groß. Dann fahren wir damit auch mal auf der Straße. Das ist dann nicht mehr spezifikationsbasiert, sondern wirklich Systemtests.[...] Vor der Serie, wird das dann nochmal alles intensivst durchgetestet.”

**Auswirkungen von Fehlern**

**Interviewteilnehmer 06:** ”Bevor der Kunde auf der eigenen Testbahn fahren darf, haben wir bestimmte Mindestanforderungen und die müssen dann getestet werden.”

**Interviewteilnehmer 11:** ”Es kommt auch auf die Resilienz deines Systems an. Bei Applikationen wo es egal ist, ob ein Service mal funktioniert oder nicht, weil das Gesamtsystem so modular ist, dass es nicht in die Knie geht [wird kein extra Testteam benötigt]. Bei Applikationen wo eine Sache nicht mehr funktioniert und dadurch das gesamte System nicht mehr funktioniert [wird ein extra Testteam benötigt]. Da kommt es auf die Projektart drauf an.”

### C.4.4 Das Risiko von zu viel Trennung beim Testen

#### C.4.4.1 R4.2: Verzögerungen bei der Entwicklung

**Interviewteilnehmer 13:** ”Tests werden automatisch durchgeführt. Wenn du Code in drei Monaten depolyst und ein Fehler auftritt, dann weißt du nicht mehr, was du da gemacht hast.”

**Rottier et al. [100, S.4]**: ”Initially we tried to circumvent this issue by allowing the testing of a User Story to happen during the subsequent Sprint while the developers worked on something else. As most of those experienced in Agile will know, this proved to be a bad idea. Stories that were supposed to be done kept creeping back

by way of bugs into the backlog and this quickly started to seriously degrade our velocity."

Heeagar and Nielson [66, S.7]: "The fact that the system test is not done by the software team, but by the system engineering group, limits the flexibility of handling the errors found at this test."

## C.5 Koordination

### C.5.1 Das Risiko von zu wenig zentraler Koordination

#### R5.1: Eintreten von Kommunikationsfehlern

**Ramesh et al. [89, S.10]:** "There were several incidents of miscommunication among the members of the distributed teams in Manco. These were attributed to "overreliance on informal communication"".

**Xu [92, S.6]:** "Second, individuals on each team may not have complete information regarding the project because of its size and complexity. In this case, mainly relying on horizontal communication at the individual level between two teams can result in incomplete, ambiguous, and even inconsistent information and knowledge, creating coordination obstacles for the project."

#### C.5.1.1 Kontextfaktoren für R5.1

#### Größe und Verteilung des Projektteams

Xu [92, S.4]: "When the team size increases beyond a certain point, decentralization can cause confusion among team members because of the larger volume of information and more complex interdependence among team members and tasks. It is reasonable to argue that in large teams, self-organizing alone is not sufficient in decision-making."

#### C.5.1.2 R5.2: Die Architektur bildet sich zufällig

**Interviewteilnehmer 01:** "Da ist es natürlich zum einen wichtig, dass die [Entwicklerteams] sich da über das Integrationsteam sauber abstimmen und sagen, ok, wir haben jetzt vor einen vielleicht gewissen Architekturrahmen zu verlassen. Ist das ok, können wir das überhaupt so machen, weil wir möchten wesentliche Dinge ändern."

**Interviewteilnehmer 02:** ”Das heißt, die [extra Team] haben so den Design und Architektur Gesamtüberblick. Die müssen gucken, dass das, was da vorne hergestellt wird, auch gewissen Bedingungen genügt oder Richtlinien, die bei uns im Unternehmen zwingend eingehalten werden müssen, damit da kein Wildwuchs entsteht.”

**Interviewteilnehmer 09:** ”Wenn die Leute auf Tools, Entwicklungsumgebung oder Architekturen festgepinnt sind, [..]. Aber auf der anderen Seite erhöht es die Wartbarkeit des Systems ungemein, weil die Architektur, die wir vorgeben, dafür sorgt, dass das Produkt langlebiger ist.”

#### C.5.1.3 Kontextfaktoren für R5.2

**Größe und Verteilung der Entwicklung**

**Interviewteilnehmer 01:** ”Ich brauche ein Integrationsteam, was halt dafür sorgt, dass diese beiden Teams nicht aneinander vorbeilaufen. Weil das [mehrere Teams nutzen], kann ich auch nicht beliebig machen. Irgendwann steigt meine Komplexität in der Kommunikation. Das heißt, ich kann maximal drei Teams parallel laufen lassen. Wenn ich ein viertes Team noch parallel habe, dann bekomme ich es irgendwann nicht mehr koordiniert.”

**Nutzung von Anwendungsframeworks**

**Waterman et al. [39, S.9]:** ”S5 (USE FRAMEWORKS AND TEMPLATE ARCHITECTURES) can be used to significantly reduce development and design effort; in particular it significantly reduces risk and up-front effort for standard problems.[...] S5 provides the benefit of standard solutions to standard problems, which means that software engineers do not need to make as many architectural decisions, and can greatly reduce the effort required to design a system and get it up and running.”

### C.5.2 Das Risiko von zu viel zentraler Koordination

#### C.5.2.1 Verzögerung bei der Entwicklung

Xu [92, S.3]: ”The logical reasoning behind this proposition is straightforward. One of the agile principles is to empower individual members to make decisions effectively without going through a hierarchy, thus making it possible to address problems more quickly.”

# D

# Äquivalenzklassen für die Anpassung des APH-Frameworks

## D.1 Requirements Engineering

**Tabelle D.1:** Äquivalenzklassen beim Requirements Engineering

| Risiko von zu wenig Anforderungsanalyse | Risiko von zu viel Anforderungsanalyse | |
|---|---|---|
| | R1.4 ist nicht kritisch | R1.4 ist kritisch |
| R1.1 - R1.3 sind nicht kritisch | A1 | A2 |
| R1.1 ist kritisch | A3 | A4 |
| R1.2 ist kritisch | | |
| R1.3 ist kritisch | | |

**Äquivalenzklasse A1: R1.1, R1.2, R1.3 und R1.4 sind nicht kritisch**

> **Interviewteilnehmer 13:** ”Wir versuchen früh, Wert zu schaffen in den Teams und uns nicht groß mit Anforderungsdefinition zu beschäftigen, sondern nur bis zu einem

Punkt, wo wir sagen, ok, wir wissen jetzt genug, um loszulaufen. Und wir haben eine Gesamtvorstellung, in welchem Umfang ungefähr, in welchen Dimensionen sich dieses Ding bewegt. Und wir laufen damit einfach los."

**Interviewteilnehmer 01:** "So einen gewissen Detailierungsgrad benötigen Sie [bei der Anforderungsanalyse] und der muss so gut sein, dass Sie in der Lage sind, auf der Basis eine initiale Aufwandsschätzung abzugeben, um dann quasi das Projekt zu beauftragen und anzugehen. [...] Mein Ziel ist es am Anfang eines Projektes, die Anforderungen zumindest auf einem Detaillevel zu verstehen, sodass ich sie A verstanden habe und B, dass ich in der Lage bin, zumindest mal eine grobe Indikation zu geben, wie aufwändig die Umsetzung wäre. Das heißt ich habe mal so eine grobe Idee, was das denn IT technisch bedeutet, was ich machen muss. Dann kann ich das auch entsprechend abschätzen."

**Äquivalenzklasse A2: R1.1, R1.2, R1.3 sind nicht kritisch und R1.4 ist kritisch**

**Interviewer:** "Wie viel Anforderungsanalyse muss mindestens durchgeführt werden?"
**Interviewteilnehmer 08:** "Ich würde sagen so, dass man einen kompletten Anwendungsfall modelliert hat, mit dem man etwas machen kann. Das ist wahrscheinlich der Erfolgsfall. Damit würde ich mindestens anfangen."

**Interviewer:** "Welche Informationen müssen zu Beginn mindestens gesammelt werden?"
**Interviewteilnehmer 12:** Eine Vision mit ein paar Features ist so der Idealfall. Die ein bisschen was konkret haben, aber lösungsneutral sind.

**Äquivalenzklasse A3: Mindestens ein Risiko von R1.1, R1.2 und R1.3 ist kritisch und R1.4 ist nicht kritisch**

**Interviewer:** "Was ist der Vorteil von Anforderungsanalyse zu Beginn?"
**Interviewteilnehmer 11:** "Bei Projekten, die intensiv in die vertikale Struktur einer Organisation eingreifen [...]. Da sind ganz viele Abteilungen dran, die sich daraus bedienen. Und da [ist es wichtig], die Anforderungen erstmal sauber aufsammeln zu können. Ich kann nicht sagen ich, fange mal an und setze was auf und dann mache ich nochmal eine Schleife und nochmal eine Schleife. Dann komme ich zu einem Punkt, wo ich feststelle, verdammt, vier Schleifen vorher habe ich etwas falsch konfiguriert. Erstmal alles auf null und alles von vorne. Und das funktioniert einfach

bei diesen großen konfigurierbaren Systemen nicht. Da muss man erstmal wirklich in die Analysephase gehen. Die Anforderungen festhalten, einen RE Prozess durchführen und dann anfangen iterativ zu implementieren. [...] Du bekommst halt diese Sicherheit, besonders in konfigurationsträchtigen Systemen, dass du vorab Konfigurationen nicht fehlerhaft betreibst oder mit falschen Annahmen."

**Äquivalenzklasse A4: Mindestens ein Risiko von R1.1, R1.2 und R1.3 ist kritisch und R1.4 ist kritisch**

**Interviewteilnehmer 11:** "Besonders wenn das so große Projekte sind mit vielen Stakeholdern. Die bekommst du ja nie sauber zusammen. Dann sucht man halt irgendwo den kleinsten gemeinsamen Nenner, wo sich alle drauf verständigen."

**Interviewteilnehmer 01:** "Dann [nach der Erhebung der Anforderungen] geht natürlich eine Diskussion los. Das muss man als Product Owner eben auch abkönnen, dass dann eben das Gefeilsche losgeht. Also türkischer Basar ist ein Anfänger dagegen, weil eben jeder der Stakeholder versucht irgendwie seine Anforderungen möglichst irgendwie noch unterzubekommen."

**Interviewteilnehmer 08:** "Wir haben das [Anforderungsanalyse] Anhand von Prototypen gemacht und vorher aufgeschrieben, was es überhaupt für Anwendungsfälle gibt. [...] Die haben geholfen, erstmal etwas Vorzeigbares zu machen, damit man mit den Leuten richtig sprechen konnte. Mit ganz vielen Leuten hätte man vorher stundenlang erklären können, was ein Identidy Access Management ist [...]. Da [mit den Prototypen] war das Verständnis dann schnell da."

**Interviewteilnehmer 09:** "Wenn ich den Kunden etwas frage und habe dazu kein konkretes Beispiel, dann wird er immer nur relativ vage bleiben."

**Kontext: Viele Stakeholder**

**Interviewteilnehmer 01:** "Ich habe aber auch schon so adaptierte Modelle gesehen, weil manchmal einfach zu viele Stakeholder da sind und ich vielleicht da so Mittelsmänner brauche und die dann genau solche Anforderungen aufnehmen und eben in die Backlogverwaltung gehen."

**Tabelle D.2:** Äquivalenzklassen bei der Architektur

| **Risiko von zu wenig Architekturdefinition** | **Risiko von zu viel Architekturdefinition** | |
|---|---|---|
| | **R2.4 ist nicht kritisch** | **R2.4 ist kritisch** |
| **R2.1 - R2.3 sind nicht kritisch** | D1 | D2 |
| **R2.1 ist kritisch** | | D3 |
| **R2.2 ist kritisch** | | |
| **R2.3 ist kritisch** | | |

## D.2 Architektur

### Äquivalenzklasse D1: R2.4 ist nicht kritisch

Waterman et al. [39, S.7]: ”Using Reducing Risk, a team designs the architecture in sufficient detail that it is comfortable that it is actually possible to build the system with the required ASRs [architecturally significant requirements ] with a satisfactory level of risk.”

**Interviewteilnehmer 12:** ”Man muss diese Langzeitplanung auf jeden Fall haben, damit man nicht nur von Tag zu Tag entwickelt, sondern auf Sichtweite. Du musst ja auch ungefähr wissen, wo es hingeht, damit du bestimmte Architekturentscheidungen jetzt schon abschätzen kannst.”

### Äquivalenzklasse D2: R2.1, R2.2 und R2.3 sind nicht kritisch und R2.4 ist kritisch

**Waterman et al. [39, S.8]:** ”S3 (EMERGENT ARCHITECTURE) produces an architecture in which the team makes only the minimum architecture decisions up-front, such as selecting the technology stack and the highest level architectural styles and patterns.”

**Interviewteilnehmer 02:** ”Das heißt, die [Architekturteam] haben so den Design und Architektur Gesamtüberblick. Die müssen gucken, dass das was da vorne hergestellt wird, auch gewissen Bedingungen genügt oder Richtlinien, die bei uns im Unternehmen zwingend eingehalten werden müssen, damit da kein Wildwuchs entsteht.”

**Äquivalenzklasse D3: Mindestens eines der R2.1, R2.2 oder R2.3 ist kritisch und R2.4 ist kritisch**

> **Waterman et al. [39, S.7]:** ”Using Reducing Risk, a team designs the architecture in sufficient detail that it is comfortable that it is actually possible to build the system with the required ASRs [architecturally significant requirements ] with a satisfactory level of risk.”
>
> **Dingsor et al. [91, S.17]:** ”An important benefit of the architecture was that it helped to separate the work of different teams into different parts of the code. In Perform, we found that the software architecture for the most part allowed the teams to work autonomously and effectively.”
>
> **Interviewteilnehmer 11:** ”Deswegen gibt es ja auch den API-first Ansatz, dass man sagt, na gut, ich nagle die Anforderungen und die Kommunikation der Systeme auf Basis des Interfaces fest. Wenn ich das erstmal beschreibe und Aufwand reinstecke, dann ist mir eigentlich egal, was die Systeme links und rechts machen. Das ist das, was die [Entwickerteams] brauchen und dann können die schon miteinander.”
>
> **Interviewteilnehmer 02:** ”Das heißt, die [Architekturteam] haben so den Design- und Architektur-Gesamtüberblick. Die müssen gucken, dass das was da vorne hergestellt wird, auch gewissen Bedingungen genügt oder Richtlinien, die bei uns im Unternehmen zwingend eingehalten werden müssen, damit da kein Wildwuchs entsteht.”

## D.3 Planung

**Äquivalenzklasse P1 und P2**

> **Interviewteilnehmer 01:** ”Der wesentliche Unterschied zu einem klassischen Projekt ist, dass Sie im Agilen eben nicht planen, wir wollen bis dann und dann die Inhalte zu hundert Prozent umgesetzt haben, sondern Sie sagen am Anfang, ok, das dauert Pi mal Auge, sagen wir mal anderthalb Jahre und das sind irgendwie so die Skills, die wir benötigen. [...] Und wenn man dann eine Entscheidung trifft und sagt, ok, da fallen dann Dinge raus, die aber super wichtig sind, dann müsste ich natürlich dann einen entsprechenden Change beantragen und sagen, ok, ich brauche die Leute noch ein bisschen länger.[...] Wenn Sie eine Produktverantwortung

**Tabelle D.3:** Äquivalenzklassen bei der Planung

| **Risiko von zu wenig Langzeitplanung** | **Risiko von zu viel Langzeitplanung** | |
|---|---|---|
| | **R1.4 ist nicht kritisch** | **R1.4 ist kritisch** |
| **R3.1 - R3.2 sind nicht kritisch** | P1 | |
| **R1.1 ist kritisch** | P2 | |
| **R3.2 ist kritisch (Existenz von Deadlines)** | P3 | P4 |
| **R3.2 ist kritisch (Bindung von Zeit, Budget und Umfang)** | | P5 |

haben, im Sinne von, das ist jetzt mein kontinuierlicher Job und ich habe ein Team, was das kontinuierlich begleitet, dann müssen Sie natürlich eine Roadmap für das Produkt machen als Product Owner. Weil, Sie müssen ja auch innerhalb des Unternehmens sagen, ok, wie sieht eigentlich meine Vision aus für das Produkt und wie will ich diese Vision eigentlich operationalisieren."

**Adelakun et al. [77, S.7]:** "What we find in the case of hybrid processes is that these constraints are loosely defined at the start of the project for the PM. The PM is to monitor these constraints, mitigate the expansion of any one area to the point that it would lead to customer dissatisfaction or violate a contract."

**Interviewteilnehmer 10:** "So zwei, drei Sprints in die Zukunft geplant ist schon Minimum, so dass dann schon relativ klar ist, welche Items folgen werden."

**Äquivalenzklasse P3 und P4: Das Risiko 3.2 ist kritisch und R3.3 ist nicht kritisch**

**Interviewteilnehmer 02:** "Der Scope kann im Rahmen innerhalb dieser vier Jahre angepasst werden, also in der Reihenfolge. Als Beispiel ganz aktuell haben wir ja Dinge schon umgesetzt, die dann im Juni aktiv geschaltet werden und wir hatten am Anfang teilweise Sachen im Scope drin für diesen ersten Teilschritt, die wir dann aber auf Grund der agilen Vorgehensweise und der gewonnenen Erkenntnisse rausgenommen haben. Wo wir gesagt haben, das machen wir jetzt später, dafür ziehen wir etwas anderes vor."

**Interviewteilnehmer 02:** "Natürlich werden diese regelmäßig überprüft. Gerade jetzt wo man ganz am Anfang steht, hat man ja dann Erkenntnisse, die man gewinnt und wo man gegensteuern muss."

**Interviewteilnehmer 02:** ”Die haben ihre Daylies und wie das alles heißt und von der Projektleitung her haben wir es so organisiert, dass wir dann regelmäßig mit den Teilprojektleitern sogenannte Teammeetings oder so Surfixe machen, wo wir dann im Grunde abfragen, wo die stehen. Dann müssen die das halt berichten, was sie gerade tun, wo sie gerade stehen, ob sie gewisse Termine, Meilensteine erreicht haben. Wenn nicht, wenn es Probleme gibt, dann müssen sie die benennen und müssen auch Maßnahmen vorschlagen, wie das Problem behoben werden kann, und das machen wir regelmäßig.”

**Äquivalenzklasse P5: Das Risiko 3.2 ist kritisch (Bindung von Zeit, Umfang und Budget) und R3.3 ist kritisch**

**Interviewteilnehmer 03:** ”Bei uns ist erstmal alles Change Request, was zu einer Produktänderung führt, und das fahren wir auch nach wie vor so weiter. Also wir fahren Richtung Kunden nach wie vor Wasserfall, das muss man auch sagen. [. . . ] Letztendlich hat man immer das Problem, dass sich das vertrieblich, sprich mit dem Einkauf nicht vereinbaren lässt. Wo wir dann auf den klassischen Wasserfall beim Kunden zurückgreifen.”

**Interviewteilnehmer 03:** ”Das heißt immer, so gegen Ende der Entwicklung eines Releases bauen wir schon die Release Angebote zusammen für die Dinge, die wir im nächsten Jahr gebaut haben wollen. Sodass wir im Zyklus der Budgetphase der Kunde die Aufträge bekommen und rechtzeitig dann auch die neuen Change Requests fürs neueste Release [...] einphasen können.”

**Interviewteilnehmer 13:** ”Du hast mit einem dreimonatigen Vorlauf tatsächlich so einen Planungszyklus. Du hast quasi eine Idee und schleust die da ein. In den ersten drei Monaten wird dort eingeplant, dass dort jemand sich mal deine Idee genauer anschaut und mit dir zusammen anschaut, wie groß ist der Aufwand. Und wenn du den Block definiert hast, dann kann der für den nächsten Cycle mit eingeplant werden. Und da hättest du dann so einen dreimonatigen Vorlauf. [...] Da hängen die Teams [von anderen Firmen] dran und die bekommen es nicht hin sich so konsequent auf das Thema agil einzulassen, wie wir das tun. Daher ist das für uns ein Kompromiss an dieser Schnittstelle, dass wir uns auf ihren Prozess mit einlassen.”

**Interviewteilnehmer 03:** ”Mit der Anforderungsdefinition geht dann dieser Change Request an die Entwicklung. Die Entwicklung macht daraus eine Lösungsbeschreibung und wenn die Lösungsbeschreibung dann von uns bestätigt wird, also vom

PM [Product Manager] bestätigt wird, dann kann die Schätzung erfolgen. Also die Aufwandsschätzung, wie teuer es wird oder wie viel Aufwand es wird."

## D.4 Testen

**Tabelle D.4:** Äquivalenzklassen beim Testen

| **Risiko von zu wenig Trennung beim Testen** | **Risiko von zu viel Trennung beim Testen** |
|---|---|
| | **R4.2** |
| **R4.1 ist nicht kritisch** | T1 |
| **R4.1 ist kritisch (kleines Projektteam)** | T2 |
| **R4.1 ist kritisch (Großes und/oder verteiltes Projektteam)** | T3 |

#### Äquivalenzklasse T1: R4.1 ist nicht kritisch

**Interviewteilnehmer 13:** "Wir machen Test Driven Development und alle Tests müssen im Team mitgeschrieben werden. Tests werden automatisch durchgeführt. Wenn du Code in drei Monaten depolyst und ein Fehler auftritt, dann weißt du nicht mehr, was du da gemacht hast. Komplette DevOps Verantwortung. Die Verantwortung für den Code liegt im Team [...]."

#### Äquivalenzklasse T2 und T3: R4.1 ist kritisch

**Interviewteilnehmer 13:** "Die Entwicklungsteams wollen gerne schnell deployen, aber wir haben so ein Testteam, das die noch integrativ testet, dazwischen. Was dann doch seine Hand darauf legt und versucht, den Releaseprozess zu greifen und den in eine gelenkte Bahn zu führen."

**Interviewteilnehmer 01:** "Stellen Sie sich mal vor, Sie haben einen N-to-N Prozess der 7 oder 8 Softwareprodukte beinhaltet. Wem wollen Sie da die Verantwortung geben, können Sie gar nicht. Also brauchen Sie irgendwo ein Team, das sich dann quasi darum kümmert, dass dieser [Test] Prozess von vorne nach hinten durchläuft."

**Interviewteilnehmer 12:** ”Ein Vorteil ist [von einem extra Testteam], du weißt ja nicht, wenn du es ausrollst, ob es noch mit den anderen funktioniert. Wenn du da nicht ordentlich getestet hast oder mit den anderen kommunizierst, dann ist natürlich so ein Testteam Gold wert, das das Gesamte so ein bisschen kanalisiert.”

#### Kontext: Eingebettete Software

**Interviewteilnehmer 06:** ”Wir testen dann immer zu den jeweiligen Quartalsreleases und auch am Ende nochmal sehr groß. Dann fahren wir damit auch mal auf der Straße. Das ist dann nicht mehr spezifikationsbasiert, sondern wirklich Systemtests.[...] Vor der Serie, wird das dann nochmal alles intensivst durchgetestet.”

## D.5 Koordination

**Tabelle D.5:** Äquivalenzklassen für die Koordination

| Risiko von zu wenig zentraler Koordination | Risiko von zu viel zentraler Koordination |
|---|---|
| | **R5.3** |
| **1-2 Entwicklerteams und keine Verteilung** | K1 |
| **Großes und/oder verteiltes Projektteam** | K2 |

#### Äquivalenzklasse K2: Große und/oder verteilte Entwicklung

**Interviewteilnehmer 12:** ”Damit sich mehrere Teams miteinander abstimmen, werden Scrum of Scrums genutzt. Es gibt Entwicklerteams, die ihr Stand up zusammen haben.”

**Interviewteilnehmer 12:** ”Ansonsten haben wir die PI Plannings, die dienen auch der Koordination. Am Ende des PI Plannings fällt ein Programm Board raus, wo wir die Abhängigkeiten zwischen den einzelnen Teams machen. Welches Feature des einen Teams ist abhängig vom Feature des anderen Teams.”

**Interviewteilnehmer 05:** ”Und wir sprachen auch über Rückfragen, Abstimmungen, Ergänzungen, Change und und. Und diese Zeit nenne ich mal Rüstzeit. Und diese Rüstzeit müsste man sich eigentlich vor der Entwicklerphase gönnen. Das tut man aber nicht. Sondern man verlagert diese Rüstzeit der Abstimmung in die

Entwicklerphase. Und das wiederum heißt defacto, dass du eigentlich, wenn du, ich sage mal, von der gesamten Entwicklerphase die Rüstzeitphase abziehst, nur noch ein Bruchteil der Entwicklerphase übrighast."

**Bick et al. [32, S.16]:** "The inter-team counterparts of these standard team-level [Sprint Planning, Daily, etc.] agile meetings should include informed representatives from all teams concerned in addition to the members of the central team. This should establish a two-way feedback channel between the individual teams on the one hand and the central team on the other, as well as a platform for all teams to become aware of the existing and potential dependencies between them."

# E

# Aussagen und Erklärungen in den Äquivalenzklassen des APH-Frameworks

## E.1 Requirements Engineering

**Anforderungsanalyse zu Beginn des Projektes - Tätigkeiten**

- Analyse der Stakeholder
- Analyse des Projektgrundes und Erstellung einer Vision
- Analyse der Anwendungsfälle der Nutzer und Stakeholder
- Erstellung und Priorisierung eines Produkt Backlogs
- Analyse der nicht-funktionalen Anforderungen
- Analyse von existierenden Einschränkungen (Standards, Abhängigkeiten und ähnliches)
- Analyse des Anforderungsumfanges

- Schätzung der Dauer und des benötigten Budgets für das Projekt

**Kontinuierliche Anforderungsanalyse - Tätigkeiten**

- Aufnahme von neuen oder geänderten Anforderungen
- Aufwandsschätzung von neuen Anforderungen oder Änderungen durch die Entwicklerteams
- Weiterführung und Pflege des Backlogs (Schreiben von User Stories / Verfeinern von Anforderungen)
- Refinements mit den Entwicklerteams
- Planung der Entwicklung (Zusammenentstellung von Sprint Backlogs / Weiterführung der Roadmap / Zusammenstellung von Iterationen)

### E.1.1 Äquivalenzklasse A1

**Anforderungsanalyse zu Beginn des Projektes**

A1.1: Das Projektteam sollte die Anforderungen soweit erheben, bis es ein gemeinsames Verständnis der Anforderungen erreicht hat. Dies ist erreicht, wenn das Projektteam die Zeit und das Budget für das Projekt zu einem geplanten Umfang abgeben kann und gemeinschaftlich der Meinung ist, genug Informationen zu haben, um mit dem Projekt starten zu können. Das Projektteam sollte anschließend in ein kontinuierliches Requirements Engineering übergehen.

Erklärung: Um die Anforderungsanalyse zu Beginn des Projektes nicht unnötig zu verlängern, sollte das Projektteam die Anforderungsanalyse beenden, wenn das Projektteam eine Schätzung abgeben kann.

### E.1.2 Äquivalenzklasse A2

**Anforderungsanalyse**

A2.1: Das Projekt sollte sich bei der Analyse der Anforderungen auf die Schaffung einer Vision und die Analyse der nötigsten Anwendungsfällen und deren Funktionen oder auf die komplette Beschreibung eines Anwendungsfalls beschränken. Danach sollte das Projektteam in ein kontinuierliches Requirements Engineering übergehen.

Erklärung: Durch die geringe Bekanntheit der Anforderungen ist das Risiko groß, dass während der Anforderungsanalyse falsche Anforderungen aufgenommen werden. Daher sollte sich das Projektteam auf die Definition einer Vision und weniger Anwendungsfälle beschränken und frühzeitig in ein kontinuierliches Requirements Engineering übergehen.

Erklärung: Die Risiken, zu Beginn zu wenig Anforderungsanalyse zu betreiben, sind gering. Daher ist es dem Projektteam möglich, frühzeitig in ein kontinuierliches Requirements Engineering überzugehen.

**Kontinuierliche Anforderungsanalyse**

A2.2: Während des Projektes sollte sich das Projektteam auf die Aufnahme von neuen Anforderungen und die Durchführung von Feedbackschleifen konzentrieren.

Erklärung: Zu Beginn des Projektes nimmt das Projektteam nur die minimalen Anforderungen auf. Daher müssen die meisten Anforderungen während der Entwicklungs aufgenommen werden. Feedbackschleifen unterstützten dabei, dass bei einer geringen Bekanntheit der Anforderungen keine falschen Anforderungen entwickelt werden.

### E.1.3 Äquivalenzklasse A3

**Anforderungsanalyse zu Beginn eines Projektes**

A3.1: Das Projektteam sollte sich darauf fokussieren einen, kompletten Überblick über die Anwendung zu erreichen. Dies beinhaltet das Erstellen einer klaren Vision, die Analyse der Anwendungsfälle und deren Funktionen und die Abhängigkeiten im Projekt.

Erklärung: Das große und/oder verteilte Projektteam und /oder die Existenz von Abhängigkeiten erschweren die Koordination im Projekt. Ein Überblick über die gesamte Anwendung hilft einem Projektteam bei der Koordination von Entwicklerteams.

Erklärung: Die hohe bis mittlere Bekanntheit der Anforderungen ermöglicht es dem Projektteam, Anforderungen detaillierter zu analysieren.

R1.2 ist kritisch $\rightarrow$ A3.2: Das Projektteam sollte die Anwendungsfälle und deren Funktionen ausreichend untersuchen und deren Umsetzung verstehen, bis sie genug Informationen haben, eine Schätzung darüber abzugeben.

Erkläurung: Der Kunde hat ein starkes Bedürfnis danach, dass die Schätzung von Zeit und Budget möglichst korrekt ist. Um diese abzugeben, muss das Projekt ein ausreichendes Verständnis der Anwendungsfälle und deren Funktionen gewinnen.

R1.3 ist kritisch → A3.3: Das Projektteam sollte einen Fokus auf die Analyse von nicht-funktionalen Anforderungen zu Beginn des Projektes legen.

Erklärung: Das Projekt hat viele nicht-funktionale Anforderungen, die zu Beginn berücksichtigt werden müssen, damit sie umgesetzt werden können. Daher ist es wichtig, diese Anforderungen direkt zu Beginn zu analysieren.

A3.4: Wenn das Projektteam gemeinsam feststellt, dass es genug Informationen hat, um mit dem Projekt starten zu können, dann sollte das Projektteam die Anforderungsanalyse abbrechen und in ein kontinuierliches Requirement Engineering übergehen.

Erklärung: Trotz der hohen Bekanntheit der Anforderungen ist es immer noch möglich, dass sich Anforderungen ändern. Daher sollte das Projektteam, soweit es möglich ist, Freiheiten bei den Anforderungen lassen und nach Erreichung von genügend Informationen in ein kontinuierliches Requirements Engineering übergehen.

### E.1.4 Äquivalenzklasse A4

**Anforderungsanalyse zu Beginn eines Projektes**

A4.1: Das Projektteam sollte zu den Standardtasks während der Anforderungsanalyse zusätzlich Negotiationstechniken und Prototyping einsetzen, um ein größeres Verständnis und Bekanntheit der Anforderungen zu erreichen.

R1.1 ist kritisch → Erklärung: Das große und/oder verteilte Projektteam und Abhängigkeiten erschweren die Koordination im Projekt. Ein höheres Verständnis und Bekanntheit der Anforderungen erleichtern die Koordination eines Projektes.

R1.2 ist kritisch → Erklärung: Der Kunde hat ein starkes Bedürfnis danach, dass die Schätzung von Zeit und Budget möglichst korrekt ist. Ein höheres Verständnis und Bekanntheit der Anforderungen unterstützt ein Projektteam dabei, eine möglichst genaue Schätzung abzugeben.

R1.3 ist kritisch → Erklärung: Das Projekt hat viele nicht-funktionale Anforderungen, die zu Beginn berücksichtigt werden müssen, damit sie umgesetzt werden können. Es ist daher wichtig, dass das Projektteam diese Anforderungen zu Beginn kennt. Negotiationstechniken können dabei unterstützen, diese Anforderungen zu bestimmen.

A4.2: Wenn das Projektteam gemeinsam feststellt, dass sie ausreichend genug Informationen erhoben haben, um mit dem Projekt starten zu können, sollte es die Anforderungsana-

lyse abbrechen und in ein kontinuierliches Requirements Engineering übergehen. Ein Indikator dafür ist, wenn das Projektteam eine Schätzung für das Projekt abgeben kann.

Erklärung: Dadurch dass die Anforderungen eine geringe Bekanntheit haben, ist die Wahrscheinlichkeit groß, dass während der Anforderungsanalyse falsche Anforderungen aufgenommen werden. Um dieses Risiko zu minimieren, sollte das Projektteam nicht mehr Informationen aufnehmen, als sie benötigen, um mit dem Projekt starten zu können.

**Kontinuierliche Anforderungsanalyse**

A4.3 Während des kontinuierlichen Requirements Engineering sollte das Projektteam einen Fokus auf Feedbackschleifen legen.

Erklärung: Durch die Anwendung von Negotiotionstechniken und Protoyping bei einer geringen Bekanntheit der Anforderungen, kann es passieren, dass falsche Anforderungen aufgenommen werden. Daher ist es wichtig, dass das Projektteam während des kontinuierlichen Requirements Engineerings einen Fokus auf Feedbackschleifen legt.

### E.1.5 Kontext: Viele Stakeholder

Das Projektteam sollte zusätzlich aus einem Team von Business Experten bestehen, das die Product Owner während des kontinuierlichen Requirements Engineering unterstützt.

Erklärung: Ein Team von Business Experten untersützt dabei, eine große Menge an Stakeholdern zu verwalten.

## E.2 Architektur

### E.2.1 Architekturdesign zu Beginn eines Projektes - Tätigkeiten

- Festlegung des grundlegenden Aufbaues der Software (z.B. Schichtenarchitektur)
- Festlegung des Technologiestacks (Verwendeten Technologien)

### E.2.2 Kontinuierliche Architekturdefinition - Tätigkeiten

- Treffen von Architekturentscheidungen auf Projektebene (zum Beispiel für die Umsetzung von Anforderungen, die für die gesamte Software gelten)

- Verwaltung von Schnittstellen und Abhängigkeiten zwischen Entwicklerteams und zu anderen Projekten
- Unterstützung der Entwicklerteams bei der Entwicklung der Architektur

### E.2.3 Äquivalenzklasse D1

**Architekturdesign zu Beginn eines Projektes**

D1.1: Das Projektteam sollte die nötigen Architekturentscheidungen soweit treffen, bis es zuversichtlich ist, mit dem erreichten Stand der Architekturdefinition das Projekt entwickeln zu können. Dies bezieht die Definition von Komponenten und Schnittstellen und die Berücksichtigung von nicht-funktionalen Anforderungen, die für eine Umsetzung gleich zu Beginn des Projektes berücksichtigt werden müssen, mit ein. Zusätzlich sollten die möglichen Abhängigkeiten zu anderen Projekten berücksichtigt werden. Darüber hinaus kann das Projektteam zusätzlich einen groben Plan über die Weiterentwicklung der Architektur machen. Danach sollte die Architekturdefinition abgebrochen werden und das Projekt sollte in eine kontinuierliche Definition der Architektur übergehen.

Erklärung: Durch die hohe Bekanntheit der Anforderungen ist die Wahrscheinlichkeit gering, dass Teile der Architektur geändert werden müssen. Das Projektteam ist also in der Lage, die Architektur soweit zu definieren, bis es zuversichtlich ist mit dem Projekt starten zu können. Es ermöglicht dem Projektteam ebenfalls, einen groben Plan für die Weiterentwicklung Architektur zu machen.

D1.2: Um genug Freiheiten bei der Architekturdefinition zu behalten, sollten die weiteren Architekturentscheidungen zurückgehalten werden und nur als Plan vorliegen.

Erklärung: Die reine Planung von weiteren Architekturentscheidungen ermöglicht es diese noch zu ändern. Gleichzeitig gibt es dem Projektteam Stabilität darüber, wie die Architektur sich weiter entwickelt. Dies unterstützt bei der Koordination von Entwicklerteams und dabei, dass sich die Architektur nicht zufällig entwickelt.

### E.2.4 Äquivalenzklasse D2

**Architekturdesign zu Beginn eines Projektes**

D3.1: Das Projektteam sollte sich nur darauf fokussieren, die grundlegenden Architekturpattern und die verwendete Technologie zu bestimmen und zu definieren.

D3.2 Das Projektteam sollte alle weiteren Architekturentscheidungen während der Architekturdefinition zurückhalten und erst während der Entwicklung des Projektes treffen.

Erklärung: Durch die geringe Bekanntheit der Anforderungen sollten die Architekturentscheidungen soweit wie möglich zurückgehalten werden, um auf Änderungen reagieren zu können.

D3.3: Das Projekt benötigt keinen parallelen Prozess für die Entwicklung der Architektur. Stattdessen kann die Architektur von den Entwicklern entwickelt werden.

Erklärung: Durch das kleine Projektteam ist die Gefahr gering, dass sich die Architektur zwischen Entwicklerteams zufällig entwickelt. Zusätzlich benötigt das Projektteam die Architektur nicht zur Unterstützung der Koordination. Dadurch kann das Projektteam mit einer minimalen Anzahl an Architekturentscheidungen in das Projekt starten und die Entwicklung der Architektur kann in den Händen der Entwickler liegen.

### E.2.5 Äquivalenzklasse D3

**Architekturdesign zu Beginn eines Projektes**

D2.1: Das Projektteam sollte die minimalen Architekturentscheidungen treffen, um mit dem Projekt starten können. Danach sollte das Projektteam die Architekturdefinition abbrechen und in ein kontinuierliche Definition der Architektur übergehen.

Erklärung: Da die Anforderungen eine niedrige Bekanntheit haben, kann es sein, dass die Architektur während der Entwicklung geändert werden muss. Dafür ist es wichtig, dass das Projektteam Architekturentscheidungen soweit wie möglich zurückhält und sich Handlungsalternativen offenlässt.

D2.2: Das Projektteam sollte sich darauf konzentrieren, die Entwicklung von einzelnen Entwicklerteams durch die Definition von Schnittstellen zu kapseln und mögliche Abhängigkeiten zu anderen Projekten zu berücksichtigen.

Erklärung: Durch die geringe Bekanntheit der Anforderungen muss das Projekt flexibel sein. Gleichzeitig ist ein Überblick über die Architektur wichtig, damit die Architektur strukturiert wächst. Die Definition von Schnittstellen unterstützt den Überblick über eine Software, ein strukturiertes Wachsen der Architektur und gleichzeitig die Flexibilität von den Entwicklerteams.

D2.3: Das Projektteam sollte sich während des Architekturdesigns darauf fokussieren, verschiedene Optionen für die Weiterentwicklung der Architektur zu ermöglichen.

Erklärung: Da die Anforderungen gering bekannt sind, ist es wichtig, dass in der Architekturdefinition genug Optionen verbleiben, um auf Änderungen reagieren zu können.

R2.3 ist kritisch $\rightarrow$ D2.1 Das Projekt sollte sich darauf konzentrieren, die nicht-funktionalen Anforderungen zu berücksichtigen, die direkt zu Beginn des Projektes berücksichtigt werden müssen, damit diese umgesetzt werden können.

R2.3 ist kritisch $\rightarrow$ Erklärung: Das Projekt besitzt nicht-funktionale Anforderungen, die zu Beginn berücksichtigt werden müssen, damit diese umgesetzt werden können. Daher ist es wichtig, diese in der Architekturdefinition zu berücksichtigen.

### E.2.6 Kontext: R2.1, R2.2 oder R2.3 sind kritisch

**Kontinuierliche Architekturdefinition sollte genutzt werden**

Das Projektteam sollte sich während der kontinuierlichen Definition der Architektur darauf fokussieren, dass die Entscheidungen für die Weiterentwicklung der Architektur auf Projektebene getroffen werden.

Erklärung: Während der Architekturdefinition zu Beginn eines Projektes trifft das Projektteam nur die minimalen Architekturentscheidungen. Das heißt, der Großteil der Architektur entwickelt sich während des Projektes. Eine zentrale Steuerung der Architektur unterstützt dabei, dass sich die Architektur nicht zufällig weiterentwickelt und dass alle Anforderungen berücksichtigt werden.

## E.3 Planung

### E.3.1 Planung zu Beginn eines Projektes - Tätigkeiten

- Festlegung einer Roadmap zur Entwicklung
- Zusammenstellung des Projektteams

### E.3.2 Kontinuierliche Planung - Tätigkeiten

- Überprüfung des Projektstatus
- Verwaltung und Weiterentwicklung der Roadmap

### E.3.3 Äquivalenzklasse P1

**Planung zu Beginn eines Projektes**

P1.1: Das Projektteam kann eine Schätzung der Zeit und des Budgets für den geplanten Umfang des Projektes abgeben. Diese Schätzung sollte mit einem ausreichend großen Toleranzbereich versehen werden.

Erklärung: Die Kunden verlangen nur eine grobe Schätzung des Projektes, daher kann das Projekt eine Schätzung mit einem Toleranzbereich abgeben.

Erklärung: Da Änderungen während der Entwicklung auftreten können, sollte ein ausreichend großer Toleranzbereich gewählt werden.

P1.2: Innerhalb der groben Schätzung sollte es möglich sein, dass Anforderungen ausgetauscht oder fallen gelassen werden können. Wenn sich die grobe Schätzung im Laufe des Projektes durch eine Änderung verändert, sollte das Projektteam mit dem Kunden darüber entscheiden, ob ein Projekt länger geht oder andere Anforderungen fallen gelassen werden.

Erklärung: Dadurch dass Anforderungen ausgetauscht oder fallen gelassen werden können, kann das Projektteam auf Änderungen und Probleme reagieren.

P1.3: Das Projektteam sollte eine Roadmap erstellen und verwalten, die die Entwicklungsreihenfolge von Features plant. Das Projektteam sollte keine anderen Meilensteine in ihrer Planung nutzen, sondern die Roadmap zur Darstellung der Vision nutzen.

Erklärung: Das Projektteam ist nicht mit einer schwierigen Koordination, durch zum Beispiel ein großes oder verteiltes Projektteam konfrontiert, daher reicht es aus, wenn die Roadmap dazu dient die Vision für das Projekt zu verwalten und zu transportieren.

Erklärung: Da es während der Entwicklung zu Änderungen kommen kann, sollten in der Roadmap keine detaillierteren Meilensteine verwendet werden, damit das Projektteam schnell auf Änderungen reagieren kann.

P1.4: Die Arbeit von einzelnen Entwicklerteams sollte nicht durch die zentrale Roadmap geplant werden. Die Entwicklerteams planen ihre Arbeit durch die Befüllung ihrer Sprint Backlogs.

Erklärung: Die Planung in den einzelnen Entwicklerteams, sollte nicht zentral geschehen, da sichdie Entwicklung ansonsten zu stark verlangsamt und Entwicklerteams unflexibel werden.

### E.3.4 Äquivalenzklasse P2

**Planung zu Beginn eines Projektes**

P2.1: Das Projektteam kann eine Schätzung der Zeit und des Budgets für den geplanten Umfang des Projektes abgeben. Diese Schätzung sollte mit einem ausreichend großen Toleranzbereich versehen werden.

Erklärung: Die Kunden verlangen nur eine grobe Schätzung des Projektes, daher kann das Projektteam eine Schätzung mit einem Toleranzbereich abgeben.

Erklärung: Da Änderungen während der Entwicklung auftreten können, sollte ein ausreichend großer Toleranzbereich gewählt werden.

P2.2: Für den Umfang sollte es möglich sein, dass im Rahmen der Schätzung Features ausgetauscht oder fallen gelassen werden können.

P2.3: Das Projektteam sollte eine Roadmap erstellen und zentral verwalten, in der sie Abhängigkeiten innerhalb des Projektes und zu anderen Projekten verwaltet und die Entwicklungsreihenfolge von Features plant. Das Projektteam sollte keine feineren Meilensteine definieren, die zum Beispiel die Arbeit in den einzelnen Entwicklerteams planen. Die Entwicklerteams planen ihre Arbeit basierend auf der zentralen Roadmap durch eine kontinuierliche Planung ihrer Sprints.

Erklärung: Das Projekt hat möglicherweise ein großes und/oder verteiltes Projektteam und/oder Abhängigkeiten zu beachten. Die Koordination eines solchen Projektes kann durch eine Roadmap für Abhängigkeiten unterstützt werden.

Erklärung: Da es während der Entwicklung zu Änderungen kommen kann, sollte in der Roadmap keine detaillierteren Meilensteine verwendet werden, damit das Projektteam schnell auf Änderungen reagieren kann.

Erklärung: Die Planung in den einzelnen Entwicklerteams sollte nicht zentral geschehen, da sich die Entwicklung ansonsten zu stark verlangsamt und Entwicklerteams inflexibel werden.

### E.3.5 Äquivalenzklasse P3

**Planung zu Beginn eines Projektes**

P3.1: Das Projektteam kann eine Schätzung der Zeit und des Budgets für den geplanten Umfang des Projektes abgeben. Diese Schätzung kann mit einem möglichst kleinen Toleranzbereich versehen werden.

Erklärung: Die Kunden verlangen eine möglichst genaue Schätzung des Projektes, daher sollte das Projektteam sich bemühen, eine Schätzung mit einem möglichst kleinen Toleranzbereich abgeben zu können.

Erklärung: Da Änderungen oder Probleme während der Entwicklung auftreten können, sollte die Schätzung mit einem Toleranzbereich versehen werden.

Erklärung: Die mittlere bis hohe Bekanntheit der Anforderungen und der Technologie ermöglichen es dem Projektteam, eine möglichst genaue Schätzung durchzuführen.

P3.2: Das Projektteam sollte eine Roadmap erstellen, in der Deadlines und Abhängigkeiten zentral durch eine Projektleitung verwaltet werden.

P3.3: Für die Planung von Deadlines sollte das Projektteam kontinuierlich analysieren, welche Anforderungen für die Deadlines umgesetzt werden müssen und wie viel Zeit die Umsetzung der Anforderungen kostet.

Erklärung: In dem Projekt gibt es Deadlines, zu denen der Kunde gewisse Anforderungen geliefert haben möchte. Daher muss das Projekteam beständig planen, welche Anforderungen die Deadlines enthalten und wie lange das Projektteam für die Umsetzung der Anforderungen benötigt. Dies ist wichtig, damit das Projektteam rechtzeitig mit der Entwicklung der Anforderungen anfängt und die Deadlines erfüllt.

P3.4: Das Projekt sollte nur die Meilensteine auf Projektebene planen, die für die Einhaltung von Deadlines nötig sind.

Erklärung: Zu viele und detaillierte Meilensteine verursachen viel Aufwand bei Änderungen und machen ein Projekt unflexibel. Das Projektteam sollte daher die geplanten Meilensteine auf die nötigsten beschränken.

P3.5: Die Arbeit von einzelnen Entwicklerteams sollte nicht durch die zentrale Roadmap geplant werden. Die Entwicklerteams planen ihre Arbeit basierend auf der zentralen Roadmap durch die Befüllung ihrer Sprint Backlogs.

Erklärung: Die Planung in den einzelnen Entwicklerteams sollte nicht zentral geschehen, da sich die Entwicklung ansonsten zu stark verlangsamt und Entwicklerteams unflexibel werden.

P3.6: In dem Umfang für Deadlines sollten flexible Anforderungen enthalten sein. Solche Anforderungen können vom Projektteam vorgezogen, verschoben oder komplett fallen gelassen werden.

Erklärung: Während der Entwicklung kann es immer zu Problemen oder zu Änderungen kommen. Flexible Anforderungen können bei Problemen verschoben oder fallen gelassen werden. Flexible Anforderungen unterstützen daher dabei, dass Deadlines eingehalten werden können.

#### Kontinuierliche Planung

P3.7: Während der Laufzeit des Projektes sollte das Projektteam einen Fokus auf die Überwachung des Projektstatus legen. Basierend auf dem Projektstatus sollte das Projektteam beständig analysieren, welche der flexiblen Anforderungen in der Entwicklung vorgezogen, verschoben oder fallen gelassen werden müssen.

Erklärung: Die Überwachung des Projektstatus ist wichtig, um zu überprüfen, ob die gesetzten Deadlines mit dem Umfang umgesetzt werden können. Die kontinuierliche Überprüfung des Status ist wichtig, damit das Projektteam rechtzeitig entscheiden kann, welche Anforderungen verschoben oder sogar fallen gelassen werden sollten.

### E.3.6 Äquivalenzklasse P4

#### Planung zu Beginn eines Projektes

P4.1: Das Projektteam kann eine Schätzung der Zeit und des Budgets für den geplanten Umfang des Projektes abgeben. Diese Schätzung sollte mit einem ausreichend großen Toleranzbereich versehen werden.

Erklärung: Die Kunden verlangen nur eine grobe Schätzung des Projektes, daher kann das Projektteam eine Schätzung mit einem Toleranzbereich abgeben.

Erklärung: Da Änderungen während der Entwicklung auftreten können, sollte ein ausreichend großer Toleranzbereich gewählt werden.

P4.2: Das Projektteam sollte zu den Standardtätigkeiten während der Anforderungsanalyse zu Beginn eines Projektes zusätzlich Negotiationstechniken und Prototyping einsetzen, um ein größeres Verständnis und Bekanntheit der Anforderungen zu erreichen.

P4.3: Das Projektteam sollte eine Roadmap erstellen, in der Deadlines und Abhängigkeiten zentral verwalten werden.

P4.4: Für die Planung von Deadlines sollte das Projektteam kontinuierlich analysieren, welche Anforderungen für die Deadlines umgesetzt werden müssen und wie viel Zeit die Umsetzung der Anforderungen kostet.

Erklärung: In dem Projekt gibt es Deadlines zu denen der Kunde gewisse Anforderungen geliefert haben möchte. Daher muss das Projektteam beständig planen, welche Anforderungen die Deadlines enthalten und wie lange das Projektteam für die Umsetzung der Anforderungen benötigt. Dies ist wichtig, damit das Projektteam rechtzeitig mit der Entwicklung der Anforderungen anfängt und die Deadlines erfüllt.

P4.5: Das Projekt sollte nur die Meilensteine planen, die für die Einhaltung von Deadlines nötig sind.

P4.5: Die Arbeit von einzelnen Entwicklerteams sollte nicht durch die zentrale Roadmap geplant werden. Die Entwicklerteams planen ihre Arbeit durch die Befüllung ihrer Sprint Backlogs.

Erklärung: Die Planung in den einzelnen Entwicklerteams, sollte nicht zentral geschehen, da sich die Entwicklung ansonsten zu stark verlangsamt und Entwicklerteams unflexibel werden.

P4.6: In dem Umfang für Deadlines sollten flexible Anforderungen enthalten sein. Solche Anforderungen können vom Projektteam vorgezogen, verschoben oder komplett fallen gelassen werden.

Erklärung: Während der Entwicklung kann es immer zu Problemen oder zu Änderungen kommen. Flexible Anforderungen können bei Problemen verschoben oder fallen gelassen werden. Flexible Anforderungen unterstützen daher dabei, dass Deadlines eingehalten werden können.

**Kontinuierliche Planung**

P4.7: Während der Laufzeit des Projektes sollte das Projektteam einen Fokus auf die Überwachung des Projektstatus legen. Basierend auf dem Projektstatus sollte das Projektteam beständig analysieren, welche der flexiblen Anforderungen in der Entwicklung vorgezogen, verschoben oder fallen gelassen werden müssen.

Erklärung: Die Überwachung des Projektstatus ist wichtig, um zu überprüfen, ob die gesetzten Deadlines mit dem Umfang umgesetzt werden können. Die kontinuierliche Überprüfung des Status ist wichtig, damit das Projektteam rechtzeitig entscheiden kann, welche Anforderungen verschoben oder sogar fallen gelassen werden sollten.

### E.3.7 Äquivalenzklasse P5

**Planung zu Beginn eines Projektes**

P5.1: Das Projektteam sollte nur eine grobe Schätzung für das gesamte Projekt durchführen. Negotiationstechniken und Prototyping können dabei unterstützen eine solche Schätzung durchzuführen.

Erklärung: Die geringe Bekanntheit der Anforderungen und/oder de Technologie lassen nur eine grobe Schätzung des gesamten Projektes zu. Da der Kunde eine möglichst genaue Schätzung des Projektes benötigt, kann die Bekanntheit der Anforderungen und der Technologie durch Negotiationstechniken und Prototyping verbessert werden.

P5.2 Das Projektteam sollte die Entwicklung in Iterationen aufteilen, für die der Umfang genau festgelegt wird. Für die Iterationen können mit dem Kunden einzelne Verträge geschlossen werden.

Erklärung: Der Kunde verlangt eine Bindung von Umfang, Preis und Zeit für Features. Durch die niedrige Bekanntheit der Anforderungen und/oder der Technologie ist es aber schwierig, diese Bindung für ein ganzes Projekt zu gewährleisten. Es ist daher von Vorteil, kontinuierlich Verträge über einzelne Iterationen zu erstellen, da diese einen überschaubaren Zeitraum darstellen.

P5.3: Das Projektteam sollte eine Roadmap erstellen, in der sie Deadlines und Abhängigkeiten zentral verwaltet.

Erklärung: Die Verwendung einer Roadmap ist wichtig, damit das Projektteam planen kann, wann sie mit der Entwicklung von bestimmten Anforderungen beginnen muss, um Deadlines und Iterationsziele einzuhalten.

P5.4: Die Arbeit von einzelnen Entwicklerteams sollte nicht durch die zentrale Roadmap geplant werden. Die Entwicklerteams planen ihre Arbeit durch die Befüllung ihrer Sprint Backlogs.

Erklärung: Die Planung in den einzelnen Entwicklerteams, sollte nicht zentral geschehen, da sich die Entwicklung ansonsten zu stark verlangsamt und Entwicklerteams unflexibel werden.

#### Kontinuierliche Planung

P5.5: Während des kontinuierlichen Requirements Engineering sollte ein Fokus auf der Schätzung von neuen Anforderungen gelegt werden. Dabei analysieren Entwicklerteams neue Anforderungen und geben eine Schätzung über die benötigte Dauer und Budget für die Umsetzung der Features an.

Erklärung: Die kontinuierliche Schätzung von neuen Anforderungen ist wichtig, damit das Projektteam bestimmen kann, wie viel es in einer Iteration genau umsetzen kann. Da der Umfang einer Iteration vertraglich festgelegt ist, ist dies wichtig.

## E.4 Testen

### E.4.1 Äquivalenzklasse T1

#### Durchführung der Tests direkt nach Fertigstellung von Features

T1.1 Die Testverantwortung für Integrations- und Systemtests liegt in den Händen der Entwickler. Die Entwickler können danach selbstständig die Features deployen.

Erklärung: Es besteht kein Risiko bei der Integration einzelner Features, daher können die Integration- und Systemtests direkt von den Entwicklern durchgeführt werden.

T1.2 Das Projektteam sollte einen Fokus auf Testautomatisierung legen.

Erklärung: Eine Testautomatisierung ermöglicht es einem Projektteam, schnell und ohne hohen Aufwand festzustellen, ob Neuentwicklungen zu Fehlern im Rest der Software führen.

### E.4.2 Äquivalenzklasse T2

**Durchführung der Tests am Ende eines Sprintes**

T2.1 Die Integrations- und Systemtests sollten in einer Phase am Ende des Sprintes durchgeführt werden. Das Entwicklerteam kann darauf selbstständig die Software deployen.

Erklärung: Dadurch dass die Integration aufwändig und/oder kritisch ist (z.B. Auswirkung von Fehlern) sollten die Integrations- und Systemtests vom gesamten Entwicklungsteam am Ende des jeweiligen Sprintes durchgeführt werden.

T1.2 Das Projektteam sollte einen Fokus auf Testautomatisierung legen.

Erklärung: Eine Testautomatisierung ermöglicht es einem Projektteam schnell und ohne hohen Aufwand festzustellen, ob Neuentwicklungen zu keinen Fehlern im Rest der Software führen.

### E.4.3 Äquivalenzklasse T3

**Durchführung der Tests durch ein extra Testteam**

T3.1 Die Integration- und Systemtests sollten durch ein extra Testteam durchgeführt werden, das parallel zur Entwicklung arbeitet.

Erklärung: Der Projektkontext (Großes und/oder verteiltes Team und/oder Kritikalität der Integration und/oder großer Integrationsaufwand) erfordert, dass die Integration und die Durchführung der Integration- und Systemtests zentral durchgeführt werden.

T1.2 Das Projektteam sollte einen Fokus auf Testautomatisierung legen.

Erklärung: Eine Testautomatisierung ermöglicht es einem Projektteam schnell und ohne hohen Aufwand festzustellen, ob Neuentwicklungen zu Fehlern im Rest der Software führen.

## E.5 Koordination

### E.5.1 Äquivalenzklasse K1

**Kontinuierliche Planung**

K1.1: Der Product Owner oder das Entwicklerteam können den Überblick über das Projekt behalten, die Roadmap verwalten und den Projektstatus überprüfen.

Erklärung: Dadurch dass es maximal ein oder zwei Entwicklerteams gibt, ist kein extra Team oder Rolle für die Projektplanung notwendig.

K1.2: Das Projektteam kann eigenständige Entscheidungen treffen solange, das Team nicht fundamental von der gestellten Aufgabe abweicht oder die Projektparameter (Budget und Zeit) von einer Entscheidung betroffen sind.

Erklärung: Eigenständige Entscheidungen des Projektteams führen zu mehr Kommunikation und Flexibilität.

### E.5.2 Äquivalenzklasse K2

**Kontinuierliche Planung**

K2.1: Die Entwicklerteams sollen selbstständige Entscheidungen und Absprachen treffen, solange nicht das ganze Projekt von einer Entscheidung betroffen ist, das Team fundamental von der Aufgabenstellung abweichen muss oder sich eine Entscheidung auf die Projektparameter (Budget und Zeit) auswirkt.

Erklärung: Da das Projektteam groß und/oder verteilt ist und/oder Abhängigkeiten zu berücksichtigen hat, unterstützt diese Verteilung der Absprachen und Entscheidungen dabei, genug Kontrolle über die Entwicklerteams zu behalten und diese nicht unnötig zu verlangsamen.

K2.2 Das Projektteam sollte eine schnelle informelle Kommunikation zwischen der Projektleitung und den Entwicklungsteams ermöglichen. Planungsänderungen und andere Informationen müssen schnell zu den Entwicklern transportiert werden können. Die Entwickler müssen schnell und einfach Feedback zu der Planung geben und anmerken können, wenn sie auf Probleme treffen.

Erklärung: Es ist wichtig, dass keine Asynchronisation in der Planung zwischen der Projektleitung und den Entwicklerteams entsteht. Andernfalls kann dies zu Koordinationsproblemen führen.

### E.5.3 Kontext: Verteilte Entwicklung

**Kontinuierliche Planung**

Es sollte eine zusätzliche Rolle geschaffen werden, die die Kommunikation zwischen verschiedenen Standorten lenkt.

Erklärung: Zwischen verschiedenen Standorten können Kommunikationsprobleme auftreten. Eine Kommunikationsrolle unterstützt die Koordination von verschiedenen Entwicklungsstandorten.

## E.6 Fragen bei der Eingabe der Kontextfaktoren

1. **Größe des Projektteams**

   Wie groß ist das Projektteam?

   (a) Klein (1 - 2 Entwicklerteam)

   (b) Groß (3 - 9 Entwicklerteams)

   (c) Sehr groß (+ 10 Entwicklerteams)

   Schwellwert bei (b)

   Ähnlich verwendet von [113]

2. **Verteilung des Projektteams**

   Wie verteilt ist das Projektteam?

   (a) Co-located

   (b) Innerhalb eines Landes

   (c) Global

   Schwellwert bei (b)

   Ähnlich verwendet von [122]

3. **Anzahl der Stakeholder**

   Wie viele Stakeholder gibt es?

   (a) 1-5

   (b) 5-10

   (c) 10-15

   (d) ¿15

   Schwellwert bei (c)

   - Ähnlich verwendet von [123]

4. **Verfügbarkeit der Stakeholder**

Wie verfügbar sind die Stakholder?

(a) Über den gesamten Projektverlauf verfügbar

(b) Teilweise über den Projektverlauf verfügbar

(c) Vorwiegend bis nur am Anfang des Projektes verfügbar

Schwellwert bei (b)

5. **Art der Abhängigkeiten**

Welche Abhängigkeiten gibt es in dem Projekt?

(a) Es existieren keine Abhängigkeiten in dem Projekt

(b) Es gibt Abhängigkeiten zwischen unterschiedlichen Entwicklerteams

(c) Es gibt Abhängigkeiten zwischen unterschiedlichen Entwicklerteams und/oder zu anderen Projekten

Schwellwert bei (b)

6. **Anzahl der kritischen nicht-funktionalen Anforderungen**

Wie viele kritische nicht-funktionalen Anforderungen gibt es?

(a) Das Projekt besitzt keine nicht-funktionalen Anforderungen, die direkt am Anfang des Projektes berücksichtigt werden müssen.

(b) Das Projekt besitzt wenige nicht-funktionale Anforderungen, die direkt am Anfang des Projektes berücksichtigt werden müssen.

(c) Das Projekt besitzt viele nicht-funktionale Anforderungen, die direkt am Anfang des Projektes berücksichtigt werden müssen.

Schwellwert bei (b)

7. **Bekanntheit der Anforderungen**

Wie bekannt sind die die Anforderungen?

(a) Sehr bekannt (Einfacher Bereich)

(b) Gut Bekannt (Komplizierter Bereich

(c) Wenig Bekannt (Komplexer Bereich)

(d) Sehr wenig bis gar nicht bekannt (Chaotischer Bereich)

Schwellwert bei (c)

Ähnlich verwendet von [110]

8. **Bekanntheit der Technologie**

   Wie bekannt ist der Umgang mit der Technologie?

   (a) Sehr bekannt (Einfacher Bereich)

   (b) Gut Bekannt (Komplizierter Bereich

   (c) Wenig Bekannt (Komplexer Bereich)

   (d) Sehr wenig bis gar nicht bekannt (Chaotischer Bereich)

   Schwellwert bei (c)

   Ähnlich verwendet von [110]

9. **Nutzung von Anwendungsframeworks**

   In welchem Umfang können bestehende Anwendungsframeworks bei der Entwicklung eingesetzt werden?

   (a) Für alle Bereiche der Anwendung können Anwendungsframeworks eingesetzt werden.

   (b) Für einige Bereiche der Anwendung können Anwendungsframeworks eingesetzt werden.

   (c) Es können keine Anwendungsframeworks eingesetzt werden.

   Schwellwert bei (b)

10. **Bedürfnis nach fixierten Ressourcen**

    Wie groß ist das Bedürfnis des Kunden nach fixierten Ressourcen und Planungssicherheit?

    (a) Der Kunde benötigt keine Schätzung der Dauer und des Budgets für das Projekt.

    (b) Der Kunde benötigt nur eine grobe Schätzung der Dauer und des Budgets für das Projekt.

    (c) Der Kunde möchte eine möglichst genaue Schätzung der Dauer und des Budgets für das Projekt.

    (d) Der Kunde möchte eine Bindung von Dauer, Budget und Umfang.

    Schwellwert bei (c)

11. **Existenz von Deadlines**

    Existieren genaue Liefertermine (Deadlines) für bestimmte Anforderungen?

    (a) Nein

(b) Ja

Schwellwert bei (b)

12. **Verfügbarkeit des Projektteams**

    Arbeiten die Projektteilnehmer zu 100% an dem Projekt?

    (a) Nein

    (b) Ja

    Schwellwert bei (b)

13. **Eingebettete Software**

    Handelt es sich bei der Anwendung um eingebettete Software (Kombination aus Hardware und Software)?

    (a) Nein

    (b) Ja

    Schwellwert bei (b)

14. **Aufwand der Integration**

    Wie komplex ist die kontinuierliche Integration von Softwareteilen aus verschiedenen Entwicklerteams?

    (a) Die Integration findet automatisiert statt

    (b) Die Integration muss manuell durchgeführt werden

    (c) Die Integration verursacht viel Aufwand

    Schwellwert bei (b)

15. **Verlust bei Fehlern**

    Welche Auswirkung haben Fehler in der Software?

    (a) Komfort des Benutzers

    (b) Verlust von wenig Geld

    (c) Verlust von viel Geld

    (d) Verlust von einem Menschen

    (e) Verlust von vielen Menschen

    Schwellwert bei (c)

    Ähnlich verwendet von [36]

16. **Kritikialität der Integration**

    Können einzelne Fehler zu einem Versagen des gesamten Systems führen?

    (a) Nein

    (b) Ja

    Schwellwert bei (b)

17. **Qualitätsstandards**

    Gibt es für die Entwicklung Qualitätsstandards, die eingehalten werden müssen?

    (a) Nein

    (b) Ja

    Schwellwert bei (b)

# F

# Daten der Evaluation

## F.1 Testszenarien

**Szenario 1**

- Das Projektteam besteht aus fünf Entwicklungsteams und ist innerhalb eines Landes verteilt. Die Projektteilnehmer arbeiten zu 100% an dem Projekt
- Es gibt 15 Stakeholder, die über die gesamte Laufzeit des Projektes verfügbar sind
- Das Projekt besitzt Abhängigkeiten zu anderen Projekten und wenige nicht-funktionale Anforderungen, die direkt zu Beginn umgesetzt werden müssen
- Die Anforderungen und die Technologie sind jeweils ausreichend bekannt
- Die Entwicklung kann teilweise durch den Einsatz von vorgefertigten Anwendungsframeworks unterstützt werden
- Es gibt Deadlines für Anforderungen und der Kunde möchte eine möglichst genaue Schätzung darüber, wie viel das gesamte Projekt kostet und wie lange es dauert
- Die Integration von einzelnen Features muss manuell durchgeführt werden und Fehler führen nicht zu einem Versagen des Gesamtsystems. Fehler führen zu einem geringen

finanziellen Verlust.

- Das Projekt enthält keine Hardwareentwicklung
- Es sind keine besonderen Qualitätsstandards einzuhalten

**Szenario 2**

- Das Projektteam besteht aus 2 Entwicklungsteams und ist co-located. Die Projektteilnehmer arbeiten zu 100% an dem Projekt
- Es gibt 3 Stakeholder, die über die gesamte Laufzeit des Projektes verfügbar sind
- Das Projekt besitzt Abhängigkeiten zu anderen Projekten und keine nicht-funktionalen Anforderungen, die direkt zu Beginn umgesetzt werden müssen
- Die Anforderungen und die Technologie sind jeweils wenig bekannt
- Die Entwicklung kann nicht durch den Einsatz von vorgefertigten Anwendungsframeworks unterstützt werden
- Es gibt keine Deadlines für einzelne Anforderungen und der Kunde möchte nur grob wissen, wie viel das gesamte Projekt kostet und wie lange es dauert
- Die Integration von einzelnen Features wird automatisch durchgeführt und Fehler führen nicht zu einem Versagen des gesamten Systems. Fehler führen nur zu einer Unzufriedenheit bei Nutzern.
- Das Projekt enthält keine Hardwareentwicklung
- Es sind keine besonderen Qualitätsstandards einzuhalten

**Szenario 3**

- Das Projektteam besteht aus 7 Entwicklungsteams und arbeitet verteilt innerhalb eines Landes. Die Projektteilnehmer arbeiten nicht zu 100% an dem Projekt
- Es gibt 4 Stakeholder, die vorwiegend zu Beginn des Projektes verfügbar sind
- Das Projekt besitzt Abhängigkeiten zu anderen Projekten und viele nicht-funktionale Anforderungen, die direkt zu Beginn umgesetzt werden müssen
- Die Anforderungen und die Technologie sind jeweils sehr wenig bekannt

- Die Entwicklung kann nicht durch den Einsatz von vorgefertigten Anwendungsframeworks unterstützt werden
- Es gibt Deadlines für einzelne Anforderungen und der Kunde möchte eine Bindung von Zeit, Budget und Dauer
- Die Integration von einzelnen Features ist mit viel Aufwand verbunden. Einzelne Fehler führen zu einem hohen finanziellen Verlust.
- Das Projekt enthält keine Hardwareentwicklung
- Es sind Qualitätsstandards einzuhalten

## F.2 Ergänzungen und Ablehnungen der Experten

**Tabelle F.1:** Ergänzungen und Ablehnungen der Experten in Szenario 1

| | Experte 1 | Experte 2 | Experte 3 | Experte 4 | APH Framework |
|---|---|---|---|---|---|
| Kick-off Meeting bei verteilten Teams | X | | | | |
| Architekturdefinition zu Beginn des Projektes ermöglicht eine bessere Schätzung | X | | | | |
| Architekturdefinition zu Beginn des Projektes senkt den Projektaufwand | | X | X | | |
| Teamzusammensetzung beeinflusst Schätzung | | X | | | |
| Aufwandsschätzung sollte durch die Entwickler durchgeführt werden | X | X | | | |
| Nutzung einer Chief Product Owner Rolle | X | | X | | |
| Eigenständige Architekturentscheidungen der Entwicklerteams + Einschränkung | X | | | | |
| Aufnahme von Feedbackschleifen bei den Tätigkeiten während der kontinuierlichen Anforderungsanalyse | | X | X | | |
| Rolle für Standortkommunikation während der Anforderungsanalyse integrieren | | | X | | |
| Statusüberprüfung in jedem Sprint Review | | | X | | |
| Aufnahme der Kommunikation mit Stakeholdern bei den Tätigkeiten während der kontinuierlichen Anforderungsanalyse | | | X | | |
| Fokus auf das Was während der Anforderungsanalyse | | | | X | |
| A3.1 | | | | X | X |

**Tabelle F.2:** Ergänzungen und Ablehnungen der Experten in Szenario 2

| | Experte 1 | Experte 2 | Experte 3 | Experte 4 | APH Framework |
|---|---|---|---|---|---|
| Einsatz von Business Experten während der Anforderungsanalyse | X | | | | |
| Kommunikation mit dem Kunden über dessen Erwartungshaltung | | X | | | |
| Feedbackschleifen mit Endnutzern | X | X | | | |
| Einsatz von Personas | | | X | | |
| Anpassen des Projektes in Abhängigkeit des Projektstatus | | | | X | |
| Nutzung der Prozessaktivität für die kontinuierliche Architekturdefinition | | X | X | X | X |

**Tabelle F.3:** Ergänzungen und Ablehnungen der Experten in Szenario 3

| | Experte 1 | Experte 2 | Experte 3 | Experte 4 | APH Framework |
|---|---|---|---|---|---|
| Absprache mit Kunden über Implementierung von Feedbackschleifen | X | X | X | | |
| Gemeinsames Bedürfnis nach Feedbackschleifen bilden | | X | | | |
| Puffer in Iterationen integrieren | | X | | | |
| Pauschale bei der kontinuierlichen Schätzung verwenden | X | | | | |
| Festlegung der Entwickler für eine Iteration | | | X | | |
| Jeweilige Verfügbarkeit von relevanten Stakeholdern | | | X | | |
| Verwendung von Risikomanagement | | | X | | |
| Verminderte Performance bei wechselnden Entwicklern | | | X | | |
| Absprache über benötigte Qualität | | | | X | |

# Literaturverzeichnis


[1] J. Klünder, R. Hebig, P. Tell, M. Kuhrmann, J. Nakatumba-Nabende, R. Heldal, S. Krusche, M. Fazal-Baqaie, M. Felderer, M. F. G. Bocco, S. Küpper, S. A. Licorish, G. Lopez, F. McCaffery, O. O. 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 *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP '19. IEEE Press, 2019, p. 255–264.

[2] P. Tell, J. Klünder, S. Küpper, D. Raffo, S. MacDonell, J. Münch, D. Pfahl, O. Linssen, and M. Kuhrmann, "Towards the statistical construction of hybrid development methods," *Journal of Software: Evolution and Process*, vol. 33, no. 1, p. e2315, 2021.

[3] N. Prenner, C. Unger-Windeler, and K. Schneider, "Goals and challenges in hybrid software development approaches," *Journal of Software: Evolution and Process*, vol. 33, no. 11, p. e2382, 2021.

[4] N. Prenner, J. Klünder, and K. Schneider, "Defining Frames to Structure Agile Development in Hybrid Settings - A Multi-Case Interview Study," ser. ICSSP'22. New York, NY, USA: Association for Computing Machinery, 2022, p. 34–44.

[5] B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*. USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[6] J. Noll and S. Beecham, "How Agile Is Hybrid Agile? An Analysis of the HELENA Data," ser. PROFES'19. Springer International Publishing, 2019, pp. 341–349.

[7] 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*, ser. ICSSP 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 30–39.

[8] N. Prenner, "Towards Improving the Organization of Hybrid Development Approaches," in *Proceedings of the International Conference on Software and System Processes*, ser. ICSSP '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 185–188.

[9] O. Armbrust and H. D. Rombach, "The right process for each context: objective evidence needed," in *International Conference on Software and Systems Process, ICSSP 2011, Honolulu, HI, USA, May 21-22, 2011, Proceedings*. ACM, 2011, pp. 237–241.

[10] G. Kalus and M. Kuhrmann, "Criteria for Software Process Tailoring: A Systematic Review," in *Proceedings of the 2013 International Conference on Software and System Process*, ser. ICSSP 2013. New York, NY, USA: Association for Computing Machinery, 2013, p. 171–180.

[11] P. Clarke and R. V. O'Connor, "The Situational Factors That Affect the Software Development Process: Towards a Comprehensive Reference Framework," *Information and Software Technology*, vol. 54, no. 5, p. 433–447, may 2012.

[12] S. Küpper, A. Rausch, and U. Andelfinger, "Towards the Systematic Development of Hybrid Software Development Processes," in *Proceedings of the 2018 International Conference on Software and System Process*, ser. ICSSP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 157–161.

[13] J. Klünder, D. Karajic, P. Tell, O. Karras, C. Münkel, J. Münch, S. G. MacDonell, R. Hebig, and M. Kuhrmann, "Determining Context Factors for Hybrid Development Methods with Trained Models," in *Proceedings of the International Conference on Software and System Processes*, ser. ICSSP '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 61–70.

[14] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Upper Saddle River, New Jersey: Prentice Hall, 2002.

[15] W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," in *Proceedings of the 9th International Conference on Software Engineering*, ser. ICSE '87. Los Alamitos, CA, USA: IEEE Computer Society Press, 1987, pp. 328–338.

[16] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change (2nd Edition)*. Boston: Addison-Wesley Professional, 2004.

[17] V. R. Basili and H. D. Rombach, "Tailoring the Software Process to Project Goals and Environments," in *Proceedings of the 9th International Conference on Software Engineering*, ser. ICSE '87. Washington, DC, USA: IEEE Computer Society Press, 1987, p. 345–357.

[18] V. Basili and H. Rombach, "The TAME project: towards improvement-oriented software environments," *IEEE Transactions on Software Engineering*, vol. 14, no. 6, pp. 758–773, 1988.

[19] M. Kuhrmann, P. Diebold, and J. Münch, "Software Process Improvement: A systematic mapping study on the state of the art," *PeerJ Computer Science*, vol. 2, pp. 1–38, 05 2016.

[20] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[21] A. R. Hevner, "A Three Cycle View of Design Science Research," *Scandinavian Journal of Information Systems*, vol. 19, p. 4, 2007.

[22] N. Keshta and Y. Morgan, "Comparison between traditional plan-based and agile software processes according to team size project domain (A systematic literature review)," *8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2017.

[23] B. Boehm and R. Turner, "Using Risk to Balance Agile and Plan-Driven Methods," *Computer*, vol. 36, no. 6, pp. 57–66, Jun. 2003.

[24] T. Thesing, C. Feldmann, and M. Burchardt, "Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project," *Procedia Computer Science*, vol. 181, pp. 746–756, 2021.

[25] P. Hohl, J. Klünder, A. Bennekum, R. Lockard, J. Gifford, J. Münch, M. Stupperich, and K. Schneider, "Back to the future: origins and directions of the "Agile Manifesto" – views of the originators," *Journal of Software Engineering Research and Development*, vol. 6, 12 2018.

[26] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor,

K. Schwaber, J. Sutherland, and D. Thomas, “Manifesto for Agile Software Development,” zuletzt besucht: 07/2022. [Online]. Available: http://www.agilemanifesto.org/

[27] R. Knaster and D. Leffingwell, *SAFe 5. 0 Distilled: Achieving Business Agility with the Scaled Agile Framework*. ADDISON WESLEY Publishing Company Incorporated, 2020.

[28] L. T. Heeager, “The Agile and the Disciplined Software Approaches: Combinable or Just Compatible?” *Information systems development*, pp. 35–49, 2013.

[29] D. West, M. Gilpin, T. Grant, and A. Anderson, “Water-scrum-fall is the reality of agile for most organizations today,” *Forrester Research*, vol. 26, pp. 1–17, 2011.

[30] G. Theocharis, M. Kuhrmann, J. Münch, and P. Diebold, “Is water-scrum-fall reality? on the use of agile and traditional development practices,” in *International Conference on Product-Focused Software Process Improvement*, ser. PROFES’15. Springer, 2015, pp. 149–166.

[31] J. Klünder, M. Busch, N. Dehn, and O. Karras, “Towards Shaping the Software Lifecycle with Methods and Practices,” *2021 IEEE/ACM Joint 15th International Conference on Software and System Processes (ICSSP) and 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE)*, pp. 1–11, 2021.

[32] 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*, vol. 44, no. 10, pp. 932–950, 2018.

[33] Lan Cao, K. Mohan, Peng Xu, and B. Ramesh, “How extreme does extreme programming have to be? Adapting XP practices to large-scale projects,” *37th Annual Hawaii International Conference on System Sciences*, pp. 10 pp.–, 2004.

[34] L. T. Heeager and P. A. Nielsen, “Meshing agile and plan-driven development in safety-critical software: a case study,” *Empirical Software Engineering*, vol. 25, no. 2, pp. 1035–1062, 2020.

[35] S. Bick, A. Scheerer, and K. Spohrer, “Inter-Team Coordination in Large Agile Software Development Settings: Five Ways of Practicing Agile at Scale,” in *Proceedings of the Scientific Workshop Proceedings of XP2016*, ser. XP ’16 Workshops. New York, NY, USA: Association for Computing Machinery, 2016.

[36] B. Boehm and R. Turner, “Management challenges to implementing agile processes in traditional development organizations,” *IEEE Software*, vol. 22, no. 5, pp. 30–39, 2005.

[37] S. Theobald and P. Diebold, “Interface Problems of Agile in a Non-agile Environment,” *Agile Processes in Software Engineering and Extreme Programming*, pp. 123–130, 2018.

[38] R. Hoda, P. Kruchten, J. Noble, and S. Marshall, “Agility in context,” in *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*, 2010, pp. 74–88.

[39] M. Waterman, J. Noble, and G. Allan, “How Much Up-Front? A Grounded theory of Agile Architecture,” *IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, pp. 347–357, 2015.

[40] L. Taxén and U. Pettersson, “Agile and Incremental Development of Large Systems,” in *7th European Systems Engineering Conference (EuSEC 2010), Stockholm, Sweden, May 23–26, 2010*, 2010.

[41] V. T. Heikkilä, M. Paasivaara, C. Lassenius, and C. Engblom, “Continuous Release Planning in a Large-Scale Scrum Development Organization at Ericsson,” in *Agile Processes in Software Engineering and Extreme Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 195–209.

[42] J. Cho, “A hybrid software development method for large-scale projects: rational unified process with scrum,” *Issues in Information Systems*, vol. 10, no. 2, pp. 340–348, 2009.

[43] A. A. Abdelaziz, Y. El-Tahir, and R. Osman, “Adaptive Software Development for developing safety critical software,” *International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*, pp. 41–46, 2015.

[44] M. McHugh, F. McCaffery, and G. Coady, “An Agile Implementation within a Medical Device Software Organisation,” in *Software Process Improvement and Capability Determination*. Cham: Springer International Publishing, 2014, pp. 190–201.

[45] A. Geras, M. Smith, and J. Miller, “Configuring hybrid agile-traditional software processes,” in *International Conference on Extreme Programming and Agile Processes in Software Engineering*. Springer, 2006, pp. 104–113.

[46] L. Silvestre, M. C. Bastarrica, J. A. Hurtado, and J. Marin, “Formalizing the Goal-directed and Context-based Software Process Tailoring Method,” in *2021 XLVII Latin American Computing Conference (CLEI)*, 2021, pp. 1–9.

[47] P. Xu and B. Ramesh, “Using Process Tailoring to Manage Software Development Challenges,” *IT Professional*, vol. 10, no. 4, pp. 39–45, 2008.

[48] 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*, ser. ICSSP ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 145–154.

[49] K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering: An update,” *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[50] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic Mapping Studies in Software Engineering,” in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE ’08, Swindon, GBR, 2008, p. 68–77.

[51] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[52] M. S. Mirza and S. Datta, “Strengths and Weakness of Traditional and Agile Processes - A Systematic Review,” *Journal Software*, vol. 14, pp. 209–219, 2019.

[53] M. Špundak, “Mixed Agile/Traditional Project Management Methodology – Reality or Illusion?” *Procedia - Social and Behavioral Sciences*, vol. 119, pp. 939 – 948, 2014, selected papers from the 27th IPMA (International Project Management Association), World Congress, Dubrovnik, Croatia, 2013.

[54] D. Karlström and P. Runeson, “Integrating agile software development into stage-gate managed product development,” *Empirical Software Engineering*, vol. 11, no. 2, pp. 203–225, 2006.

[55] H.-F. Chang and S. Lu, “Toward the Integration of Traditional and Agile Approaches,” *International Journal of Advanced Computer Science and Applications*, vol. 4, 03 2013.

[56] R. Hoda and J. Noble, “Becoming Agile: A Grounded Theory of Agile Transitions in Practice,” in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, ser. ICSE’17. Los Alamitos, CA, USA: IEEE Computer Society, may 2017, pp. 141–151.

[57] V. Braun and V. Clarke, “Using thematic analysis in psychology,” *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[58] J. Maxwell, “Understanding and Validity in Qualitative Research,” *Harvard Educational Review*, vol. 62, pp. 279–300, 01 1992.

[59] K. M. Zaki and R. Moawad, “A hybrid disciplined Agile software process model,” in *2010 The 7th International Conference on Informatics and Systems (INFOS)*, March 2010, pp. 1–8.

[60] W. D. Millard, D. M. Johnson, J. M. Henderson, N. Lombardo, R. B. Bass, and J. Smith, “Embedding Agile Practices within a Plan-Driven Hierarchical Project Life Cycle,” in *INCOSE International Symposium*, vol. 24, no. 1. Wiley Online Library, 2014, pp. 745–758.

[61] I. P. D. Lesmana, R. N. Karimah, and B. Widiawan, “Agile-Waterfall hybrid for prevention information system of dengue viral infections: A case study in Health Department of Jember, East Java, Indonesia,” in *2016 14th International Conference on ICT and Knowledge Engineering (ICT KE)*, Nov 2016, pp. 1–6.

[62] S. Sultana, Y. H. Motla, S. Asghar, M. Jamal, and R. Azad, “A hybrid model by integrating agile practices for Pakistani software industry,” in *2014 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, Feb 2014, pp. 256–262.

[63] G. Lozo and S. Jovanović, “A flexible hybrid method for IT project management,” *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 7, pp. 1027–1036, 2012.

[64] K. Smojver, H. Belani, and Z. Car, “Building a hybrid process model for a complex software system integration,” in *2009 10th International Conference on Telecommunications*, June 2009, pp. 147–153.

[65] L. Bose and S. Thakur, “Introducing Agile into a Non Agile Project: Analysis of Agile Methodology with its Issues and Challenges.” *International Journal of Advanced Research in Computer Science*, vol. 4, no. 1, 2013.

[66] L. T. Heeager and P. A. Nielsen, “Agile software development and its compatibility with a document-driven approach? A case study,” in *20th Australasian Conference on Information Systems Compatibility of Agile and Document-Driven Approaches, Melbourne*, 2009.

[67] R. F. Paige, R. Charalambous, X. Ge, and P. J. Brooke, “Towards Agile Engineering of High-Integrity Systems,” in *Computer Safety, Reliability, and Security*, M. D. Harrison and M.-A. Sujan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 0–43.

[68] A. Read and R. O. Briggs, “The Many Lives of an Agile Story: Design Processes, Design Products, and Understandings in a Large-Scale Agile Development Project,” in *2012 45th Hawaii International Conference on System Sciences*, Jan 2012, pp. 5319–5328.

[69] V. T. Heikkilä, M. Paasivaara, C. Lasssenius, D. Damian, and C. Engblom, “Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson,” *Empirical Software Engineering*, vol. 22, no. 6, pp. 2892–2936, 2017.

[70] L. T. Portela and G. Borrego, “Scrumconix: Agile and Documented Method to AGSD,” in *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, Aug 2016, pp. 195–196.

[71] A. Shimoda and K. Yaguchi, “A Method of Setting the Order of User Story Development of an Agile-Waterfall Hybrid Method by Focusing on Common Objects,” in *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, July 2017, pp. 301–306.

[72] N. Sharma and M. Wadhwa, “eXSRUP: Hybrid Software Development Model Integrating Extreme Programing, Scrum & Rational Unified Process,” *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 16, no. 2, pp. 377–388, 2015.

[73] S. U. Nisa and M. R. J. Qureshi, “Empirical estimation of hybrid model: A controlled case study,” *IJ Information Technology and Computer Science*, vol. 4, no. 8, pp. 43–50, 2012.

[74] G. Ahmad, T. R. Soomro, and M. N. Brohi, “XSR: Novel Hybrid Software Development Model (Integrating XP, Scrum & RUP),” *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 3, pp. 126–130, 2014.

[75] G. Rong, D. Shao, and H. Zhang, “SCRUM-PSP: Embracing Process Agility and Discipline,” in *2010 Asia Pacific Software Engineering Conference*, Nov 2010, pp. 316–325.

[76] M. Waterman, J. Noble, and G. Allan, “The effect of complexity and value on architecture planning in agile software development,” *International Conference on Agile Software Development*, pp. 238–252, 2013.

[77] O. Adelakun, R. Garcia, T. Tabaka, and R. Ismail, “Hybrid Project Management: Agile with Discipline,” ser. International Conference on Information Resources Management (CONF-IRM). Association For Information Systems, 2017.

[78] P. C. Anitha, D. Savio, and V. S. Mani, “Managing requirements volatility while “Scrumming” within the V-Model,” *3rd International Workshop on Empirical Requirements Engineering (EmpiRE)*, pp. 17–23, 2013.

[79] X. Ge, R. F. Paige, and J. A. McDermid, “An Iterative Approach for Development of Safety-Critical Software and Safety Arguments,” *Agile Conference*, pp. 35–43, 2010.

[80] L. Tordrup and P. Nielsen, “Agile software development and its compatibility with a document-driven approach? A case study,” *ACIS Proceedings*, pp. 205–214, 01 2009.

[81] L. Tordrup, “Introducing Agile Practices in a Documentation-Driven Software Development Practice: A Case Study,” *Journal of Information Technology Case and Application Research*, vol. 14, pp. 3–24, 07 2014.

[82] D. Batra, W. Xia, D. Meer, and K. Dutta, “Balancing agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry,” *Communications of the Association for Information Systems*, vol. 27, pp. 379–394, 01 2010.

[83] J. Binder, L. I. Aillaud, and L. Schilli, “The Project Management Cocktail Model: An Approach for Balancing Agile and ISO 21500,” *Procedia - Social and Behavioral Sciences*, vol. 119, pp. 182 – 191, 2014, selected papers from the 27th IPMA (International Project Management Association), World Congress, Dubrovnik, Croatia, 2013.

[84] 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*, vol. 07, pp. 39–46, 03 2017.

[85] M. R. Qureshi, “Agile software development methodology for medium and large projects,” *Software, IET*, vol. 6, pp. 358–363, 08 2012.

[86] G. van Waardenburg and H. van Vliet, “When agile meets the enterprise,” *Information and Software Technology*, vol. 55, no. 12, pp. 2154 – 2171, 2013.

[87] T. Hayata and J. Han, “A hybrid model for IT project with Scrum,” in *Proceedings of 2011 IEEE International Conference on Service Operations, Logistics and Informatics*, 2011, pp. 285–290.

[88] I. Laux and J. Kranz, “Coexisting Plan-driven and Agile Methods: How Tensions Emerge and Are Resolved,” *International Conference on Information Systems (ICIS)*, 12 2019.

[89] B. Ramesh, K. Mohan, and L. Cao, “Ambidexterity in Agile Distributed Development: An Empirical Investigation,” *Information systems research*, pp. 323–339, Jun. 2012.

[90] G. Islam and T. Storer, “A case study of agile software development for safety-critical systems projects,” *Reliability Engineering and System Safety*, vol. 200, p. 106954, 2020.

[91] T. Dingsøyr, N. B. Moe, T. E. Fægri, and E. A. Seim, “Exploring Software Development at the Very Large-Scale: A Revelatory Case Study and Research Agenda for Agile Method Adaptation,” *Empirical Software Engineering*, vol. 23, no. 1, p. 490–520, Feb. 2018.

[92] P. Xu, “Coordination In Large Agile Projects,” *Review of Business Information Systems (RBIS)*, vol. 13, 05 2011.

[93] J. Pechau, “Rafting the Agile Waterfall: Value Based Conflicts of Agile Software Development,” *Proceedings of the 16th European Conference on Pattern Languages of Programs*, pp. 1–15, 2011.

[94] P. Matkovic, M. Maric, P. Tumbas, and M. Sakal, “Traditionalisation of agile processes: Architectural aspects,” *Computer Science and Information Systems*, vol. 15, pp. 79–109, 2018.

[95] O. Uludağ, M. Kleehaus, X. Xu, and F. Mat, “Investigating the Role of Architects in Scaling Agile Frameworks,” *IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC)*, pp. 123–132, 2017.

[96] J. Heidenberg, M. Matinlassi, M. Pikkarainen, P. Hirkman, and J. Partanen, “Systematic Piloting of Agile Methods in the Large: Two Cases in Embedded Systems Development,” *Product-Focused Software Process Improvement*, pp. 47–61, 2010.

[97] A. Scheerer, T. Schimmer, and T. Kude, “Coordination in Large-Scale Agile Software Development: A Multiteam Systems Perspective,” *47th Hawaii International Conference on System Sciences (HICSS)*, pp. 4780–4788, 01 2014.

[98] 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,” *IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pp. 315–322, 2017.

[99] A. Tjørnehøj, “The role of the IT-Project Manager in Organizations that Balance Agile and Traditional Software Development,” *IRIS: Selected Papers of the Information Systems Research Seminar in Scandinavia*, no. 9, 2018.

[100] P. A. Rottier and V. Rodrigues, “Agile Development in a Medical Device Company,” *Agile Conference*, pp. 218–223, 2008.

[101] B. Glaser, *Emergence Vs Forcing: Basics of Grounded heory Analysis*, ser. Emergence vs. forcing. Sociology Press, 1992.

[102] B. G. Glaser and A. L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. New York, NY: Aldine de Gruyter, 1967.

[103] S. E. Hove and B. Anda, “Experiences from conducting semi-structured interviews in empirical software engineering research,” in *11th IEEE International Software Metrics Symposium (METRICS’05)*, 2005, pp. 10 pp.–23.

[104] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical Software Engineering*, vol. 14, pp. 131–164, 2008.

[105] “IEEE Standard for Software Life Cycle Processes - Risk Management,” *IEEE Std 1540-2001*, pp. 1–24, 2001.

[106] B. Boehm, “Get ready for agile methods, with care,” *Computer*, vol. 35, no. 1, pp. 64–69, 2002.

[107] O. Boruszewski, “Unterstuetzung der Koexistenz von agilen und traditionellen Anforderungsartefakten,” Ph.D. dissertation, Bonn, 2017.

[108] B. Boehm, D. Port, and A. W. Brown, “Balancing Plan-Driven and Agile Methods in Software Engineering Project Courses,” *Computer Science Education*, vol. 12, no. 3, pp. 187–195, 2002.

[109] C. Rupp, *Requirements-Engineering und -Management*, 6th ed. München: Hanser, 2014.

[110] R. D. Stacey, *Strategic management and organisational dynamics: The challenge of complexity to ways of thinking about organisations*. Pearson education, 2007.

[111] O. Sohaib and K. Khan, “Integrating usability engineering and agile software development: A literature review,” in *2010 International Conference On Computer Design and Applications*, vol. 2, 2010, pp. V2–32–V2–38.

[112] O. Stender, *Entwicklung eines Editors für Informationsflüsse für die Konstruktion von hybriden Entwicklungsansätzen*. Leibniz Universität Hannover, Fachgebiet Software Engineering, 2022.

[113] T. Dingsøyr, T. E. Fægri, and J. Itkonen, “What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development,” in *Product-Focused Software Process Improvement*. Cham: Springer International Publishing, 2014, pp. 273–276.

[114] D. Morbach, *Entwicklung eines kontextgestützten Editors für die Erstellung hybrider Prozesse*. Leibniz Universität Hannover, Fachgebiet Software Engineering, 2021.

[115] K. Peffers, M. Rothenberger, T. Tuunanen, and R. Vaezi, “Design Science Research Evaluation,” in *Design Science Research in Information Systems. Advances in Theory and Practice*, K. Peffers, M. Rothenberger, and B. Kuechler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 398–410.

[116] R. M. O’Keefe and D. E. O’Leary, “Expert system verification and validation: a survey and tutorial,” *Artificial Intelligence Review*, vol. 7, no. 1, pp. 3–42, 1993.

[117] P. D. Grogono, A. D. Preece, R. Shinghal, and C. Y. Suen, “A review of expert systems evaluation techniques,” in *Workshop on Validation and Verification of Knowledge-Based Systems*, 1993, pp. 120–125.

[118] J. Klünder, P. Hohl, and K. Schneider, “Becoming Agile While Preserving Software Product Lines: An Agile Transformation Model for Large Companies,” in *Proceedings of the 2018 International Conference on Software and System Process*, ser. ICSSP ’18. New York, NY, USA: ACM, 2018, pp. 1–10.

[119] R. Y. Takahira, L. R. Laraia, F. A. Dias, A. S. Yu, P. T. S. Nascimento, and A. S. Camargo, “Scrum and Embedded Software development for the automotive industry,” in *Proceedings of PICMET ’14 Conference: Portland International Center for Management of Engineering and Technology; Infrastructure and Service Integration*, July 2014, pp. 2664–2672.

[120] M. Tanveer, “Agile for large scale projects - A hybrid approach,” in *2015 National Software Engineering Conference (NSEC)*, Dec 2015, pp. 14–18.

[121] M. McHugh, F. McCaffery, and G. Coady, “Adopting Agile Practices when Developing Medical Device Software,” *J Comput Eng Inf Technol 4*, vol. 9307, p. 2, 2015.

[122] M. Kuhrmann, P. Tell, J. Klünder, R. Hebig, S. Licorish, and S. MacDonell, “HELENA Stage 2 Results,” 11 2018.

[123] C. Unger-Windeler, “Supporting the tailoring of the product owner role to hybrid development environments,” Ph.D. dissertation, Hannover: Institutionelles Repositorium der Leibniz Universität Hannover, 2020.

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Lebenslauf

| **Persönliche Daten** | |
|---|---|
| Name: | Nils Prenner |
| Geburtsdatum: | 28.04.1992 |
| Geburtsort: | Hannover |

| **Beruflicher Werdegang** | |
|---|---|
| Seit 01/2023: | **IT Berater** |
| | PPI AG |
| 11/2017 - 08/2022: | **Wissenschaftlicher Mitarbeiter** |
| | Fachgebiet Software Engineering |
| | Leibniz Universität Hannover |
| 12/2015 - 08/2017: | **Übungstutor** |
| | Fachgebiet Software Engineering |
| | Leibniz Universität Hannover |

| **Akademischer Werdegang** | |
|---|---|
| 11/2017 - 05/2023: | **Promotionsstudium der Informatik** |
| | Leibniz Universität Hannover |
| 04/2015 - 10/2017: | **Masterstudium der Informatik** |
| | Leibniz Universität Hannover |
| | Abschluss: Master of Science |
| 10/2011 - 04/2015: | **Bachelorstudium der Informatik** |
| | Leibniz Universität Hannover |
| | Abschluss: Bachelor of Science |
| 1998 - 06/2011: | **Grundschule und Gymnasium in Langenhagen** |
| | Abschluss: Abitur |