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

Unterstützung der Balance in hybriden Entwicklungsansätzen mit geeigneten Methoden und Praktiken

Supporting the balance in hybrid development approaches with appropriate methods and practices

Masterarbeit

im Studiengang Informatik

von

Fabian Natusch

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

Hannover, 26.04.2023

Erklärung der Selbstständigkeit

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

Hannover, den 26.04.2023

Fabian Natusch

Zusammenfassung

In der hybriden Entwicklung muss immer entschieden werden, wie ein Projekt mit Methoden und Praktiken gefüllt wird. Jede Methode und Praktik hat Vor- und Nachteile und von daher auch einen Einfluss auf den Erfolg bzw. den Misserfolg in einem Projekt. Für die Auswahl von Methoden und Praktiken muss daher sorgfältig eine Entscheidung getroffen werden.
Um den genannten Aufwand zu verringern und die Auswahl der Methoden und Praktiken zu optimieren, setzt sich diese Arbeit damit auseinander, wie Methoden und Praktiken für hybride Entwicklungsansätze automatisch anhand von geeigneten Kriterien konstruiert werden können.
Für die Konstruktion des Prototyps wird sich zunächst einmal mit der Organisation von hybriden Entwicklungsansätzen auseinandergesetzt. Mithilfe dieser Struktur und den einhergehenden Methoden lassen sich im Konzept durch die sogenannte HELENA-Studie weitere Praktiken festlegen.
Der erstellte Prototyp liefert dabei Ergebnisse, die für nachfolgende Untersuchungen von Relevanz sind. Es wird festgestellt, dass noch einige weitere Forschungsfragen zu klären sind, bevor der Prototyp mit einer weiteren Version Praxisbezug finden kann. Denn die Entscheidungsfindung, welche Methoden und Praktik verwendet werden sollten, hängt von zu vielen Faktoren ab. Es gibt tiefgreifende Erkenntnisse, die mithilfe von Experteninterviews in einer Studie festgestellt werden konnten.
Die Tendenz liegt darin, dass hybride Entwicklungsansätze mit ähnlichen Methoden und Praktiken konstruiert werden. Es zeigt auf, dass die Auswahl des Prototyps und der Studienteilnehmer einige Ähnlichkeiten aufweisen. Das hat zur Folge, dass die Methoden und Praktiken von jedem hybriden Entwicklungsprojekt gleich aufgebaut werden können.

Abstract

In hybrid development, it is always necessary to decide how to fill a project with methods and practices. Each method and practice has advantages and disadvantages and therefore also an influence on the success or failure in a project. Therefore, a careful decision must be made for the selection of methods and practices.
In order to reduce the aforementioned effort and to optimize the selection of methods and practices, this thesis deals with how methods and practices for hybrid development approaches can be constructed automatically based on suitable criteria.
To construct the prototype, we first address the organization of hybrid development approaches. With the help of this structure and the accompanying methods, further practices can be defined in the concept through the so-called HELENA study.
The prototype created provides results that are relevant for subsequent studies. It is noted that there are still some further research questions to be answered before the prototype can find practical relevance with a further version. This is because decision making on which methods and practice to use depends on too many factors. There are profound findings that could be established with the help of expert interviews in a study.
The tendency is that hybrid development approaches are constructed with similar methods and practices. It indicates that the selection of the prototype and the study participants have some similarities. As a result, the methods and practices can be constructed the same by any hybrid development project.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Problemstellung und Ziel der Arbeit . . . 1
- 1.2 Lösungsansatz . . . 2
- 1.3 Ergebnisse der Arbeit . . . 3
- 1.4 Struktur der Arbeit . . . 3

2 Grundlagen **5**
- 2.1 Organisation für hybride Entwicklung . . . 5
  - 2.1.1 Wasserfall-Agile-Ansatz . . . 5
  - 2.1.2 Wasserfall-Iterative-Ansatz . . . 8
  - 2.1.3 Pipeline-basierter-Ansatz . . . 9
- 2.2 Ziele und Herausforderungen für hybride Entwicklung . . . 10
- 2.3 Frames von hybrider Entwicklung . . . 12
- 2.4 Kombination von Methoden und Praktiken . . . 15
- 2.5 Agilität von Methoden, Praktiken und Projektdisziplinen . . . 18

3 Verwandte Arbeiten **23**

4 Konzept **27**
- 4.1 Grundlagen des Konzeptes . . . 27
- 4.2 Auswahl der Methoden . . . 28
- 4.3 Auswahl von Praktiken . . . 30
- 4.4 Grenzen . . . 32
- 4.5 Kriterien von hybrider Entwicklung . . . 35
- 4.6 Zusammensetzung der Teilkonzepte . . . 39

5 Implementierung **41**
- 5.1 Umsetzung . . . 41
- 5.2 Grafische Benutzeroberfläche . . . 42

6 Studiendesign **45**
- 6.1 Ziel . . . 45
- 6.2 Vorbereitung . . . 46
- 6.3 Auswahl und Akquise der Teilnehmer . . . 49

6.4 Ablauf 49
6.5 Studienergebnisse 50
6.6 Auswertung der Studienfragen 53

7 Diskussion **57**
7.1 Diskussion der Studienergebnisse 57
7.2 Beantwortung der Forschungsfrage 58
7.3 Diskussion des Prototyps 60
7.4 Validität 60

8 Zusammenfassung und Ausblick **63**
8.1 Zusammenfassung 63
8.2 Ausblick 64

A Appendix **67**
A.1 Kombinationen in der HELENA-Studie 68
A.2 Agilitätsgrad 70
A.3 Studienergebnisse 72
A.4 Prototyp Installation 72
A.5 Prototyp Benutzeroberfläche 73
A.6 Studie 74
A.7 CD 77

Kapitel 1

Einleitung

In dem Jahr 2001 fand ein zweitägiges Meeting mit 17 Software-Professionals (überwiegend Praktiker) in Snowbird statt [11]. Hierbei wurden die Grundsteine für das agile Manifest gelegt. Denn die vier Werte und zwölf Prinzipien bilden die Grundlage für agile Entwicklung. Viele Unternehmen mischen die traditionelle Entwicklung mit der agilen Entwicklung und arbeiten hybrid.
Die Gründe für die hybride Entwicklung liegen auf der Hand. Schließlich haben agile und traditionelle Methoden und Praktiken ihre Vor- und Nachteile. Mithilfe dessen Kombinationen sollen die Stärken der Arbeitsweisen eingebracht werden. Die traditionelle Entwicklung bringt Klarheit im Prozess und liefert eine gute Struktur für das gesamte Projekt. Agile Entwicklungen bringen hingegen mehr Flexibilität und Individualität, zusätzlich beziehen sich diese mehr auf eine kurze Markteinführungszeit und Kundenzufriedenheit [3].
Aktuell benutzen gemäß der HELENA-Studie[1] über zwei Drittel von den befragten deutschen Unternehmen die hybride Entwicklung [16], weshalb in dieser Arbeit ein Prototyp für Projekte erstellt wird, die noch mit geeigneten Methoden und Praktiken gefüllt werden müssen.

1.1 Problemstellung und Ziel der Arbeit

Wird ein Projekt in der Softwareentwicklung durchgeführt, dann müssen geeignete Methoden und Praktiken ausgewählt werden, damit ein Projekt durchgeführt werden kann. Schon allein die Anzahl der Methoden und Praktiken in der HELENA-Studie ist riesig und beträgt 24 Frameworks[2] und Methoden, sowie 36 Praktiken. Diese Auswahl umfasst allerdings nicht

[1] Bei der HELENA (Hybrid dEveLopmENt Approaches in software systems development)-Studie handelt es sich um eine Umfrage, mit mehr als 75 Forschern, die sehr umfangreiche Befragungen an Praktikern durchgeführt haben, welche unter https://helenastudy.wordpress.com/ kostenfrei verfügbar sind.

[2] Die Frameworks werden in der Arbeit zu den Methoden zusammengefasst.

alle vorhandenen Methoden und Praktiken. Von daher ist die Auswahl groß, dies macht allerdings den Entscheidungsprozess nicht einfacher.
Ein Unternehmen möchte seine Prozesse stets optimieren und mit dem gleichen Aufwand den besten Ertrag erzielen [38]. Dementsprechend ist es gut, wenn ein Prozess optimiert ist. Dazu zählt die bestmögliche Auswahl von Methoden und Praktiken. Diese Auswahl erfordert einen Aufwand und eine Expertise. Die Expertise soll hoch sein und der Aufwand gering gehalten werden.
Von daher ist das Ziel der Arbeit, dass anhand von geeigneten Kriterien Methoden und Praktiken für hybride Entwicklungsansätze entwickelt werden. Das bedeutet konkret, dass ein Praktiker ein Projekt hat und mithilfe eines Prototyps die Methoden und Praktiken erstellen soll. Dabei ist es wichtig, dass der Aufwand für den Praktiker verringert wird und zum Aufbauen des Projektes soll ihm die Ausgabe helfen. Der Einsatz eines solchen Prototyps rechtfertigt sich durch eine entsprechende Arbeitserleichterung.

Forschungsfrage

Inwiefern ist es möglich, auf Grundlage der Organisation Methoden und Praktiken für hybride Entwicklungsansätze zu finden?

1.2 Lösungsansatz

Es gibt bereits einige Literatur zu hybriden Entwicklungsansätzen, weshalb diese den Grundstein für diese Arbeit liefert. Die vorhandene Literatur wird hierbei genutzt, um drei Untersuchungsfragen zu klären, anhand derer der Prototyp erstellt wird:

1. Wie wird hybride Entwicklung organisiert?
2. Welche Kriterien gibt es, um Methoden und Praktiken auszuwählen?
3. Welche Methoden und Praktiken lassen sich auswählen bzw. kombinieren?

Anhand der Literatur werden diese Fragen beantwortet und ein Konzept entwickelt, um einen Prototyp zu entwickeln, der dann verwendet werden kann. Dieser Prototyp wird als eine Webanwendung implementiert.
Um dies zu evaluieren, wird eine Studie durchgeführt, wo Experten konkret befragt werden, wie sie Methoden und Praktiken auswählen würden. Dies wird dann anschließend mit den Methoden und Praktiken der Software verglichen, um einen Überblick zu bekommen, wie sinnvoll die Ausgabe bzw. der Prototyp ist.

1.3 Ergebnisse der Arbeit

In dieser Arbeit wird ein Prototyp umgesetzt, der für hybride Entwicklungen konkrete Methoden und Praktiken vorschlägt. Dieser Prototyp basiert zunächst einmal auf der Organisation von Prenner et al. [30]. Hierbei wird klar, dass in der hybriden Entwicklung das Wasserfallmodell, Scrum und iterative Development die Basis bilden. Mithilfe der HELENA-Studie werden dann die weiteren Methoden und Praktiken kombiniert. Diese basieren auf den häufigsten Kombinationen, die in der HELENA-Studie angegeben wurden, welche bei Tell et al. [42] verschriftlicht wurden. Klare Kriterien für weitere Methoden und Praktiken konnten nicht gefunden werden. Dennoch konnte mithilfe der Studie der Eindruck bestätigt werden, dass es schwierig ist, weitere Methoden und Praktiken auszuwählen. Des Weiteren konnten weitere Forschungsfragen entwickelt werden, um die Forschung des Themengebietes weiter voranzutreiben.
Neben den gewünschten Methoden und Praktiken bietet das Tool zusätzlich die Information wie Anforderungen erhoben (kontinuierlich vs. Up-front), die Architektur erstellt (kontinuierlich vs. Up-front) und getestet bzw. integriert werden soll (separat vs. integriert). Indem das Wasserfallmodell konkretisiert wird, schafft dies eine Grundlage für die Auswahl weiterer Methoden und Praktiken, die möglicherweise für zukünftige Untersuchungen von Nutzen sein könnten.

1.4 Struktur der Arbeit

In Kapitel 2 werden die Grundlagen für die Arbeit erläutert, auf denen die Software aufgebaut wird. Anschließend werden verwandte Arbeiten zu dem Themengebiet in Kapitel 3 erläutert, sowie die Unterschiede von der Arbeit zu den bisherigen Arbeiten dargestellt. Da somit die Quellenlage klar ist, wird dann in Kapitel 4 ein Konzept erklärt, auf dem die Software basiert. Dies muss dann noch implementiert werden und was dort zu beachten ist, wird in Kapitel 5 erklärt. Die Evaluation dieses Prototyps wird mithilfe von Experteninterviews in Kapitel 6 erläutert. Diese Ergebnisse werden dann ausführlich in Kapitel 7 diskutiert, bevor im letzten Kapitel 8 die wesentlichen Ergebnisse der Arbeit zusammengefasst werden.

Kapitel 2

Grundlagen

In diesem Kapitel werden die Grundlagen für die Auswahl von verschiedenen Praktiken und Methoden dargestellt. Anfänglich wird betrachtet, wie hybride Entwicklungsmethoden organisiert sind. Danach werden die jeweiligen Ziele und Herausforderungen in hybriden Entwicklungsmethoden erläutert. In Teilkapitel 2.5 wird erklärt, inwiefern einzelne Methoden und Praktiken Disziplinen agiler machen können. Zuletzt werden in diesem Kapitel häufige Kombinationen von Methoden und Praktiken beschrieben.

2.1 Organisation für hybride Entwicklung

Durch eine systematische Literaturanalyse von Prenner et al. [30] hat sich herausgestellt, dass es drei unterschiedliche Ansätze gibt, wie hybride Entwicklungsmethoden organisiert werden. Diese nennen sich der Wasserfall-Agile-Ansatz, der Wasserfall-Iterative-Ansatz und der Pipeline Ansatz. Diese Ansätze werden im Folgenden näher erläutert.

2.1.1 Wasserfall-Agile-Ansatz

Der Wasserfall-Agile Ansatz besteht aus sechs Phasen: der Anforderungsanalyse, dem Design, der Erstellung des Backlogs, der Entwicklung, dem Testen und der Auslieferung. Die einzelnen Phasen werden, wie beim Wasserfallmodell, nacheinander abgearbeitet, wobei die Entwicklung iterativ abläuft. Nachfolgend werden die einzelnen Phasen nach Prenner et al. [30] beschrieben.

Requirements Analysis und Planning Phase. Als Erstes werden die übergeordneten Entscheidungen getroffen, das bedeutet, dass nicht sonderlich ins Detail gegangen wird. Dies passiert meistens mit einem Product Owner zusammen. Die grundlegenden Anforderungen werden häufig in User Stories oder Use Cases formuliert, um sie für das Backlog

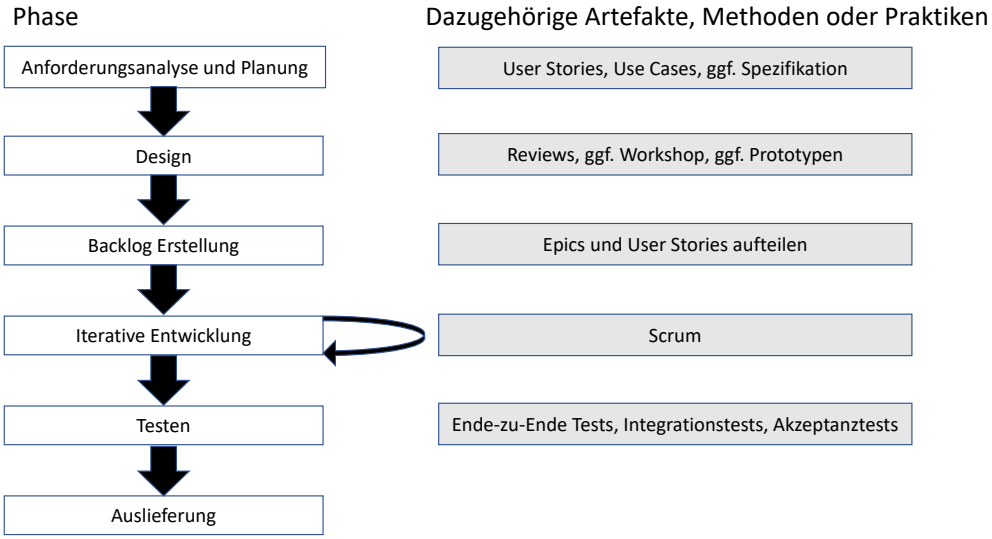

Abbildung 2.1: Wasserfall-Agile-Ansatz nach Prenner et al. [30]

vorzubereiten [25, 36, 41, 46]. Allerdings gibt es auch einen Ansatz in Millard et al. [22], der eine Spezifikation schreibt. Weitere Ziele für diese Phase sind es, dass die Projektparameter definiert werden und die Ressourcen abgeschätzt werden. Das bedeutet, dass benötigte Ressourcen wie Personal, Geld sowie Zeit für das Projekt gemeinsam mit dem Kunden abgeschätzt wird. Meist ist dieser Plan allerdings nicht fest, was zur Folge hat, dass am Ende dieser Phase eine konkrete Struktur steht, die allerdings kontinuierlich ausgebessert werden kann.

Initial Design Phase. Beim Design geht es zunächst einmal, um die Klärung der zu verwendeten Technologien und Tools. Um Risiken zu vermeiden, können dafür Prototypen erstellt werden. Als einzelne Artefakte werden häufig Diagramme erstellt, wie Klassendiagramme oder Sequenzdiagramme. Aber auch die einzelnen Interfaces können bereits erstellt werden. Das Design kann laut Sultana et al. [39] als Workshop initialisiert werden, wird allerdings häufig als eine eigene Phase gesehen. Häufig wird die Phase des Designs reviewt, womit am Ende eine detaillierte Beschreibung der Architektur vorliegt, die zusätzlich geprüft wurde.

Backlog Creation Phase. In der Phase werden die Artefakte genutzt, die bereits in der Anforderungsanalyse erstellt worden sind. Epics und User Stories sind hier meistens schon vorhanden, sodass das Backlog vom Product Owner gefüllt werden kann. Bei Millard et al. [22] und Nisa et al.

[25] ist in dieser Phase der Scrum Master als Hilfe eingesetzt.

Agile Development Phase. Die Entwicklungsphase ist im Wasserfall-Agilen-Ansatz mit agilen Methoden gestaltet. Meistens wird hier Scrum genutzt [20, 22, 46]. Jede Iteration startet mit einem Planmeeting, wo eine Teilmenge der Anforderungen für die jeweilige Iteration zum Entwickeln ausgewählt wird. Falls es Unklarheiten gibt, können hier noch einmal die Anforderungen genauer erhoben werden [20, 28, 46]. Sind die Anforderungen eindeutig, dann schätzt das Entwicklerteam die einzelnen Aufgaben ab und die einzelnen Entwickler bekommen die jeweiligen Aufgaben zugewiesen. Innerhalb der Iteration wird dann das detaillierte Design erstellt [21, 33, 37], das jeweilige Feature implementiert[1, 6, 22, 28], integriert und die einzelnen Tests durchgeführt [12, 14, 20, 46], wobei es auch häufig zu Reviews kommen kann [20, 21, 32, 33]. Einerseits können die Tests durch das Entwicklerteam selber [28], andererseits auch separat durchgeführt werden [21]. Da auf die Nutzung von Scrum zugegriffen wird, gibt es auch tägliche Stand-ups und regelmäßige Backlog-Grooming-Meetings [20, 22, 46], sowie häufig einen Scrum Master [22]. Die Dauer der einzelnen Iterationen ist flexibel, häufig sind es zwei [14, 41], drei [22] oder vier Wochen [32]. Am Ende der jeweiligen Iteration werden die erzeugten Ergebnisse dem Product Owner oder dem Kunden in einem Review Meeting vorgestellt, zudem gibt es eine Retroperspektive der Iteration [20, 46]. Ist dies erledigt, dann geht es an den Beginn der Iteration, bis alle Anforderungen erfüllt sind.

Testing Phase. Sind alle Anforderungen erfüllt, dann kommt es zu der Phase des Testens. Zwar wurden bereits die Unit-Tests ausgeführt, allerdings fehlen noch die Tests zur Verifikation. Hierbei sind Ende-zu-Ende Tests [20], Integrationstests [21] oder aber auch Akzeptanztests [6, 12, 40] möglich.

Operations Phase. Zu guter Letzt kommt es zu der Auslieferung. Dort muss dann die Software deployed und den jeweiligen Nutzern Zugriff für die Software gewährleistet werden [1, 9, 20, 21]. Des Weiteren muss eine Dokumentation erstellt und der Wartungssupport vorbereitet sein [46]. Falls notwendig ist ebenso möglich, dass der Nutzer auf die Software trainiert wird [35]. Somit steht mit der letzten Phase die Software und der Prozess ist beendet.

2.1.2 Wasserfall-Iterative-Ansatz

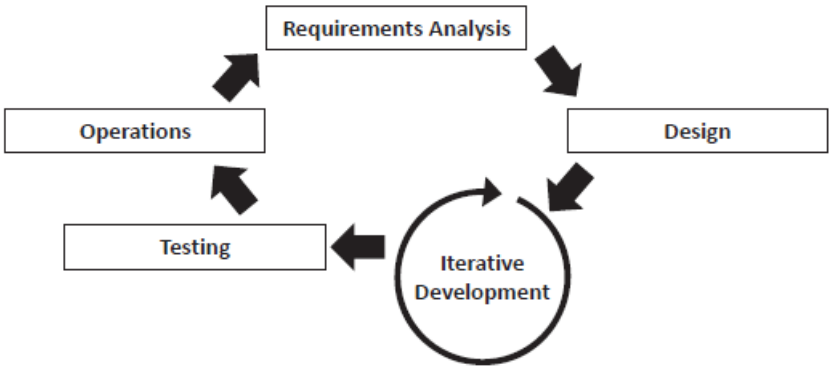

Abbildung 2.2: Wasserfall-Iterative-Ansatz aus Prenner et al. [30]

Der Ansatz durchläuft fast die gleichen Phasen wie der Wasserfall-Agile-Ansatz. Lediglich die *Backlog Creation Phase* fällt weg. Zudem laufen alle Phasen hintereinander iterativ ab. Das bedeutet, dass nach der Auslieferung nicht abgeschlossen wird, sondern es wieder zurück in die Anforderungsanalyse geht, was zusätzlich auch der Abbildung 2.2 entnommen werden kann.

Requirements Analysis und Planning Phase Die Phasen laufen ähnlich zu dem Wasserfall-Agilen-Ansatz ab. Allerdings beziehen sich die erhobenen Anforderungen [6, 25, 33] und Definitionen der Ziele [33] auf die jeweiligen Iterationen. Da die Phase sich nur auf die Iteration bezieht, ist diese auch wesentlich kürzer als im Wasserfall-Agilen-Ansatz. Ebenso werden die einzelnen Aktivitäten detaillierter geplant. Als Dokumente werden hier zumeist Use Cases und User Stories genutzt, wobei auch hier Rong et al. [33] eine Spezifikation für jede Iteration erstellen. Aufgabe dieser Phase ist es auch hier den Aufwand, die Zeit, die Kosten, die Risiken und die benötigten Ressourcen der Iteration zu schätzen [6, 37]. Am Ende der Phase werden die einzelnen Anforderungen in Aufgaben verteilt, analysiert und in einem Projektplan für die Iteration definiert. [12, 39]

Initial Design Phase. In der Designphase geht es darum, dass die Architektur erstellt wird. Allerdings nur für die aktuelle Iteration. Die Entscheidungen sind erneut auf einer höheren Ebene, da die Details, genauso wie im Wasserfall-Agilen-Ansatz, in der Entwicklung entschieden werden [6, 12, 37]. Ebenfalls kann es hierbei dazu kommen, dass bereits Interfaces definiert werden [32].

Agile Development Phase und Testing Phase. In der Entwicklung und bei dem Testen sind die Phasen analog zu denen vom Wasserfall-Agilen-Ansatz umgesetzt. Deswegen werden die Phasen nicht noch einmal weiter beschrieben.

Operation Phase. Zu guter Letzt wird die Software ausgeliefert. Dafür wird ein Review durchgeführt und eine Entscheidung für die Weiterentwicklung durchgeführt [37, 39]. Auch hier wird die Software deployed [37, 39] und die Dokumentation erstellt [6, 12, 40]. Zudem wird noch einmal die vergangene Iteration analysiert [37, 39], bevor es anschließend wieder in die nächste Iteration geht.

2.1.3 Pipeline-basierter-Ansatz

Requirements Engineering	N	N+1	N+2	N+3	N+4
Design	N-1	N	N+1	N+2	N+3
iterative Development	N-2	N-1	N	N+1	N+2
Testen	N-3	N-2	N-1	N	N+1
Auslieferung	N-4	N-3	N-2	N-1	N
Iteration	I-2	I-1	I	I+1	I+2

Abbildung 2.3: Pipeline-basierter-Ansatz aus Prenner et al. [30]

Der große Unterschied zwischen dem Pipeline-Ansatz und den anderen beiden Ansätzen ist, dass die Phasen nicht sequentiell ausgeführt werden, sondern parallel. Das bedeutet, dass dies inkrementeller ist.

Requirements Analysis Phase. Auch hier werden die Anforderungen erhoben, allerdings geht es hier nur um einzelne Features. In dieser Phase gibt es zwei unterschiedliche Ansätze. Read and Bick [32] schreibt über Product Management Meetings, in denen neue Features diskutiert werden. Dafür setzen sich der Product Owner, UI-Experten, Softwarearchitekten und der Scrum-Master zusammen und erstellen Epics, die vom Product Owner priorisiert werden. Heikkilä et al. [14] hingegen beschreiben eine Portfolio Steering Gruppe, die in Meetings neue Features einführen. Dabei sind der Chef Product Owner, das Portfolio Management, technische Management und Qualitätsmanagement beteiligt. Nach dem Meeting schreibt ein technischer Experte auf einer Seite einen Überblick über die neuen entwickelten Features.

Initial Design Phase. Bei Heikkilä et al. [14] werden in dieser Phase die Anforderungen erfasst und ein grobes Design für das Feature beschrieben.

Anschließend werden für die Anforderungen die Epics und zugehörigen Stories geschrieben, um sie den einzelnen Entwicklerteams zuweisen zu können.
Read und Briggs [32] lassen den Softwarearchitekten eine komplexe Analyse des Features durchführen und Zaki und Moawad [46] lassen die Architektur für das neue Feature entwickeln.

Agile Development Phase. Die Umsetzung dieser Phase ist genauso wie im Wasserfall-Agilen-Ansatz und Wasserfall-Iterativen-Ansatz mit Sprints und Scrum umgesetzt.

Testing Phase. Die Phasen zum Testen werden als formale Verifikationsphase beschrieben, wo Integration- und Regressionstests durchgeführt werden [27, 32].

Operations Phase. Die Auslieferung wird nach Heikkilä et al. [14] mit der Integration der neuen Features und der nötigen Dokumentation beschrieben.

2.2 Ziele und Herausforderungen für hybride Entwicklung

Die systematische Literaturanalyse von Prenner et al. in [31] ergibt, welche Ziele und Herausforderungen Einflüsse auf hybride Entwicklungen haben.

- Management von Anforderungsänderungen
- Konflikt zwischen Up-front und continuous Requirements Engineering
- Konflikt zwischen Up-front und continuous Architektur und Design
- Konflikt zwischen zentralen Entscheidungen und selbstorganisierten Teams
- Schwierigkeiten einer separaten Testphase
- Spannungen zwischen der Geschäftsführung und der Entwicklungsabteilung
- Konflikt zwischen Lang- und Kurzzeitplanung
- Konflikt zwischen explizierter Dokumentation und explizitem Wissen

Die einzelnen Herausforderungen sind nichts anderes als Konflikte, die eine Lösung benötigen. In denen wird entschieden, ob ein Prozess agiler

oder traditioneller sein soll. Beispielsweise bedeuten viele Anforderungsänderungen, dass die hybride Entwicklung agiler sein sollte. Die einzelnen Herausforderungen stellen sich als Konflikt für folgende Bereiche dar:

- Requirements Engineering
- Architektur
- Planung
- Koordination
- Dokumentation
- Testen und Auslieferung

Um diese einzelnen Herausforderungen für die jeweiligen Bereiche zu lösen, gibt es eine Liste von Zielen, die den einzelnen Herausforderungen zugeordnet werden können. Die komplette Liste der Herausforderungen und Ziele gibt es in dem Paper von Prenner et al. [31] zu lesen. Um ein Verständnis darüber zu bekommen, wird der Konflikt über die Anforderungserhebung erläutert.

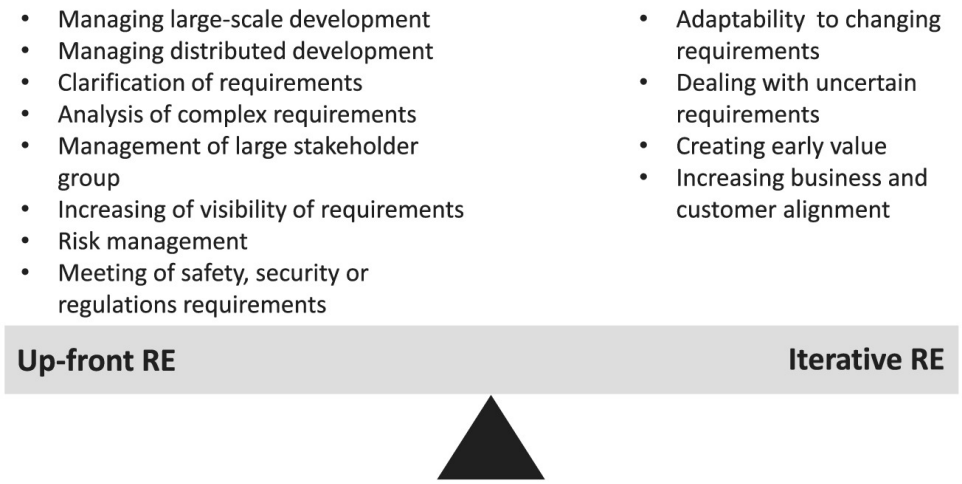

Abbildung 2.4: Up-front Anforderungserhebung und iterative Anforderungserhebung mit den einhergehenden Zielen aus Prenner et al. [19]

An der Entscheidung, wie eine Herausforderung in der hybriden Entwicklung bewältigt wird, hängen demzufolge viele Faktoren ab. In diesem Fall gibt es acht Ziele, die für Up-front Anforderungserhebung sprechen und vier, die für iterative Anforderungserhebung sprechen. Anhand dieser Ziele kann festgelegt werden, ob Up-front (traditioneller Ansatz) oder iterativ (agiler Ansatz) in der Anforderungserhebung gehandelt werden soll.

Diese Abwägung von Zielen gibt es für die anderen sieben Herausforderungen ebenfalls. Auch hier gibt es Ziele, die für eine agile oder eine traditionelle Arbeitsweise sprechen. Wie dann die genaue Umsetzung erfolgen soll, muss auch dort abgewogen werden.

2.3 Frames von hybrider Entwicklung

In Prenner et al. [29] wurden Interviews mit 15 Praktikern durchgeführt. Anhand dieses Papers wurde eine Theorie abgeleitet. Die besagt, dass die hybride Entwicklung mithilfe von Frames definiert wird.
Diesem liegt zu Grunde, dass die einzelnen Projekte anfänglich die groben Grenzen für ein Projekt definieren wollen. Das sind das Requirements Engineering, die Architektur, die Planung, das Testen und der Koordinationsrahmen. Die Frames sollten zuvor schon bekannt sein, denn schließlich finden sich diese auch in den Herausforderungen aus dem Teilkapitel 2.2 wieder. Allerdings gibt es dort ein paar neue Aussagen und Erkenntnisse. Schließlich nehmen die einzelnen Phasen Einfluss auf die Wahl der Methoden und Praktiken.
In der **Planung** geht es darum, dass das Unternehmen einen Rahmen für das Projekt festlegt. Dazu gehört das Budget, die Fähigkeiten und die Abklärung von Fristen. Die Planung wird in vier Modi unterteilt.
Der **Framed-Agile-Mode** hat einen sehr flexiblen Rahmen und unterscheidet sich zum agilen Vorgehen nur darin, dass Schätzungen für das ganze Projekt durchgeführt werden. Wenn die Anforderungen mit Deadlines gekennzeichnet sind, dann wird das von Prenner et al. [29] **Milestone-Agile-Mode** genannt. Hierbei werden die restlichen Anforderungen um die Meilensteine herum gebaut. Der dritte Modus ist der **Plan-based-Agile-Mode**, bei dem sind die Anforderungen ungewiss, trotzdem soll es aber Planungssicherheit geben. Dort wird dann die Entwicklung in Perioden (drei bis sechs Moden) eingeteilt, in denen der jeweilige Umfang festgelegt wird. Der letzte Modus ist der Execution-Agile-Mode, bei dem die Anforderungen bekannt sind, allerdings ist die Lösung des Projekts unbekannt. Hierbei werden agile Methoden zur Steuerung und Führung genutzt, sodass zum Beispiel der Projektmanager Scrum nutzt, um den Status des Projektes zu kennen. Alle Modi haben als maßgebendste Aufgabe, dass der Status der Entwicklung überprüft werden kann und zusätzlich die Entwicklungsreihenfolge geklärt ist.
In dem **Requirements Engineering Frame** geht es darum, welche Erledigungen in der Up-Front Phase getätigt werden müssen. Es geht häufig darum, alle Anforderungen zu sammeln und sicherzustellen, dass alle erfasst werden. Zudem ist es schwierig, dass nach Beginn des Projektes alle Stakeholder noch einmal erreichbar sind. Des Weiteren ist es sehr wichtig, dass Compliance-Anforderungen und andere nicht funktionale

Anforderungen berücksichtigt werden. Zudem kann es auch sein, dass sich in Details verfangen wird, dadurch muss aufgepasst werden, dass nicht zu detailliert gearbeitet wird. Als Beispiel wird dort die Meinung eines Interviewten genannt, dass über 30.000 Euro nicht diskutiert werden müssen, wenn im Millionenbereich entwickelt wird. Zudem gibt das Up-front Requirements Engineering Frame einen genauen Aufschluss darüber, welche Zeit, Kosten und Fähigkeiten benötigt werden, um das Projekt durchführen zu können. Alle Interviewten, die ein neues Projekt anfingen, haben eine Up-front Requirements Engineering benötigt.
Im **Architektur Frame** geht es darum, wie die einzelnen Schnittstellen definiert werden. Wird die Pipeline Methode angewandt, dann gibt es sogar häufig keine Architekturphase. Diese entwickelt sich einfach mit der Implementierung. Zudem ist von Relevanz, was dokumentiert wird. Wichtig ist hierbei, dass nur dokumentiert wird, was dabei zum Verständnis der Software beiträgt.
Im **Test Frame** ist es wichtig, dass entschieden wird, wer was testet. Sicherlich soll der Zeitraum zwischen Implementierung und Tests verkürzt werden, um die Tests nahe an den Entwicklern zu halten. Daher ist es gerne gesehen, dass die Tests integriert und von den Entwicklern durchgeführt werden. Allerdings sorgen eigene Testteams dafür, dass am Ende klar ist, ob die Software bei anderen Personen vollständig funktioniert. Für beide Umsetzungen gibt es weitere Argumente und sind eine Abwägungsentscheidung.
Im **Coordination Frame** werden die Entscheidungen definiert, die vom Team selbst getroffen werden. Die Entscheidungen des Teams enden meist dort, wo es um das gesamte Projekt geht. Dann entscheiden die zentralen Entscheidungsmächte.
Wichtig ist dabei, zu beachten, dass alle Frames in gewisser Weise voneinander abhängen (vgl. Abbildung 2.5). Die genauen Abhängigkeiten der Frames sind in der folgenden Abbildung aufgezeichnet:

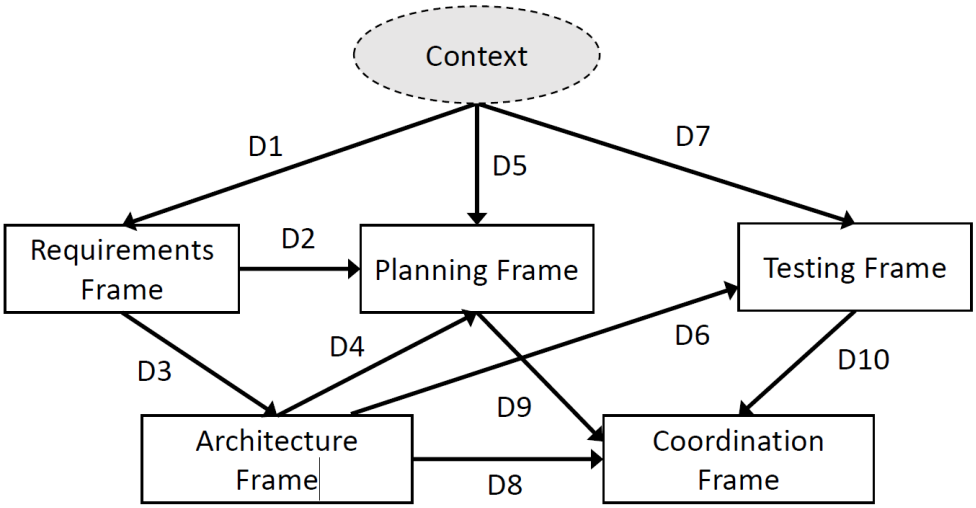

Abbildung 2.5: Die Abhängigkeiten der Frames aus Prenner et al. al.[29]

Demzufolge muss beachtet werden, dass beispielsweise die Anforderungserhebung einen Einfluss auf die Architektur oder auf die Planung nimmt. Es erscheint auch in sich schlüssig und nachvollziehbar, denn je nachdem wie viel Anforderungen vorhanden sind und ob noch welche aufgenommen werden, verschiebt sich auch die Planung bzw. Architektur.
Eine weitere Aussage von Prenner et al. ist in dem Paper, dass er empfiehlt den Wasserfall-Agilen-Ansatz und die Pipeline Methode miteinander zu kombinieren. Eine genauere Beschreibung, wie diese miteinander kombiniert werden, bleibt aus. Sehr wohl wird gesagt, dass in dem Wasserfall-Agilen-Ansatz die Anforderungs- und Architekturphase nur zum Rahmen des Projektes genutzt und die Test sowie die Integrationsphase ausgelassen werden sollen. Des Weiteren soll es einen konkreten Rahmen für die iterative Anforderungserhebung geben, um ein gemeinsames Verständnis zu haben, wie die weiteren Anforderungen aussehen sollen und anschließend umgesetzt werden. Dem kann man entnehmen, dass iterative und Up-front Anforderungserhebungen in einem Projekt kombiniert werden sollen. Außerdem empfiehlt Prenner et al., das Testen so nah wie möglich bei den Entwicklern zu halten, um schnelle Feedbackzyklen zu erhalten und die Behebung von Fehlern zu erleichtern.
Als abschließende Feststellung ist zu erwähnen, dass womöglich nicht die Kombination von agilen und traditionellen Methoden und Praktiken ausreicht, um die hybriden Entwicklungsansätze lösen zu können. Stattdessen sehen die Autoren, dass womöglich neue Methoden und Praktiken definiert werden müssen.

2.4 Kombination von Methoden und Praktiken

Um die jeweiligen Methoden und Praktiken zu kombinieren, liefert die Untersuchung zu der HELENA-Studie weitere Erkenntnisse. In dem Paper von Tell et al. [42] wurden die Daten in Bezug auf die Häufigkeit und deren Kombinationsmöglichkeiten untersucht. Dafür wurden 38 Fragen betrachtet, die von 75 Forschern und Praktikern aus der ganzen Welt gestellt wurden. Für dieses Paper gab es 845 nützliche Datenpunkte von Praktikern, die angaben, welche Methoden und Praktiken in ihren Projekten genutzt werden.

In dem Paper sind drei Fragestellungen von hoher Relevanz. Die erste Fragestellung bezieht sich darauf, welche Frameworks und Methoden die Basis von hybriden Entwicklungsmethoden bilden. Als Zweites wird die Frage gestellt, welche Frameworks, Methoden und Praktiken kombiniert werden, um hybride Entwicklungsmethoden zu bilden. Die dritte und letzte Forschungsfrage von Tell et al. ist, wie hybride Entwicklungsmethoden charakterisiert werden. Diese Frage zielt darauf ab, Kernpraktiken zu finden, die einen Ausgangspunkt für die Entwicklung spezifischer Hybridmethoden bilden.

Damit die Fragestellungen beantwortet werden können, muss zunächst ein Datensatz gebildet werden. Dieser Datensatz setzt sich aus den Methoden und Praktiken zusammen, die mindestens selten genutzt werden. Zudem gibt es einen Filter, welcher bedeutet, dass zum einen betrachtet wird, wer angibt unterschiedliche Entwicklungsansätze in einem Produkt zu kombinieren (mit Filter) und die komplette Menge. Somit auch die Praktiker, die angeben, keine unterschiedlichen Entwicklungsansätze in einem Produkt zu kombinieren (ohne Filter). Dies wird genutzt, da davon ausgegangen wird, dass die Praktiker die Mischung von traditionellen und agilen Methoden ggf. nicht selber merken.
Zusätzlich wurden zwei Annahmen getroffen. Diese Annahmen besagen, dass die Nutzung von hybriden Entwicklungsmethoden weder von der Größe eines Unternehmens, noch von dem Industriezweig des Unternehmens abhängt. Die Annahmen wurden jeweils durch einen χ^2-Test bestätigt.

Zunächst wurde herausgefunden, welche Frameworks und Methoden eingesetzt und kombiniert werden. Dabei wurden alle Methoden ausgewertet, die mit einem Threshold, also zu mindestens 35% genutzt werden. Aus 845 Datenpunkten gaben 792 an, mehrere Frameworks und Methoden zu nutzen (ohne Filter). Dabei wird Scrum am häufigsten verwendet (674 Mal), darauf folgen die iterative Development (620) und Kanban (523). Für die 24 befragten Frameworks und Methoden, ergeben sich 17 Kombinationen, wobei zwei oder drei Methoden miteinander kombiniert werden. Im gesamten Datensatz ohne Filter ergeben sich sogar 27 Kombinationen mit mindestens zwei bis vier genutzten Elementen. Daraus resultieren acht Basismethoden für die hybride Entwicklung, die zu mindestens 35% genutzt werden. Diese Basismethoden sind Scrum, Iterative Development, Kanban, Classic Water-

fall Process, DevOps, eXtreme Programming, Lean Software Development und Feature Driven Development. Dementsprechend ist die Forschungsfrage 1 von den Autoren geklärt. Ebenso gibt es in keinem Szenario mit dem Threshold mehr als vier Methoden, die miteinander kombiniert werden können. Eine mögliche Kombination von den 17 genannten Basismethoden mit Filter, ist die Kombination von Scrum wie in der Abbildung 2.6.

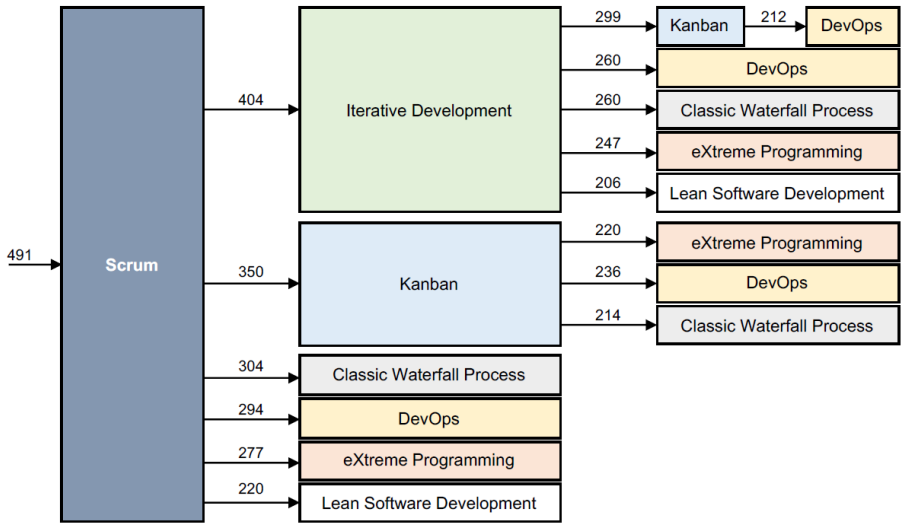

Abbildung 2.6: Überblick der Methoden, der Nutzer, die zu mindestens 35% Scrum und dessen Kombinationen nutzen aus Tell et al. [42]. Das bedeutet, dass die Kombination von Scrum, Iterative Development und DevOps auch zu mindestens 35% genutzt wird.

Aus dieser Abbildung kann entnommen werden, dass 491 Praktiker angeben, Scrum zu nutzen. Von diesen 491 Praktikern nutzen 404 iterative Development. Unter diesen Nutzern befinden sich wiederum 260 DevOps. Alle weiteren Kombinationen sind in der Grafik abgebildet.
Wie Frameworks, Methoden und Praktiken kombiniert werden können, wurden von Tell et. al. als nächstes untersucht. Hierfür wurden als Erstes die jeweiligen Praktiken herausgesucht, die am häufigsten kombiniert werden, ohne dabei die Kombination der einzelnen Methoden zu betrachten. Es stellte sich heraus, dass die kleinste Gruppe mit der höchsten Zustimmung in den Daten das Paar Code Review und Coding Standards ist. Denn dies wird mit einer Zustimmung von 87% genutzt. Dementsprechend haben die Autoren einen Threshold von 85% gebildet, um noch eine kleine Toleranz zu haben. Wird der gesamte Datensatz betrachtet, dann gibt es insgesamt drei Praktiken, die zu mehr als 85% genutzt werden (Code Review, Coding Standards und Release Planning), wobei auch die Paare in Kombination immer eine Nutzung von mindestens 85% haben. Diese werden als die

Basispraktiken bezeichnet.
Werden alle Daten angeschaut, also die ohne Filter, dann gibt es in Summe fünf Praktiken, die zu mindestens 85 % genutzt werden (Code Review, Coding Standards, Release Planning, Automated Unit Testing und Prototyping).
Anschließend wurden die Praktiken herausgezogen, die im Kontext der jeweiligen Basismethoden und Basispraktiken zu 85 % miteinander kombiniert werden. Daraus ergeben sich die 17 möglichen Praktiken bei Nutzung von bestimmten Methoden, die in der Abbildung 2.7 abgebildet sind.

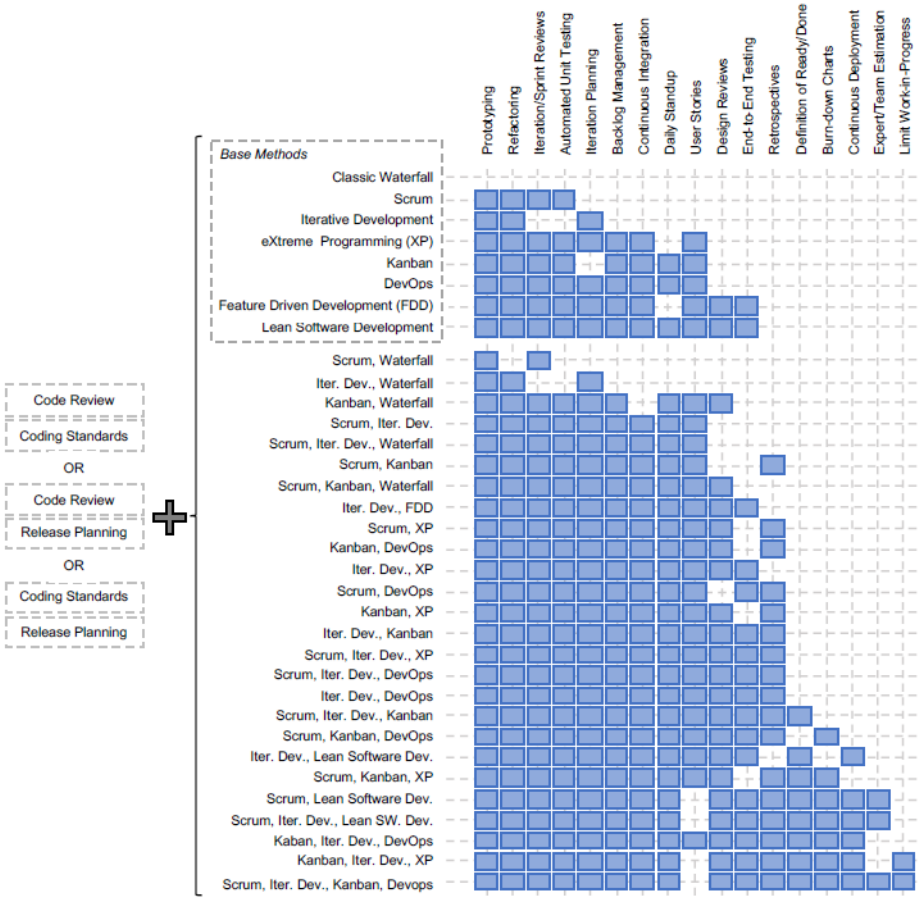

Abbildung 2.7: Die Basispraktiken kombiniert mit den Basismethoden und den Praktiken aus Tell et al. [42]. Markiert sind die Praktiken, die zu mindestens 85% kombiniert werden.

Zudem gibt es noch eine Analyse, die anzeigt, wie oft bei den Basismethoden zwei bis acht Praktiken zu 85% kombiniert wurden. Dies wird in den Abbildungen aus dem Anhang A.2 und A.9 gezeigt. Des Weiteren ist in dieser Grafik abgebildet, unter welchen Basismethoden Praktiken zu

mindestens 85% genutzt werden.

2.5 Agilität von Methoden, Praktiken und Projektdisziplinen

Im Rahmen der HELENA-Studie wurde von Kuhrmann et al. [19] untersucht, inwiefern Methoden und Praktiken einen Einfluss auf die Agilität nehmen. Dabei haben sie sich die Fragen gestellt, wie hoch die Agilität bei der Umsetzung eines typischen Projektes in den einzelnen Projektdisziplinen ist (Forschungsfrage 1) und welche Methoden und Praktiken den sogenannten Grad der Agilität beeinflussen (Forschungsfrage 2).

Der Datensatz ist der gleiche wie bei dem vorherigen Kapitel 2.4 und Paper [42]. Von daher sind es erneut die 75 Forscher und Praktiker, die von den Befragten für diese Auswertung 556 nützliche Datenpunkte bekamen.
Dabei wurden zunächst einmal elf Projektdisziplinen eingeteilt, die sich an den SWEBOK-Kategorien [7] orientieren. Diese lauten: Project Management, Quality Management, Risk Management, Configuration Management, Change Management, Requirements Analysis/Engineering, Architecture and Design, Implementation and Coding, Integration and Testing, Transition and Operation und Maintenance and Evolution. Die einzelnen Fragen für diese Untersuchung beziehen sich erneut auf die bereits bekannten 24 Methoden und 36 Praktiken.
Wie genau die Befragung analysiert wird, ist in der folgenden Abbildung 2.8 abgebildet.

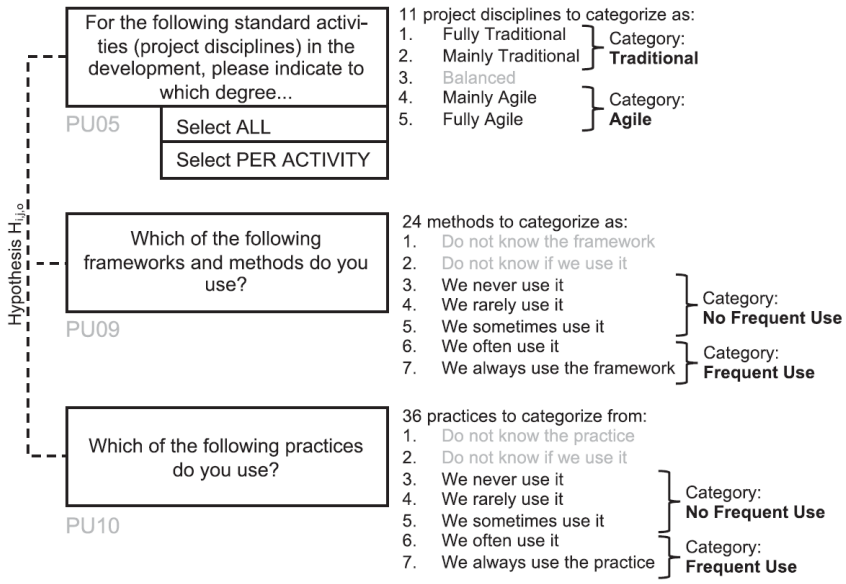

Abbildung 2.8: Analysemodell von Kuhrmann et al. [19]

In der Frage PU05 ist dargestellt, wie die einzelnen Projektdisziplinen in deren Entwicklung ausgeführt werden. Dabei wurde für die Bestimmung des sogenannten den Agilitätsgrades die jeweiligen Antworten wie folgt zusammengefasst:

$$S_{traditional}=S_{fully\_trad(p)} \cup S_{mainly\_trad(p)} \tag{2.1}$$

$$S_{agile}=S_{fully\_agile(p)} \cup S_{mainly\_agile(p)} \tag{2.2}$$

S ist hierbei die Menge der Teilnehmer, die eine Projektdisziplin p traditionell oder agil gestalten.

Auffällig ist bei der Beantwortung der Forschungsfrage 1, somit bei der Betrachtung der einzelnen Projektdisziplinen, dass die meisten Projektdisziplinen einen klaren Trend Richtung agiler Umsetzung zeigen. Die Projektdisziplinen Qualitätsmanagement, Konfigurationsmanagement und die Auslieferung zeigen eine neutrale Tendenz auf. Das Risikomanagement hingegen hat einen Trend zur traditionellen Entwicklung. Alle anderen sieben Projektdisziplinen sind hingegen agil umgesetzt.
Des Weiteren sind 15 % der Teilnehmer in allen Projektdisziplinen entweder komplett agil, traditionell oder neutral.

Zur Beantwortung der zweiten Forschungsfrage wurden die Fragen, PU05, PU09 und PU10 aus der HELENA-Studie genutzt. Das bedeutet, dass die Daten genutzt werden, wie eine Projektdisziplin ausgeführt wird (agil, neutral oder traditionell) und ob Methoden und Praktiken in der Entwicklung genutzt werden.
Wird eine Methode oder Praktik genutzt, dann wird mithilfe der Aussage, ob eine Projektdisziplin genutzt wurde und eines χ^2 Testes der sogenannte Agilitätsgrad berechnet. Anhand dessen wird zu jeder Methode und Praktik in jeder Projektdisziplin ein Agilitätsgrad mit einem χ^2-Wert berechnet. Die einzelnen Agilitätsgrade zu den Projektdisziplinen sind im Anhang A.3 abgebildet. Je geringer der Wert, desto eindeutiger ist der Einfluss. Anhand der Einschätzung, ob eine Projektdisziplin agil, neutral oder traditionell gesehen wird, wird dann eine Methode oder Praktik genutzt wird. Dies wird anhand eines konkreten Beispiels verdeutlicht:
Scrum hat in der Anforderungserhebung einen Agilitätsgrad von $2,85 \cdot 10^{-8}$. Das bedeutet, bei der Nutzung von Scrum ist die Agilität von der Projektdisziplin Anforderungserhebung größer als bei Nichtnutzung. Es gibt allerdings keinen Zusammenhang, ob Scrum in der Anforderungserhebung genutzt wird. Die Aussage, dass Scrum die Anforderungserhebung agiler macht, gibt es nicht, da nur die Projektdisziplin und die Nutzung von Scrum betrachtet wird. Ob eine Methode oder Praktik einen agilen, neutralen oder traditionellen Einfluss hat, wird anhand eines Vergleiches entschieden. Demnach werden die Nutzer betrachtet, die eine Methode oder Praktik

nutzen und die eine Methode oder Praktik nicht nutzen. Wird eine Methode oder Praktik genutzt und ist die Projektdisziplin agiler als wenn die Methode oder Praktik nicht genutzt wird, dann ist der Einfluss agil. Die genaue Aufschlüsselung, welche Methode agil, traditionell oder neutral bei den einzelnen Projektdisziplinen ist, befindet sich im Anhang A.4.
Um den eben genannten Problemen eine stärkere Aussagekraft zu geben, stellen Kuhrmann et al. [19] zwei Annahmen auf.

Annahme 1: Eine Methode oder Praktik, die keinen Einfluss auf eine Projektdisziplin hat, wird nicht die Verteilung des jeweiligen Agilitätsgrades für die jeweilige Projektdisziplin beeinflussen.

Annahme 2: Eine Methode oder Praktik, die einen Effekt auf die Agilität hat, aber nicht in der Projektdisziplin genutzt wird, wird auch keinen Einfluss auf die Projektdisziplin nehmen.

Die erste Annahme wird begründet, indem mithilfe der Nutzung von Feature-Driven Development (FDD) die Agilität betrachtet wird. Gemäß des χ^2-Tests hat die Methode keinen Einfluss auf die Agilität in der Architektur und in dem Design. Von daher wird betrachtet, wie die Agilität bei der Architektur und dem Design ist, wenn diese genutzt bzw. nicht genutzt wird. Anhand des Histogrammes aus 2.10 lässt sich erkennen, dass FDD keinen Einfluss auf die Agilität hat. Das bedeutet, egal ob FDD im Projekt genutzt wird oder nicht, die Verteilung ist ähnlich und somit hat dies keinen Einfluss auf die Architektur und das Design.

Abbildung 2.9: Die Agilität von der Architektur und dem Design bei Nutzung und Nichtnutzung von TDD aus Kuhrmann et al. [19]

Die zweite Annahme wird damit gezeigt, dass das V-Model einen Einfluss auf die Agilität im Risikomanagement hat. Anhand des Histogramms aus der Abbildung 2.10 erkennt man, dass bei Nutzung von dem V-Model eine

klare Verschiebung der Agilität gibt. Dementsprechend bestätigt sich mithilfe dieses Beispieles die Annahme 2. Allerdings weisen die Autoren darauf hin, dass die Annahme kritisch gesehen werden sollte, da diese keine direkte Beziehung zeigt und es Störfaktoren geben kann.

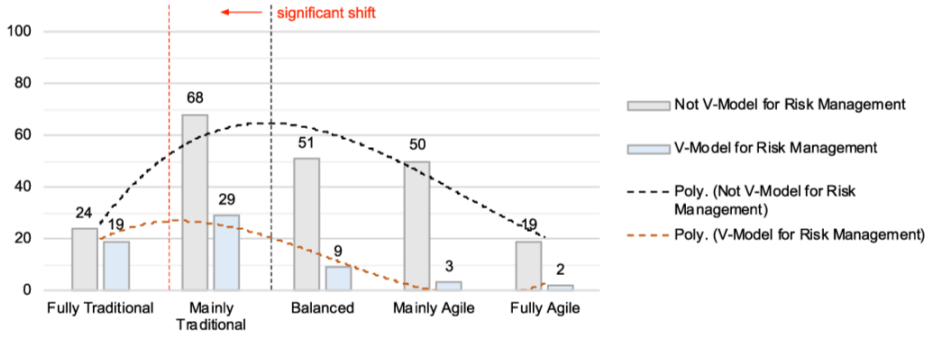

Abbildung 2.10: Die Agilität von dem Risikomanagement bei Nutzung und Nichtnutzung von dem V-Modell aus Kuhrmann et al. [19]

Neben den Forschungsfragen bezüglich des Einflusses der Agilität einzelner Praktiken und Methoden auf Projektdisziplinen, gibt es weitere Erkenntnisse, die hier kurz zugrunde liegen:

- Praktiken haben einen höheren Einfluss auf die Agilität als Methoden.
- Praktiken und Methoden haben einen stabilen Einfluss auf die Agilität, der sich nicht mit der Projektdisziplin ändert.
- Eine Methode oder Praktik entscheidet nicht darüber, ob ein Projekt traditionell oder agil ist. Jede Methode oder Praktik kann in einem Projekt zu finden sein.
- Es gibt einige Methoden, die neutral sind, also keinen Einfluss auf die Agilität haben.

Demzufolge gibt es einige Erkenntnisse bezüglich der Methoden und Praktiken und deren Agilität in den einzelnen Projektdisziplinen. Zudem gibt es auch weitere Aussagen bezüglich der Agilität von den einzelnen Projektdisziplinen. Dennoch ist die Validität eingeschränkt. Bedingt durch eine subjektive Wahrnehmung der Studienteilnehmer. Ebenso sind die Begriffe Agilität und Traditionell nicht näher definiert, weshalb klare Rückschlüsse von Methoden bzw. Praktiken und deren Agilität mit Vorsicht zu genießen sind.

Kapitel 3

Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten beschrieben, um das Thema der Masterarbeit genau einzuordnen. Die Kapitel 2.4 und 2.5 liefern Erkenntnisse aus der HELENA-Studie. Basierend auf dieser Studie wurden die Faktoren untersucht, die Einfluss auf die Auswahl von Methoden und Praktiken nehmen können. Dafür analysierten Klünder et al. [17] die Daten der HELENA-Studie und stellten fest, dass nur wenige Faktoren wie die Projekt-/Produktgröße oder Zielanwendungsbereiche die Auswahl von Methoden beeinflusst.

Dass die hybride Entwicklung in den Unternehmen umgesetzt wird, zeigen Kuhrmann et al. [18]. Denn die traditionelle Umgebung soll einen Rahmen geben, in dem sich bewegt werden kann und zusätzlich die agilen Praktiken und Methoden eingesetzt werden können. Des Weiteren zeigen die Autoren auf, dass die hybride Nutzung nicht von der Größe oder anderen Faktoren eines Unternehmens abhängig ist, sondern auch unabhängig davon genutzt wird. Ähnliches berichten, unabhängig davon Theocharis et al. [44], der benennt die Gründe für die Nutzung von hybrider Entwicklung genauer. Ein Grund dessen ist zum Beispiel auch, dass Personalmanagement, Vertrieb und Vertragswesen verbunden werden muss, was allein durch die agile Entwicklung nicht abgedeckt ist. Daraus resultiert, dass die traditionellen Ansätze unverzichtbar in den strukturellen Bereichen sind.
Weitere Untersuchung der HELENA-Studie lassen sich zudem in Konzentration auf bestimmte Länder finden. Schweden und Uganda [23], Deutschland [16], Dänemark [43], Argentinien [26], Österreich [10], Estland [34] und Costa Rica [2] wurden in unterschiedlicher Literatur untersucht.

Auch außerhalb der HELENA-Studie gibt es einige Untersuchungen zu der hybriden Entwicklung. Beispielsweise analysieren Binder et al. 2014 [5] die Funktionsweise von hybrider Entwicklung, die als wasserfallartig mit

agilen Methoden beschrieben werden.
Schon 2004 haben Cao et al [8] Extreme Programming mit Up-Front Architektur kombiniert. Dabei haben sie herausgefunden, dass die Stabilität durch die Up-Front Architektur gewonnen wird. Denn die Entwickler haben dadurch ein klares Verständnis über das System und deren Abhängigkeiten, die dann einfacher verwaltet werden können.
Bick et al. [4] konzentrierten sich hingegen auf die Koordinationsherausforderungen in Bezug auf die Fehlausrichtungen, die in der Planung von Softwareentwicklung passieren können. Die Empfehlung, die sie ausgeben, ist regelmäßige Treffen zu nutzen, um die Abhängigkeiten im Team bewusst zu machen. Zudem soll die Planung iterativ gestaltet werden, damit die Planung dem Entwicklungsprozess etwas vorausläuft.
In einer Fallstudie zu sicherheitskritischer Software wurde untersucht, was zur Mischung von traditioneller und agiler Entwicklung beigetragen werden kann. Hierbei fanden Heeagar und Nielsen [13] für den spezifischen Fall heraus, dass die Dokumentation auf niedriger Ebene iterativ und die Spezifikation oberflächlich zu schreiben ist.
Mit einem weiteren Beispiel setzten sich Heikkilä et al. auseinander, die den Weg der Anforderungen von der Erhebung bis zum Release betrachteten. Der Release-Projektmanagementprozess lief traditionell ab, Features wurden kontinuierlich entwickelt und der Implementierungsmanagementprozess verläuft agil. Als Vorteile stellen sie verkürzte Entwicklungszeiten, erhöhte Flexibilität, gesteigerte Planungseffizienz, gesteigerte Entwicklermotivation und verbesserte Kommunikation fest. Probleme stellten sie beim Ausgleich des Planungsaufwands, zu viel Engagement, unzureichendes Verständnis der Autonomie des Entwicklungsteams, Definition des Product Owners, Ausgleich der Teamspezialisierung, Organisation der Arbeit auf Systemebene und waschende technische Schulden fest.

Demzufolge gibt es einige Arbeiten, die sich mit der Umsetzung von Entwicklung auseinandersetzen. Dabei gibt es viel im Rahmen der HELENA-Studie, zusätzlich auch einige weitere Paper. Beispielsweise gab es bereits 2004 etwas zu der hybriden Entwicklung herausgefunden. Aber auch einige weitere genannte Paper liefern in den letzten Jahren weitere Erkenntnisse in dem Themengebiet.
Diese Arbeit setzt sich konkret mit den bekannten Papern aus dem Grundlagenkapitel 2 auseinander. Es werden die Arbeiten von Prenner et al. [29, 30, 31] aus den Kapiteln 2.1, 2.2 und 2.3 genutzt. Diese werden benötigt, um eine Struktur darüber zu erlangen, wie hybride Entwicklung aufgebaut ist. Um konkrete Methoden und Praktiken vorschlagen zu können, benötigt es eine grundlegende Struktur. Anhand dieser Struktur wird mithilfe der HELENA-Studie weitergearbeitet. Die Kapiteln 2.4 und 2.5 zeigen, wie Methoden und Praktiken kombiniert werden und welche sich wie agil auf einen Prozess auswirken. Mithilfe dessen wird die hybride Entwicklung

konkret mit Methoden und Praktiken gefüllt.
Bis jetzt war die Forschung lediglich an dem Punkt, dass die vorhandenen hybriden Entwicklungsmethoden analysiert wurden. Das bedeutet, dass die Projekte untersucht werden, wie sie bereits umgesetzt wurden. In dieser Arbeit hingegen wird der Blickwinkel von der anderen Seite betrachtet. Es werden die einzelnen Entwicklungsprozesse mithilfe des aktuellen Wissenschaftsstandes spezifiziert. Dafür werden erstmalig Methoden und Praktiken automatisiert empfohlen, indem die Organisation von Prenner et al. mit der HELENA-Studie kombiniert wird.

Kapitel 4

Konzept

In diesem Kapitel werden die Ideen und Konzepte vorgestellt, um den Software-Prototypen entwickeln zu können. Die Grundlage des Konzeptes besteht aus den in Kapitel 2 vorgestellten Grundlagen. Das lässt erkennen, dass die Basis die Veröffentlichungen von Prenner et al. [30, 31, 29], Tell et al. [42] und Kuhrmann et al. [19] liefern. Wie diese Paper für das Konzept genutzt werden, wird im nachfolgenden Kapitel erklärt.

4.1 Grundlagen des Konzeptes

Um ein geeignetes Konzept für den zu entwickelnden Prototypen zu konstruieren, werden zunächst einmal die wichtigsten Aussagen des Grundlagenkapitels zusammengefasst, damit anhand dessen eine Struktur für das Konzept festgelegt werden kann.
Das Kapitel 2.1 *Organisation für hybride Entwicklung* zeigt auf, wie der grundlegende Aufbau von hybrider Entwicklung ist.
Ziele und Herausforderungen für hybride Entwicklung werden in dem Kapitel 2.2 gezeigt. Konkret heißt es, dass es acht Herausforderungen gibt, die mithilfe von Zielen abgewogen werden müssen, ob sie agiler oder traditioneller gestaltet werden.
Mithilfe von Interviews werden *die einzelnen Frames von hybrider Entwicklung* in Kapitel 2.3 erklärt. Anhand dessen zeigt sich, was in den einzelnen Frames geachtet werden muss und welche Abhängigkeiten zwischen den Frames bestehen.
Die HELENA-Studie zeigt, wie Unternehmen die *Kombination von Methoden und Praktiken* durchführen. Anhand der Häufigkeiten wurden einzelne Basismethoden, Basispraktiken und häufige Kombinationen in Kapitel 2.4 gezeigt.
Was den Agilitätsgrad ausmacht, zeigt sich im Kapitel 2.5, indem die *Agilität von Methoden, Praktiken und Projektdisziplinen* aufgezeigt wird. Anhand dessen sieht man die Agilität von Projektdisziplinen und wie sich diese in

Verbindung zu Methoden und Praktiken verhalten.

Für die nachfolgenden Kapitel wurden die einzelnen Kernaussagen der Grundlagen extrahiert und zu einem Konzept zusammengefügt. Dabei fängt das Konzept zunächst einmal mit der Auswahl der Methoden 4.2 und Praktiken 4.3 an. Anhand dessen lassen sich bereits vor der Implementierung einige Schwächen feststellen, die erläutert werden. Zudem wird das Konzept im Teilkapitel 4.5 um eine weitere Funktionalität erweitert.
Da viele Faktoren auf das Konzept einen Einfluss nehmen, werden die Grundsätze der Software in einzelnen Axiomen festgehalten. Zudem wird ein Informationsfluss der kommenden Teilkapitel in der Zusammensetzung der Teilkonzepte zusammengefasst, damit eindeutig erkennbar ist, anhand welcher Daten welche Entscheidungen getroffen werden.
Ein Hinweis ist, dass lediglich die Methoden und Praktiken der HELENA-Studie angewandt werden. Dies ist der Fall, da die Kombination der Methoden und Praktiken bzw. die Literatur auf diese Menge ausgerichtet sind.

4.2 Auswahl der Methoden

Der erste Schritt eines jeden Projektes sollte es sein, eine grundlegende Struktur festzulegen. Zum Thema Organisation von hybrider Entwicklung wurde von Prenner et al. [30] bzw. im Kapitel 2.1 bereits einiges erläutert. Es handelt sich dabei um drei unterschiedliche Ansätze, wie hybride Entwicklung organisiert wird. Das sind der Wasserfall-Agile Ansatz, der Wasserfall-Iterative-Ansatz und der Pipeline-Ansatz. Laut der Interviews von Prenner et al. ist eine Kombination dieser Ansätze möglich und in der Praxis auch übliche Anwendung. Dies wird in einem weiteren Paper [29] noch einmal bestätigt.
All diese Ansätze sind Ansätze nach dem Wasserfallmodell. Zudem sind die Umsetzungen der einzelnen Modelle sehr ähnlich. Genauer betrachtet hat der Wasserfall-Agile-Ansatz im Gegensatz zu den anderen zwei Ansätzen eine zusätzliche Phase, die der Backlog Erstellung. Ansonsten sind die Phasen in allen Ansätzen gleich und werden auch in derselben Reihenfolge durchgeführt:

1. Requirements Engineering
2. Design Phase
3. Iterative Development
4. Testen
5. Auslieferung

Die grundlegenden Unterschiede der einzelnen Ansätze liegen in der Umsetzung. Denn der Wasserfall-Iterative-Ansatz durchläuft einen Zyklus, der Wasserfall-Agile Ansatz durchläuft lediglich in der Entwicklung einen Zyklus und der Pipeline-Ansatz ist wie der Wasserfall-Iterative Ansatz, wobei alle Phasen parallel ablaufen. Da dies die grundlegende Struktur ist, wird in dem Prototyp ebenso die Wasserfall-Methode umgesetzt.

Axiom 1

Die Wasserfall-Methode wird immer in dem Prototyp verwendet.

Wie an der eben genannten Reihenfolge entnommen werden kann, wird in allen drei Ansätzen die Entwicklung iterativ eingesetzt. Dies ist ebenfalls eine Methode und wird deswegen ebenso in dem Prototyp umgesetzt.

Axiom 2

Das iterative Development wird immer in dem Prototyp verwendet.

In den jeweiligen Ansätzen (Wasserfall-Agil, Wasserfall-Iterativ und Pipeline) wird Scrum in der Praxis eingesetzt. Zudem ist eine Aussage von Prenner et al. in [31], dass Scrum in der hybriden Entwicklung immer eingesetzt wird, deswegen wird diese Methode auch in dem Prototyp verwendet.

Axiom 3

Scrum wird immer in dem Prototyp verwendet.

Durch die Verwendung von dem Wasserfallmodell, der iterativen Entwicklung und Scrum gibt es bereits drei Methoden, die immer verwendet werden. Die Frage, die sich stellt, ist, ob noch weitere Methoden angewandt werden sollten und anhand welcher Kriterien dies entschieden werden kann. Für diese Entscheidung werden die Ergebnisse im Rahmen der HELENA-Studie von Tell et al. [42] aus Kapitel 2.4 benötigt. Hier wurden die gängigen Kombinationen von Methoden und Praktiken aufbereitet. Eine Feststellung ist es, dass es keine Kombination von fünf oder mehr Methoden gibt, in denen mindestens 35% der Studienteilnehmer kombinieren werden. Werden die Personen betrachtet, die angaben, unterschiedliche Entwicklungsansätze in einem Produkt zu kombinieren, dann gibt es nicht einmal vier Methoden, die eine Übereinstimmung von mindestens 35% haben. Die Personen, die angaben, unterschiedliche Entwicklungsansätze nicht in einem Produkt zu kombinieren, haben immerhin Kombinationen von vier Methoden. Das wiederum bedeutet, dass es entweder drei oder vier Methoden in einem Projekt geben kann. Doch die Menge mit den vier Methoden gibt nicht an, unterschiedliche Entwicklungsansätze zu kombinieren. Des Weiteren gibt es

auch in dieser Menge keine Kombination mit vier Methoden, die Scrum, Wasserfall und iterative Development enthalten. Das bedeutet, dass die HELENA-Studie dagegen spricht, weitere Methoden zu verwenden. Selbst wenn es sinnvoll wäre, gibt es sonst keine Möglichkeit, die HELENA-Studie zu verwenden. Diese sehen eine Menge von Methoden vor, die dann mit den einzelnen Praktiken kombiniert werden. In Folgerung dessen werden nur drei Methoden verwendet.
Ein weiterer Punkt, der dafür spricht, dass keine weiteren Methoden verwendet werden, ist, dass die Methoden einen geringeren Einfluss auf die Agilität haben als Praktiken (vgl. Kuhrmann et al. [19]). Das bedeutet, dass die Methode einen geringeren Einfluss bei der Agilität hat als die Praktiken. Somit sollte, wenn an der Agilität etwas verändert werden soll, an den Praktiken verändert werden, da diese einen höheren Einfluss haben.
Mithilfe des letzten Axioms ist die Wahl der Methoden abgeschlossen.

Axiom 4

Als Methoden werden in dem Prototyp in Kombination das Wasserfallmodell, Scrum und das iterative Development verwendet. Es werden keine weiteren Methoden genutzt.

4.3 Auswahl von Praktiken

Da die Methoden bereits entwickelt sind, fehlen lediglich für die Balance eines hybriden Projektes eine Auswahl von Praktiken. Die HELENA-Studie hat mit ihren Daten in Tell et al. [42] gezeigt, welche Methoden am häufigsten mit welchen Praktiken genutzt werden. Dies bildet auch die Grundlage für die nachfolgende Selektion. Die Praktiken werden aus diversen Gründen häufig mit den jeweiligen Methoden kombiniert, dies wird sich an dieser Stelle zu nutzen gemacht.
Es zeigte sich bereits, dass drei Praktiken quasi immer genutzt werden. Diese bezeichnen sich als die Basispraktiken und wurden bereits in dem Teilkapitel 2.4 erläutert. Das sind Code Review, Coding Standards und Release Planning.
Tell et al. haben anschließend diese Basispraktiken als Basis genommen und um herauszufinden, welche Praktiken zu 85 % genutzt werden, bei Auswahl von zwei der drei Basispraktiken. Dies ist in Abbildung 2.4 zu finden.
Für die Arbeit ist die Zeile relevant in denen Wasserfall, Scrum und iteratives Development genutzt werden. Aus dieser Abbildung ergeben sich neun weitere Praktiken, die zu den eben genannten 85% genutzt werden. Dementsprechend steht eine Menge von Praktiken zur Verfügung, die häufig unter den ausgewählten Methoden verwendet werden. Zusätzlich werden diese auch in dem Prototyp verwendet.

Axiom 5

Die Praktiken werden aus zwei der drei Basispraktiken (Code Review, Coding Standards und das Release Planning) und acht weiteren Praktiken (Prototyping, Refactoring, Automated Unit Testing, Iteration Planning, Backlog Management, Continuous Integration, Daily Standup, User Stories und Design Reviews) gebildet.

Bevor weitere Entscheidungen getroffen werden, müssen die einzelnen Praktiken zunächst noch einmal einzeln in Betracht gezogen werden. In dieser Menge, der Menge der Praktiken, ist unter anderem das Iteration Planning enthalten. Da in den Methoden bereits das iterative Development fest eingesetzt wird, ist das Iteration Planning unerlässlich.

Axiom 6

Iteration Planning wird als Praktik fest eingesetzt.

Eine weitere Praktik ist Continuous Integration genutzt wird, das hängt mit der Umsetzung des Wasserfallmodells zusammen. Denn wenn kontinuierlich integriert und getestet wird, dann muss dieses auch verwendet werden.

Axiom 7

Ob Continuous Integration genutzt wird, hängt von der Umsetzung des Wasserfallmodells ab.

Inwiefern die weiteren Praktiken verwendet werden können, kann nicht einfach aus- oder eingeschlossen werden. Backlog Management wird grundsätzlich beim Scrum genutzt. Allerdings ist dies auch eine Frage, inwiefern Scrum dort umgesetzt wird. Beispielsweise untersuchten Hron und Obwegeser [15], die unterschiedlichen Modifikationen von Scrum.
Design Reviews ist ebenso ein Sonderfall, da dies keinen Einfluss auf die Agilität nimmt. In Kuhrmann et al. [19] wird dies mithilfe des Agilitätsgrades analysiert und laut diesem hat die Praktik keinen Einfluss auf die Agilität im Projekt. Von daher ist die Auswahl der Praktik mit einer geringeren Bedeutung versehen.
Die ersten drei Praktiken sind verdeutlicht, doch sechs Praktiken müssen noch eingesetzt werden. Dafür wird eine weitere Frage gestellt, wie agil das Projekt sein soll. Anhand dessen soll entschieden werden, wie viele Praktiken von den sechs zusätzlich eingesetzt werden. Mehr Praktiken bedeuten mehr Agilität. Dies wird anhand des Agilitätsgrades aus Kuhrmann et al. [19] angekommen. Die Nutzung einer Praktik hat einen Einfluss auf die Agilität

in einer Projektdisziplin. Werden mehrere Praktiken genutzt, dann sollte daraus resultieren, dass der Einfluss dementsprechend größer sein müsste. Entsprechend müssten mehrere Praktiken automatisch mehr Einfluss auf die Agilität bedeuten.

Axiom 8

Die Anzahl der weiteren Praktiken hängt von der gewünschten Agilität des Projektes ab. Dabei wird zwischen weiteren null und vier Praktiken empfohlen.

Das bedeutet, dass aktuell lediglich für die Software eine Einstufung der Agilität benötigt wird. Ist die Agilität *wichtig* für das Projekt, werden vier weitere Praktiken empfohlen. Bei der Einstufung *weniger wichtig* werden drei Praktiken empfohlen, zwei Praktiken bei *neutral* eingestuft und *eher nicht wichtig* umfasst eine Praktik. Bei *nicht wichtig* wird empfohlen keine weitere Praktik zu nutzen.
Hierbei ist das Wort Empfehlung von hoher Bedeutung, denn letztendlich muss dies nicht so verwendet werden. Ist ein Praktiker der Meinung, dass andere Praktiken oder andere Methoden verwendet werden, dann kann dies jederzeit gemacht werden. Die Software sieht vor, dass anhand der meist eingesetzten Methoden und Praktiken eine Grundstruktur vorgegeben wird und weitere Praktiken empfohlen werden.
Theoretisch ist das Konzept anschließend abgeschlossen, da Methoden und Praktiken ausgegeben werden können. Sonderlich groß ist die Evaluation der zu erstellenden Software nicht. Von daher werden im nachfolgenden Kapitel weitere Aspekte und die Grenzen 4.4 diskutiert.

4.4 Grenzen

Das vorliegende Konzept basiert lediglich auf der HELENA-Studie und ist relativ einfach gestrickt. Die Frage, die sich resultierend stellt, ist, ob es weitere Möglichkeiten gibt, um Praktiken für hybride Entwicklung automatisiert auszuwählen. An dieser Stelle werden das Konzept sowie dessen Grenzen anhand einer Evaluation von Probleme sowie Betrachtung möglicher Lösungsansätze genauer diskutiert.

Probleme
Das Paper, über die Frames von hybrider Entwicklung von Prenner et al. [29] im Teilkapitel 2.3 liefert einige Informationen zur Struktur und Organisation. Dort werden die Frames erklärt und die Abhängigkeiten zwischen den einzelnen Einheiten dargelegt. Fünf Frames haben einen Einfluss darauf, wie die Prozesse gestaltet werden. Natürlich kann es Vorstellungen geben, wie Anforderungen gesammelt werden, getestet wird oder aber auch geplant

werden soll. Doch steht eine Methode oder Praktik wie zum Beispiel die formale Spezifikation fest, hat das Einfluss auf unterschiedliche Frames, die bei der Planung definiert werden müssen. Anhand dessen zeigt sich, dass es viele Einflussfaktoren und auch Abhängigkeiten gibt. Das bedeutet, dass es viel zu koordinieren und beachten gibt, wenn ein Projekt initiiert werden soll. Dies impliziert auch, dass die Wahl der Methoden und Praktiken nicht einfach umzusetzen sind.
Ein weiteres Problem ist, dass es keine konkreten Kriterien für die Nutzung von den Methoden und Praktiken gibt. Es gibt lediglich eine Ansammlung von Herausforderungen, die sich auf die Lösung von gezielten Problemen beziehen. Natürlich kann die Requirements Engineering traditioneller gehalten werden, wenn es dafür eine Spezifikation gibt, dies ist allerdings keine Pflicht. Zudem sind viele andere Praktiken und Methoden, nicht so einfach zu konkretisieren, da klare Kriterien fehlen, aber mehr dazu unter den möglichen Lösungsansätzen.
Die HELENA-Studie gibt an, welche Kombination von Methoden- und Praktiken am häufigsten sind. Diese werden auch in dem Prototyp genutzt. Wie viele Praktiken in normalen Projekten genutzt werden, ist unklar. Es kann daher vorkommen, dass die Anzahl der Praktiken schon mit der HELENA-Studie abgedeckt ist. Dementsprechend müssten keine weiteren Praktiken ausgewählt werden.
Das bedeutet, es fehlen sowohl Kriterien für die einzelnen Methoden und Praktiken als auch für die Anzahl der Praktiken, die ausgewählt werden müssen.
Ein weiteres Problem ist die Subjektivität der Daten. Nur weil Wasserfall als Methode genutzt wird, muss es nicht als klassisches Wasserfallmodell betrachtet werden. Auch im vorliegenden Konzept ist die Wasserfall-Methode nicht als klassisches Modell angewendet worden. Manch ein Praktiker wird das Wasserfallmodell als rein klassisch sehen, manch einer variiert dies zusätzlich. Das bedeutet, nur weil eine gewisse Methode vorgegeben ist, müssen die dazugehörenden Praktiken nicht stimmen. Im klassischen Wasserfallmodell würde immer die Wahl auf das Schreiben einer Spezifikation fallen. Das aktuelle Konzept sieht das Wasserfallmodell, aber keine Spezifikation vor.
Demzufolge liegen bereits einige Probleme vor, obwohl es noch keine Evaluation gab.

Mögliche Lösungsansätze
Trotz der Probleme wird nach weiteren Möglichkeiten gesucht, das Konzept auszuweiten. Daher stellt sich die Frage nach möglichen Kriterien zur Einbindung der einzelnen Praktiken, die bisher noch nicht bekannt sind. Ein sehr plakativer Ansatz ist es, dass je eine Frage pro Praktik aufgestellt wird. Das hat dennoch keinen Mehrwert, da der Aufwand sehr hoch ist und

somit könnte letzten Endes jede Praktik auch selber ausgewählt werden. Mit der Annahme, es gäbe die Fragen, wurde der Versuch angegangen, dass die Anzahl der einzelnen Fragen verringert wird. Dazu wurden unterschiedliche Ansätze genutzt:

1. Eine Möglichkeit wäre es, die Praktiken zusammenzufassen. Das heißt, wenn Praktik X benutzt wird, dann wird zusätzlich immer Praktik Y genutzt.

2. Eine andere Möglichkeit wäre es, die Praktiken gegenseitig auszuschließen. Das bedeutet, wenn Praktik X benutzt wird, dann wird Praktik Y nicht genutzt.

3. Eine weitere Möglichkeit wäre, Schnittmengen in dem Fragenkatalog von je einer Frage pro Praktik zu finden und die Menge dadurch zu verringern.

Für den ersten Ansatz konnten keine Praktiken gefunden werden, die eine Praktik voraussetzen oder eine Praktik automatisch mitgenutzt werden muss. Als Erklärungsansatz gilt, wird Praktik Y nach Praktik X genutzt, werden beide Praktiken zusammengefasst.
Bei dem zweiten Ansatz konnten keine Ausschlüsse festgestellt werden. Keine Praktik schließt eine andere Praktik aus. Alle Praktiken können theoretisch miteinander kombiniert werden.
Konkrete Fragen zu einer Praktik zu stellen, stellt sich als kompliziert heraus. Im Prinzip sind alle Praktiken individuell zu bewerten. Schließlich müssen die Eigenheiten, vor allem die Vor- und Nachteile einer einzelnen Praktik abgewogen werden. Von daher war es nicht möglich, eine Schnittmenge unter den Praktiken zu finden.
Demzufolge gibt es keine klaren Kriterien, mit denen die Praktiken ausgewählt werden können. Ein Ansatz ist es, dass die einzelnen Herausforderungen mithilfe der Ziele aus Prenner et al. [31] im Kapitel 2.2 gelöst werden. Die einzelnen Herausforderungen zeigen, ob in einem gewissen Bereich agiler oder traditioneller gearbeitet werden soll. Beispielsweise könnte bei mehr Up-Front Requirements Engineering immer eine Spezifikation geschrieben werden. Nur weil mehr Requirements Engineering am Anfang betrieben werden soll, muss dies keine Folgerung sein. Zudem gestaltet sich das Mapping der einzelnen Praktiken auf die Herausforderungen sehr schwierig. Bei der individuellen Betrachtung stellt sich in dieser Arbeit heraus, dass die Praktiken immer wieder einzeln betrachtet werden und nicht mithilfe der vorhandenen Kriterien gelöst werden können. Daher ist es (noch) nicht möglich, diese Praktiken automatisiert oder anhand weniger Kriterien auszuwählen.
Um für folgende Arbeiten eine Basis zu bilden und das Wasserfallmodell zu konkretisieren, wird im nachfolgenden Kapitel eine Evaluation von den

erwähnten Herausforderungen erstellt. Diese haben klare Ziele definiert und können im besten Fall zukünftig bei der Auswahl unterstützen. Dadurch steht bereits die Basis für einen Prototyp, bei dem die Evaluation nur noch mit den Praktiken kombiniert werden müsste.

4.5 Kriterien von hybrider Entwicklung

Wie bereits erwähnt, gibt es Grenzen in der Auswahl von Praktiken. Allerdings gibt es die Möglichkeit, dass die Methode des Wasserfallmodells genauer charakterisiert wird. Die einzelnen Phasen des Wasserfallmodells decken sich mit den Herausforderungen von Prenner et al. [31], die bereits in Kapitel 2.2 erläutert wurden. Um zu evaluieren, welche Herausforderungen und Ziele betrachtet werden müssen, werden die einzelnen Herausforderungen genannt und erklärt, inwiefern diese Einflüsse auf die Auswahl der Methode haben.

- **Management von Anforderungsänderungen.** Handelt davon, wie mit Anforderungen umgegangen wird, sodass dies eine grundsätzliche Entscheidung ist und von keiner Methode abhängt. Daher hat dies keine bedeutende Relevanz.

- **Konflikt zwischen Up-front und continuous Requirements Engineering.** Ob Up-front oder continuous Requirements Engineering genutzt wird, ist eine Abwägung von Interessen, die Einfluss auf die Umsetzung des Wasserfallmodells nimmt. Dies ist die erste Phase in den drei betrachteten Wasserfallmodellen der hybriden Entwicklung. Dadurch nimmt diese Abwägung Einfluss auf die Methode.

- **Konflikt zwischen Up-front und continuous Architektur und Design.** Dies muss geklärt werden, aus den gleichen Gründen wie bei dem Requirements Engineering. Es ist die zweite Phase in den drei betrachteten Wasserfallmodellen der hybriden Entwicklung und daher wichtig zu klären.

- **Konflikt zwischen zentralen Entscheidungen und selbstorganisierten Teams.** Entweder übernimmt ein Chef die Verantwortung oder die Teams entscheiden dies selbständig, sodass es eine fest gegebene Struktur innerhalb eines Unternehmens und Projektes gibt. Daher hat dies keine Relevanz.

- **Schwierigkeiten einer separaten Testphase und Auslieferung.** Ist für die Methode wichtig zu entscheiden, da es die vierte bzw. fünfte Phase ist. Entweder werden die Tests integriert oder sie werden separat ausgeführt. Das Gleiche gilt für die Integration, die direkt an die Implementierung angebunden oder final am Ende vollzogen wird.

- **Spannungen zwischen der Geschäftsführung und der Entwicklungsabteilung.** Die müssen selbstverständlich geklärt werden, sind allerdings kein Bestandteil einer Methode oder Praktik. Wie die Geschäftsführer und die Entwicklungsabteilung ihre Konflikte lösen, ist ein internes Thema, das unabhängig von der Methode gelöst werden muss. Daher hat dies keine Relevanz.

- **Konflikt zwischen Lang- und Kurzzeitplanung.** Wie lang geplant wird und die einzelnen Zyklen laufen, das wird sich in den Projekten von selbst ergeben, dafür bedarf es keiner neuen Evaluation oder konkreten Methode. Das können die Unternehmen selbst entscheiden. Daher hat dies keine Relevanz.

- **Konflikt zwischen explizierter Dokumentation und explizitem Wissen.** Wie genau dokumentiert und das Wissen geteilt wird, das hat ein Unternehmen grundsätzlich zu klären. Zum Dokumentationsumfang ist keine zusätzliche Evaluation oder Methode notwendig, dieser wird von der Geschäftsführung vorgegeben. Die Dokumentation an sich kann eine Methode oder Praktik sein. Individuell wird aber entschieden, wie diese auszusehen hat. Somit beeinflusst es keine Auswahl von Methoden oder Praktiken. Daher ist dies nicht von Relevanz.

Anhand der Analyse lassen sich drei relevante Herausforderungen herausbilden. Es sind die Herausforderungen, die auch die Phasen des Wasserfallmodells abbilden. Wie bereits geklärt, haben die Herausforderungen mehrere Ziele, die Einfluss auf die Umsetzung haben.
Dabei muss beachtet werden, dass die einzelnen Ziele auf mehrere Herausforderungen Einfluss haben. Beispielsweise findet sich die hohe Komplexität des Produktes in der Herausforderung zur Architektur und Design wieder, wiederum aber auch in der Herausforderung zum separaten Testen bzw. Ausliefern. Dieses Ziel kann also der Up-Front Architektur bzw. Design und dem separaten Testen bzw. Ausliefern zugeordnet werden. Das bedeutet, die einzelnen Ziele reduzieren sich durch Überschneidungen, sodass sich schlussendlich in den drei Herausforderungen 19 unterschiedliche Ziele befinden, die es für die Umsetzung zu klären gilt. Dabei lassen sich einige Ziele bereits zur Evaluation ausschließen. Einige Ziele sind in jedem Projekt von Relevanz. Es ist beispielsweise so, dass Anforderungen geklärt werden müssen oder das Risikomanagement ist immer wichtig. Ein früher Wert ist immer gerne gesehen oder Error sollten zudem immer früh erkannt werden. Das hat zur Folge, dass diese Ziele immer priorisiert werden und einen hohen Stellenwert haben. Diese Analyse zieht sich durch ein paar weitere Ziele genauso durch, sodass die 19 Ziele auf neun Ziele reduziert sind.

- Verwaltung groß angelegter Entwicklung

- Verwaltung verteilter Entwicklung
- Analyse der komplexen Anforderungen
- Management von großen Stakeholdergruppen
- Erfüllung von Sicherheits- oder Vorschriftanforderungen
- Handeln mit unbekannten Anforderungen
- Hohe Komplexität des Produktes
- Nutzung von Frameworks
- Produkt kann nur als Ganzes getestet werden

Anhand dieser Ziele werden die Herausforderungen für die Methoden und Praktiken geklärt. Dafür wird eine geeignete Evaluationsmethode benötigt.
Hierfür wird eine fünfstufige Likert-Skala installiert, in der die Wichtigkeit von den einzelnen Zielen abgefragt wird. Das bedeutet, dass ein Ziel *wichtig*, *weniger wichtig*, *neutral*, *eher nicht wichtig* oder als *nicht wichtig* eingestuft wird. Anhand dessen liegt zu den abgefragten Ziele eine Bewertung vor. Wie diese Werte behandelt werden, fehlt allerdings noch.
Um die Evaluation zu klären, wird an einem konkreten Beispiel, ohne konkrete Ziele zu nennen, das Vorgehen erläutert:
Die Up-Front Architektur und das Design verfolgen vier Ziele, die bewertet werden müssen. Die kontinuierliche Architektur und das Design haben hingegen nur zwei Ziele. Jedes Element wird mit einem Gewicht versehen, das heißt, wichtige Elemente erhalten einen Punkt, weniger wichtige hingegen fünf Punkte. Anschließend werden der Durchschnitt von der Up-Front Architektur und dem Design sowie der kontinuierlichen Architektur und dem Design berechnet. Diese Durchschnittswerte der jeweiligen Architekturansätze werden miteinander verglichen, indem die Lösung mit der höchsten Priorität, also dem geringsten Wert, ausgewählt wird. Im Falle von Gleichstand wird der agile Ansatz bevorzugt, da dies der modernere Ansatz und deswegen die Präferenz des Autors ist.
Wichtig ist, dass die einzelnen Punkte auf der Likert-Skala relativ zueinander sind und es keine fest definierten Abstände zwischen den Bewertungen gibt. Trotzdem kann diese Skala zur Bewertung der einzelnen Kriterien verwendet werden, da es sich um Tendenzen handelt, anhand derer Entscheidungen getroffen werden können. Wenn der Nutzer jedoch sicher wäre und es keine Tendenzen gäbe, würde der Prototyp in diesen Fällen nicht verwendet werden.

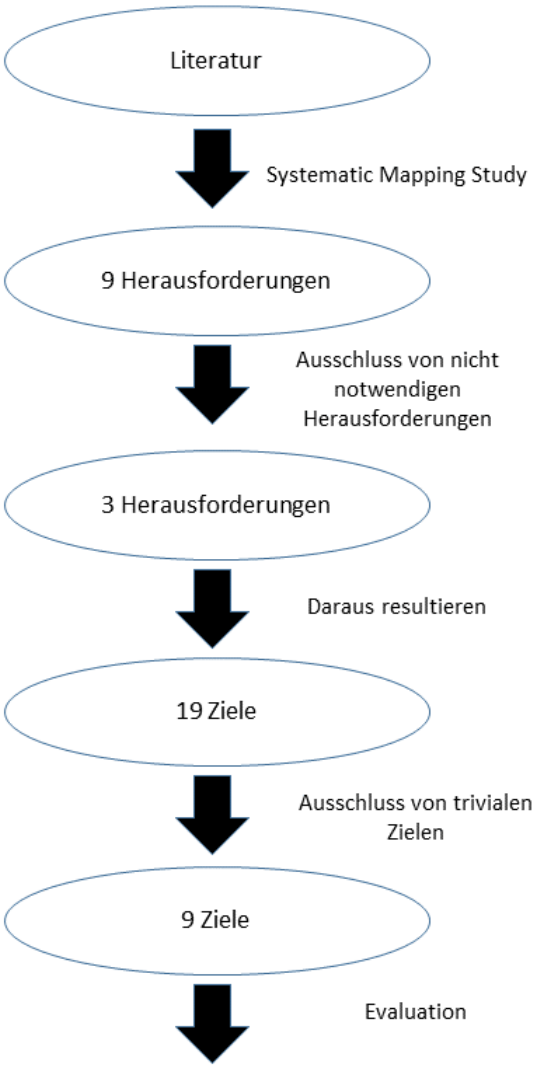

Abbildung 4.1: Der Weg von der Systematic Mapping Study über die Herausforderungen bis hin zur Evaluation.

Abbildung 4.1 visualisiert den zuvor dargestellten Prozess von der Konzepterstellung von den Herausforderungen aus Prenner et al. [31] hin zu den zu evaluierenden Zielen. Es werden nicht notwendige Herausforderungen herausgenommen und durch triviale Ziele extrahiert, sodass am Ende für die Methode neun Ziele evaluiert werden müssen. Diese geben dann Auskunft über die Umsetzung des Requirements Engineering, der Architektur bzw. dem Design und des Testens bzw. der Integration, sodass das Wasserfallmodell konkret umgesetzt werden kann. Als Ergebnis stehen dann der Software

folgende Umsetzungen des Wasserfalls Modells zur Verfügung:

- Up-Front Requirements Engineering oder kontinuierliches Requirements Engineering
- Up-Front Architektur bzw. Design oder kontinuierliche Architektur bzw. Design
- Separates Testen bzw. Auslieferung oder integriertes Testen bzw. Auslieferung

Anhand dieser Parameter lässt sich anschließend das Wasserfallmodell konkretisieren. Alle Möglichkeiten für die Umsetzung der Wasserfall-Methode sind in der beiliegenden Software enthalten. Zudem bildet die Beantwortung von den Herausforderungen eine Grundlage für weitere Arbeiten in diesem Themengebiet.

4.6 Zusammensetzung der Teilkonzepte

Aufgrund der vielfältigen Ansätze werden die einzelnen Konzepte in diesem Kapitel zusammengefügt. Am Wichtigsten ist die Grundlage, die sich in Teilkapitel 4.2 gebildet hat. Die Methoden Wasserfall, Scrum und iterative Development bilden die Grundlage für die Software. Anhand dessen werden die folgenden Praktiken ausgewählt, die sich in der HELENA-Studie bewährt haben. Das hat den Vorteil, dass zunächst einmal zwei der drei Basispraktiken (Code Review, Coding Standards und das Release Planning) genutzt werden. Werden die Methoden und Basispraktiken miteinander kombiniert, dann ergeben sich neun weitere Praktiken, die sehr häufig genutzt werden. Diese bilden zusätzlich die Grundlage für die Software, wobei der Prototyp nur eine Auswahl der weiteren Praktiken anhand der Agilität vorsieht.
Weitere Methoden haben sich nicht angeboten, da die HELENA-Studie maximal vier Methoden in Kombination vorsieht und die ausgewählten Methoden mit keiner vierten Methode kombiniert werden können. Die Auswahl von weiteren Praktiken gestaltet sich als schwierig und konnten aufgrund fehlender Kriterien nicht ausgewählt werden.
Um allerdings für weitere Arbeiten eine Vorarbeit zu leisten und die Umsetzung des Wasserfallmodells zu konkretisieren, werden drei Herausforderungen von hybrider Entwicklung aus Prenner et al. [31] genauer analysiert und bewertet. Das bedeutet, es wird mithilfe des Prototyps definiert, wie das Requirements Engineering, die Architektur bzw. das Design und das Testen bzw. die Integration auszusehen haben.
Das ganze Prinzip ist in der Abbildung 4.2 noch einmal visualisiert. Die Basismethoden werden mit weiteren Praktiken durch HELENA kombiniert. Zusätzlich gibt es eine Basis für weitere Forschungen, indem wichtige Herausforderungen für die Praktiken geklärt werden.

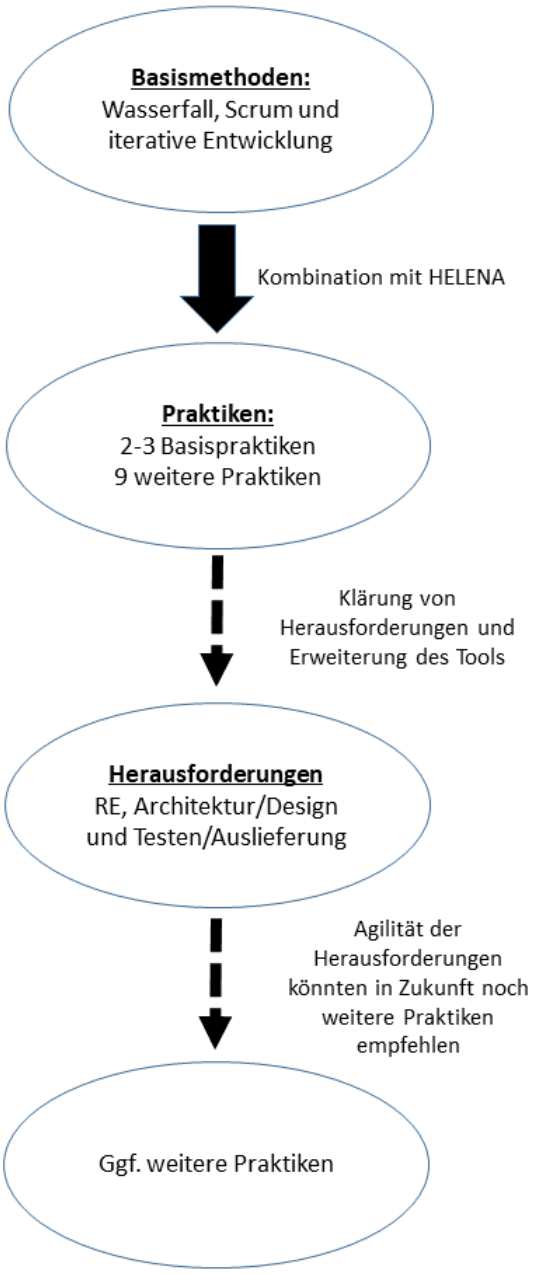

Abbildung 4.2: Finales Konzept für die Erstellung des Prototyps.

Kapitel 5

Implementierung

In diesem Kapitel wird auf die Implementierung des Prototyps eingegangen. Hierbei wird die Umsetzung sowohl technisch als auch strukturell erläutert, sowie die einzelnen Benutzeroberflächen kurz aufgezeigt und erklärt.

5.1 Umsetzung

Die technische Umsetzung ist eine Webanwendung, um den Nutzer eine angenehme Benutzeroberfläche bieten zu können, die beispielsweise für eine Nutzung auf einem Webserver laufen kann. Das bedeutet, dass die Nutzer keine weitere Software installieren müssen. Die Webanwendung läuft im Frontend mit HTML, CSS und Javascript. Das Backend ist mit dem Python Framework Flask angebunden. Für die Verwendung und Installation ist eine Anleitung im Anhang A.4 enthalten. Für die Umsetzung ist es wichtig, dass die Darstellung der einzelnen Projekte in einer kleinen Datenbank über SQLite3 gespeichert werden. Auf ein Datenbankschema wird verzichtet, da aufgrund der Einfachheit die einzelnen Prioritäten in Verbindung mit dem Namen und der ID als Schlüssel in einer Tabelle gespeichert werden können. Die errechneten Ergebnisse der Software werden nicht abgespeichert, sondern immer neu berechnet, weshalb es keine weiteren Tabellen gibt. Das heißt, dass lediglich die angekreuzten Ergebnisse gespeichert werden. Das bringt den Vorteil mit sich, dass bei Änderung der Logik keine Ergebnisse neu berechnet werden müssen, sondern die Ergebnisse direkt bei Änderung des Tools eingesehen werden können.

Die einzelnen grafischen Benutzeroberflächen sind in drei Teilbereiche zu unterteilen. Für die Umsetzung des Prototyps ist die Umfrage von Relevanz sowie die Auswertung wichtig. Zudem sollte es noch die Möglichkeit geben, Projekte zu erstellen bzw. auf diese zugreifen zu können. Von daher ist die Oberfläche aufzuteilen in die Befragung, Auswertung und die Verwaltung. Diese werden nachfolgend erklärt.

5.2 Grafische Benutzeroberfläche

Oberste Priorität der Benutzeroberfläche hat eine einfache Bedienung und eine gute Austauschbarkeit. Austauschbarkeit deswegen, weil bereits jetzt bekannt ist, dass dieser Prototyp zur weiteren Nutzung bearbeitet werden muss.

Befragung
Für die Befragung wurden neun Ziele im Konzept in Kapitel 4 aufgestellt. Diese sollen über eine Likert-Skala evaluiert werden. Die einzelnen Items werden anhand ihrer Priorität über Checkboxen eingestuft. Die Handhabung ist sehr einfach, da die Boxen leicht angeklickt werden können. Zudem ist die Austauschbarkeit der Fragen oder Ergänzung gewährleistet. Dieser Fragebogen wird nach Beendigung über eine POST Request an das Backend übergeben, damit die Daten in die Datenbank geschrieben werden und das Projekt ausgewertet werden kann.

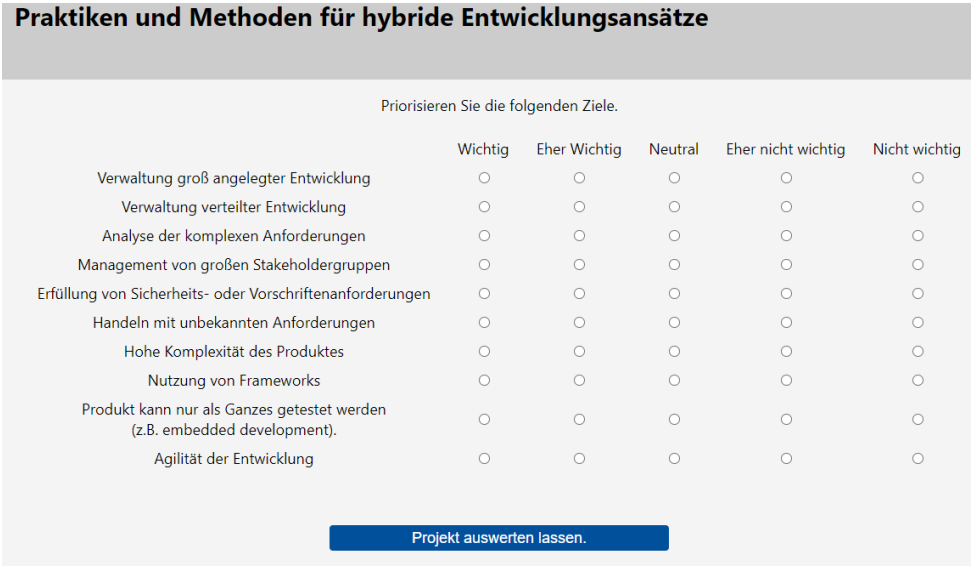

Abbildung 5.1: Einschätzung der Prioritäten in der Software.

Auswertung
Um für den Nutzer bestmöglich die Ergebnisse aufzubereiten, wird die Auswertung des Fragebogens grafisch und textuell dargestellt. Bei der grafischen Umsetzung handelt es sich um das Wasserfallmodell, welches je nach Umsetzung unterschiedlich dargestellt wird. Damit entsteht eine unterschiedliche Darstellung für die Umsetzung vom Requirements Engineering, der Architektur bzw. Design und der Auslieferung bzw. dem Testen. Wenn etwas integriert bzw. kontinuierlich geschehen soll, existiert eine zyklische

Darstellung. Bei der Up-Front Entscheidung ist hingegen keine zyklische Darstellung vorhanden.
Die einzelnen Herausforderungen werden mit einzelnen Zyklen oder ohne Zyklen aufgezeigt. Sind überall Zyklen eingezeichnet, also ist alles kontinuierlich, dann kann sogar von der Auslieferung wieder in die Requirements Engineering Phase gegangen werden, wie im Wasserfall-Iterativen-Ansatz aus Prenner et al. [30] aus dem Teilkapitel 2.1. Die Anzahl der möglichen Umsetzungen besteht aus in Summe 2^3 Möglichkeiten, da es für die zwei Möglichkeiten (Up-front oder kontinuierlich) drei Phasen gibt. Diese sind je nach Umsetzung grafisch in der Software enthalten. Neben der Grafik wird diese kurz textuell erläutert, um dem Nutzer mögliche Fragen zu erklären und die Verständlichkeit zu erhöhen.

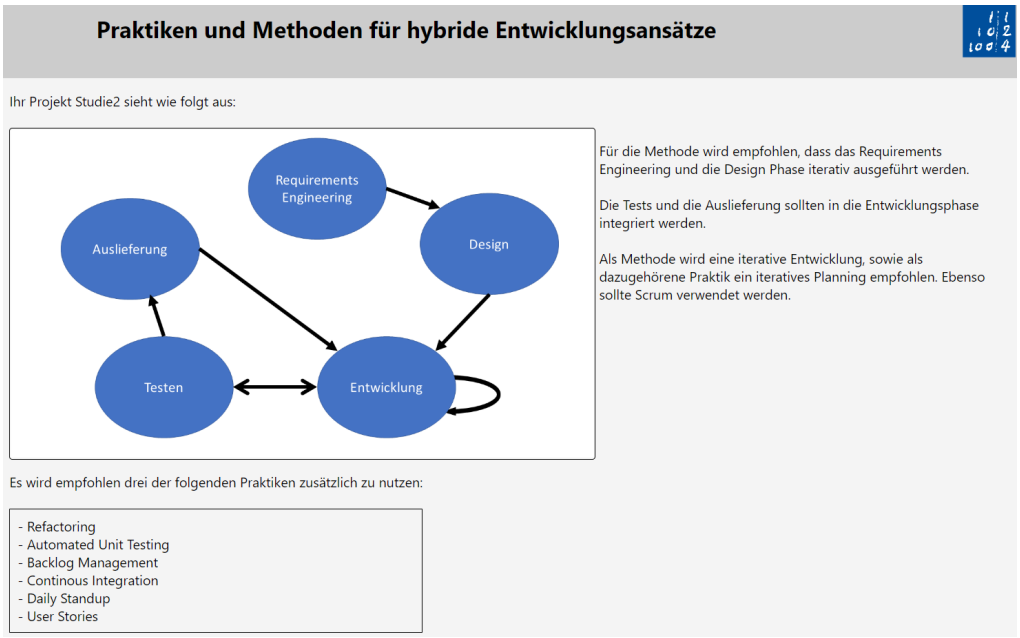

Abbildung 5.2: Ausgabe der Ergebnisse in dem Prototypen.

In der Abbildung 5.2 ist die Software mit dem Ergebnis abgebildet. Der Aufbau des Tools gestaltet sich im Aufbau durch drei unterschiedliche Teile.

1. Die Abbildung, um das grundlegende Wasserfallmodell zu visualisieren.
2. Die Erläuterung des Wasserfallmodells. Ebenfalls werden die dazugehörigen Methoden Scrum und iterative Development empfohlen. Des Weiteren wird die Praktik iteratives Planning empfohlen. Als optionaler Parameter wird je nach Integration die Praktik Continuous Integration empfohlen.
3. Eine weitere Auswahl an Praktiken, aus denen gewählt werden soll.

Die Abbildung mit der eingehenden Erklärung wurde bereits erklärt. Der restliche Teil besteht aus der Nennung, welche Methoden und Praktiken gewählt werden sollen. Zum einen wird iterative Development, Scrum und iteratives Planning empfohlen. Zum anderen wird ein Teil der sechs weiteren Praktiken empfohlen. Mithilfe der visuellen und textuellen Erklärungen sind für den Nutzer alle wesentlichen Bestandteile ersichtlich. Klar wird in den Beschreibungen auch, dass es sich lediglich um Empfehlungen handelt. Die Entscheidung, was in einem Projekt genutzt werden soll, obliegt den Nutzern immer selbst und wird nicht von dem Prototyp vorgeschrieben.

Verwaltung
Um die einzelnen Projekte zu verwalten und die Skalierbarkeit des Prototyps zu gewährleisten, müssen die Projekte angelegt werden und wieder aufrufbar sein. Von daher kann der Nutzer eigenständig entscheiden, ob er ein neues Projekt erstellen oder auf ein altes Projekt zurückgreifen möchte. Wird ein neues Projekt erstellt, dann ist es wichtig, einen Projektnamen auszuwählen, damit der Nutzer das Projekt wieder aufgreifen kann. Wird ein Projekt erstellt, dann wird zu dem bereits erklärten Fragebogen aus Abbildung 5.1 weitergeleitet. Greift der Nutzer auf ein altes Projekt zu, dann kann er dieses Projekt auswählen und wird zu der Evaluation aus der Abbildung 5.2 weitergeleitet.

Kapitel 6

Studiendesign

Um das vorliegende Tool und die dahinter stehende Logik evaluieren zu können, bedarf es einer qualitativen Untersuchung, die in diesem Kapitel erläutert wird.

6.1 Ziel

Um die Studie aufbauen zu können, wurde zunächst das Forschungsziel mithilfe des Goal-Definition-Templates von Wohlin et al. [45] formuliert:

Forschungsziel der Studie

Analysiere die vorgeschlagenen Methoden und Praktiken der Software
mit dem Ziel, diese auf Sinnhaftigkeit zu überprüfen,
in Bezug auf die Unterstützung von Software-Projekten
aus Sicht von erfahrenen Prozess-Ingenieuren
im Rahmen von Experteninterviews.

Um das Forschungsziel der Studie genauer zu definieren, werden Studienfragen aufgestellt, die darauf abzielen, die Sinnhaftigkeit des Tools und den einhergehenden Methoden und Praktiken zu evaluieren. Die Studienfragen beziehen sich auf die acht Axiome, nach denen die Software konstruiert wurde. Das bedeutet, dass zunächst die Methoden analysiert werden, dann die Basispraktiken und anschließend alle weiteren Praktiken. Von daher stellt sich zunächst die Frage, ob die drei gewählten Methoden, die in der Software immer verwendet werden, auch von den Probanden immer genutzt werden.

Studienfrage 1

Werden die gewählten drei Methoden (Wasserfall, iterative Development und Scrum) aus der Software häufig von den Studienteilnehmern genutzt?

Die HELENA-Studie liefert drei Basispraktiken, die immer zu mindestens 85% genutzt werden und die Basis der Praktiken in der Software liefern, daher stellt sich die Frage:

Studienfrage 2

Werden zwei der drei gewählten Basispraktiken (Code Reviews, Release Planning, Coding Standards) aus der HELENA-Studie häufig genutzt?

Ein Punkt ist, dass in der Software mit einer höheren Agilität mehr Praktiken empfohlen werden, daher stellt sich auch die Frage, ob dies berechtigt ist:

Studienfrage 3

Hängt die Agilität von der Anzahl der Praktiken ab?

Die Auswahl der weiteren Praktiken, die nicht den Basispraktiken entsprechen, basieren auf Tell et al. [42] und der damit verbundenen Häufigkeit in der HELENA-Studie. Daher stellt sich auch die Frage, inwiefern diese von den Studienteilnehmern berücksichtigt werden.

Studienfrage 4

Inwiefern decken sich die ausgewählten Praktiken mit denen des Tools bzw. der HELENA-Studie?

Als großer Schwachpunkt stellte sich bei der Bildung der Software heraus, dass sich keine klaren Kriterien für die Auswahl der Methoden und Praktiken finden lassen konnten, weshalb sich die Frage resultierend gestellt hat, ob sich innerhalb der Studie Kriterien herauskristallisieren. Dies wird allerdings nicht als Forschungsfrage definiert, da dies nicht den Hauptfokus der Studie darstellt. Die Evaluation der Software steht primär im Vordergrund.

6.2 Vorbereitung

Das Ziel der Experteninterviews ist es, dass die vorgeschlagenen Methoden und Praktiken auf dessen Sinnhaftigkeit ausgewertet und überprüft werden.

Zu diesem Zweck wird die Evaluation von Personen mit erweiterten Vorkenntnissen durchgeführt. Dies ist notwendig, da ausschließlich Grundkenntnisse für die Evaluation nicht ausreichend sind. Die Auswahl von Methoden und Praktiken soll systematisch erfolgen und nicht willkürlich entstehen. Deswegen ist ein tieferes Verständnis von Methoden und Praktiken in hybriden Entwicklungsansätzen notwendig. In der Bewertung der Methoden und Praktiken soll zudem eine Diskussion durchgeführt werden, um weitere Erkenntnisse zu gewinnen. Für eine Diskussion sind Experten-Interviews die optimale Möglichkeit.
Die Teilnehmerzahl wird in diesem Fall gering gehalten, denn die Studie liefert nicht den Hauptteil der Arbeit. Durch die Experteninterviews gibt es einen sehr hohen Aufwand, es ist schwierig, Personen zu akquirieren und bereits wenige Experten können eine Aussagekraft haben. Die Teilnehmerzahl beträgt fünf Probanden.
Für die Auswertung müssen Daten erhoben werden. Die demografischen Daten würden Rückschlüsse auf einzelne Personen erlauben. Daher sind die Daten pseudonymisiert. Erfasst werden lediglich Vorkenntnisse sowie ein Identifizierungscode zum Löschen der Studienergebnisse.
Die Transkribierung der Interviews wäre möglich gewesen, umfasst aber einen hohen Zeitaufwand und hat keinen großen Mehrwert, weshalb darauf verzichtet wurde.
Die Studie wurde grundsätzlich in Präsenz durchgeführt. Ein Proband konnte allerdings nur online teilnehmen, wobei dies keinen Einfluss auf die Ergebnisse der Studie nimmt.

Der grundlegende Gedanke der Studie ist, dass zunächst einmal eine Evaluation des Tools unabhängig von den vorgeschlagenen Methoden und Praktiken konzipiert wird. Das wird am besten erreicht, indem sich die Probanden für unterschiedliche Szenarien eigene Methoden und Praktiken überlegen. Für die Studie wurden drei Szenarien erstellt, die die unterschiedlichen Entwicklungsansätze darstellen sollen: agil, hybrid und traditionell. Somit wird ein Verständnis darüber entwickelt, wie die einzelnen Studienteilnehmer mit den Entwicklungsansätzen verfahren. Es kann natürlich auch herauskommen, dass die Studienteilnehmer bei allen Entwicklungen gleich verfahren, was ebenfalls eine Aussagekraft für den hybriden Ansatz hätte. Des Weiteren kann anhand der Anzahl der Methoden und Praktiken erfasst werden, inwiefern die Agilität bei der Anzahl der Methoden und Praktiken eine Rolle spielt. Für die Umsetzung der Ansätze wurden drei Szenarien erstellt, die drei realitätsnahe Konflikten enthalten. Der Hinweis, ob etwas agil, hybrid oder traditionell sein soll, wurde den Studienteilnehmern nicht gegeben, um sie nicht zu beeinflussen. Die Verfahren stellen sich hierbei wie folgt dar:

1. **Agil:** Es soll in einem kleinen Unternehmen für eine externe Firma,

eine Software für den Ladezustand von Autos geschrieben werden. Hierbei finden häufig Updates statt und auf Änderungen der Anforderungen soll jederzeit reagiert werden.

2. **Hybrid:** In einem mittelgroßen Softwareprojekt wird von der traditionellen Entwicklung auf die hybride umgestellt. Die Mitarbeiter sind allem offen, haben aber nicht die nötige Expertise, um dies durchzusetzen.

3. **Traditionell:** Eine große Firma für Reifen macht neue Reifen, die nicht platzen sollen. Es wird viel Struktur benötigt und die Reifen dürfen auf gar keinen Fall platzen.

Die einzelnen Szenarien sollen zunächst einmal mit Methoden und Praktiken gefüllt werden. Zur Verfügung werden den Studienteilnehmern die Szenarien sowie die Methoden und Praktiken der HELENA-Studie gestellt, da die jeweiligen Methoden und Praktiken auf dieser Datenmenge aufgebaut sind. Für die Probanden ist es nicht von Relevanz, ob sie alle Methoden und Praktiken kennen. Es sind alle Hilfsmittel erlaubt, die genutzt werden wollen, um diesen Auswahlprozess so realitätsnah wie möglich zu gestalten. Der gesamte Auswahlprozess geschieht anhand der Think-Aloud-Methode. Somit werden die Gründe für die Auswahl bereits geklärt bzw. Nachfragen vorweggenommen. Der Interviewer stellt während der Auswahl keine Fragen, es sei denn, es gibt Unklarheiten oder der Interviewer hat das Gefühl, dass in diesem Moment weitere Fragen zu einem Punkt gestellt werden müssen.
Im Optimalfall kommt es nach der Auswahl zu einer kleinen Diskussion, warum welche Methode oder Praktik jetzt ausgewählt wurde und warum welche nicht ausgewählt worden sind, um ein Verständnis zu der Auswahl zu erlangen. Zudem wird evaluiert, inwiefern das Wasserfallmodell Sinn ergibt und wie die Herausforderungen des Requirements Engineering, der Architektur und des Testens bzw. Ausliefern aus der Sicht des Interviewten bewältigt werden sollen.

Die weiteren Fragen, die in der Diskussion gestellt werden, sind individuell entstanden. Das bedeutet, dass diese auf jeden Probanden innerhalb des Interviews angepasst werden, je nachdem wie die Auswahl der Methoden und Praktiken verläuft. Denn je nach Auswahl und Vorgehensweise müssen die Fragen angepasst werden. Der Fokus liegt hierbei darauf, die Gründe für die Auswahl, sowie mögliche Vergleichsaspekte zu den Ergebnissen der Software herauszufinden.
Anhand dieser zwei Parameter soll evaluiert werden, warum die Software gut oder nicht gut funktioniert. Es kann die Möglichkeit bestehen, dass die Kriterien für die Methoden und Praktiken offensichtlich sind, der Autor diese aber nicht gefunden hat. Somit können Ursachen für die fehlende Funktionalität der Software identifiziert werden.

Der letzte Punkt im Studienaufbau ist, dass die Studie mit der Delphi-Methode abläuft. Das bedeutet, dass die Studienteilnehmer mit den vorherigen Ergebnissen der Probanden konfrontiert werden können [24]. Dies wird gemacht, um die Korrektheit von Aussagen einzelner Probanden zu überprüfen. Durchgeführt wird dies, wenn nötig, gegen Ende der Studie, um die Auswahl des aktuellen Probanden nicht zu beeinflussen.
Während der Studie werden Notizen von dem Interviewer gemacht. Dieser notiert zum einen die Methoden und Praktiken, die verwendet werden. Zum anderen werden von dem Interviewer noch weitere Notizen gemacht, die für die Auswertung relevant sein können. Es handelt sich also um Ergebnisprotokolle der Interviews, nicht um Verlaufsprotokolle. Alles zu notieren, würde die Interviews verlängern und die Konzentration des Interviewers und der Studienteilnehmer verringern. Die Möglichkeit eines Protokollanten bestand aufgrund des Umfanges einer Masterarbeit nicht.

6.3 Auswahl und Akquise der Teilnehmer

Die Teilnahme an der Studie ist nur beschränkt möglich. Es sind für die Studie nur Experten zugelassen, von daher wurden die einzelnen Probanden direkt angeschrieben oder angesprochen, indem ihnen die Studie und die Hintergründe erklärt wurden.
Als konkretes Kriterium für die Teilnahme an der Studie benötigen die Teilnehmer mindestens sehr gute Kenntnisse in der Vorlesung Software Process Engineering. Drei der fünf Probanden besitzen diese, der vierte leitet die Übung in diesem der eben genannten Vorlesung und der fünfte Proband hat seine Promotion in diesem Themengebiet geschrieben. Alle Studienteilnehmer haben an der Leibniz Universität Hannover ihr Studium absolviert oder absolvieren aktuell den Master.

6.4 Ablauf

Die Interviews haben im Schnitt ca. 45 Minuten gedauert. Dabei war der Ablauf wie folgt:

1. Aushändigen und Ausfüllen der Datenschutzerklärung (ca. 5min)
2. Erläuterung des Studienablaufs (ca. 3min)
3. Sammeln der Vorerfahrungen (ca. 1min)
4. Erfassung der gewählten Methoden und Praktiken (ca. 25min)
5. Rückfragen, Vergleich und Diskussion (ca. 10min)

Den Probanden wurden die Szenarien und Methoden und Praktiken aus der HELENA-Studie zur Verfügung gestellt. Nachdem die Materialien gesichtet und die Datenschutzerklärung unterschrieben wurde, konnte mit der eigentlichen Befragung begonnen werden. Der weitere Verlauf der Interviews ist aufgrund der Semistrukturiertheit sehr individuell. Durch die Think-Aloud-Methode haben sich häufig viele Fragen selbständig geklärt oder es konnten ggf. direkt Rückfragen gestellt werden, um die Gründe für die Wahl von einzelnen Methoden und Praktiken zu erlangen.
Wie bereits in der Vorbereitung erwähnt, wurde anschließend auf die mögliche Umsetzung eines Wasserfallmodells eingegangen. Zudem wurden noch einmal genauere Nachfragen zu manchen Methoden und Praktiken gestellt. Beispielsweise, wenn iterative Development genutzt wird, aber kein iterative Planning war dies ein Grund noch einmal genauer hinzuhören. Andere Nachfragen haben sich beispielsweise optionale Parameter bezogen, wie Pair Programming oder es kam vor, dass das Gefühl entstand, etwas wurde vergessen. Beispielsweise gab es auch Fälle, dass gesagt wurde, dass die Praktik immer verwendet werden solle. Im weiteren Verlauf wurde sie allerdings nicht erwähnt. Wenn innerhalb der Rückfragen Korrekturen gemacht wurden, dann wurde dies in der Auswahl der Methoden und Praktiken berücksichtigt.

6.5 Studienergebnisse

Die Studie wurde sehr individuell durchgeführt und jeder Studienteilnehmer hatte seine Spezialgebiete und Ansätze. Daher wird, um die Studienergebnisse zu erläutern, grob auf die einzelnen Ergebnisse der Probanden eingegangen.

Die Vorgehensweise der einzelnen Probanden war dabei gleich. Sie haben sich das Szenario durchgelesen und anhand dessen die einzelnen Methoden und Praktiken abgewogen und ausgewählt. Die jeweiligen Probanden sind chronologisch nach dem zeitlichen Verlauf sortiert. Bei den Ergebnissen der einzelnen Personen wird auf die Besonderheiten eingegangen. Die Studienergebnisse dienen nicht dazu, jede einzelne Methode und Praktik ins Detail zu analysieren. Bei der geringen Anzahl an Studienteilnehmern sind Abweichungen zweifellos. Diese Abweichungen spielen bei der Auswertung, durch die persönlichen Präferenzen und Zufälle bei dieser Studie eine große Rolle.
Bei der folgenden Betrachtung werden beginnend die einzelnen Studienteilnehmer analysiert. Dabei wird erst einmal auf die Vorerfahrung eingegangen, einzelne Ergebnisse sowie Tendenzen in den drei Entwicklungsansätzen vorgestellt. Anschließend wird der Umgang mit dem hybriden Entwicklungsansatz und den einhergehenden Herausforderungen

beschrieben. Falls es Besonderheiten gibt, wird als Letztes darauf eingegangen. Nach der Einzelanalyse werden die Ergebnisse als Ganzes analysiert und zusammengefasst.

Proband 1
Der Proband hat die Vorlesung Software Process Engineering erfolgreich besucht (halbes Jahr Vorerfahrung) und zudem weitere Erfahrungen in der Automobilbranche gesammelt, was bei den Szenarien von Relevanz ist.
In allen drei Szenarien wurde das iterative Development und Kanban verwendet. Des Weiteren nutzte der Proband elf Praktiken in allen drei Szenarien. Es wurden zwei oder drei Methoden in den Szenarien und zusätzlich zwischen 15 und 19 Praktiken verwendet. Der Teilnehmende würde zudem die Anforderungserhebungen und Architektur Up-Front durchführen und nur bei größeren Umsetzungen erneut die Anforderungserhebung durchführen oder die Architektur ändern. Zudem würde er das Testen und die Integration immer in die Entwicklung integrieren.

Proband 2
Der Proband hat die Vorlesung Software Process Engineering erfolgreich besucht (halbes Jahr Vorerfahrung).
Besonders auffällig ist die geringe Anzahl an Methoden und Praktiken. Es wurden lediglich ein oder zwei Methoden verwendet, sowie zwischen fünf und neun Praktiken. Es wurden immer Code Reviews und Coding Standards benutzt. Für die Person ist es wichtig, dass die Anforderungserhebungen und Architektur Up-Front durchgeführt werden, ggf. auch mehrfach. Ein weiterer Aspekt ist die Integration von Testen und Integration möglichst in die Entwicklungsphase.

Proband 3
Der Proband hat seine Promotion in diesem Themengebiet absolviert (mehrere Jahre Vorerfahrung). Durch die Arbeit und vielen Gesprächen mit Entwicklern in der freien Wirtschaft besitzt er ein sehr hohes Vorwissen.
Besonders auffällig ist, dass der Proband zwischen vier und sechs Methoden anwendete. Zudem wurden zwischen 18 und 21 Praktiken genutzt. Der Studienteilnehmer nutzte zudem die gleichen 15 Praktiken und vier Methoden in allen drei Szenarien. Für den Probanden ist es außerdem wichtig, dass immer zu einem gewissen Teil die Anforderungserhebung und Erstellung der Architektur sowie des Designs Up-Front geschieht. Ebenso wird nach seiner Einschätzung fast ausschließlich integriert ausgeliefert, da Unternehmen die Projekte in den Unternehmen immer weiter fortgeführt werden.
In der Diskussion äußerte der Proband Kritik an der HELENA-Studie.

Er kritisiert vor allem die fehlenden Abweichungen in den Methoden und Praktiken. Es gibt nur die Auswahl, ob etwas genutzt wird. Wie eine Methode oder Praktik durchgeführt wird und ob etwas davon abgewichen wird, ist nicht gegeben. Beispielsweise muss das Wasserfallmodell nicht klassisch genutzt werden. Die Nutzung von Scrum ist in jedem Unternehmen anders und es kann sehr unterschiedlich umgesetzt werden. Eine Spezifikation zu schreiben, ist häufig sehr hilfreich, allerdings muss diese nicht in tiefste Detailebenen erfolgen. All dies kann nicht durch die HELENA-Studie abgedeckt werden. Das Ausmaß und die Umsetzung der Methoden und Praktiken fehlt also völlig.

Proband 4
Der Proband hat die Vorlesung Software Process Engineering erfolgreich besucht (halbes Jahr Vorerfahrung).
Er hat in allen Szenarien vier Praktiken wiederverwendet. Im dritten Szenario hat er nur eine Methode genutzt, ansonsten drei bzw. vier Methoden. Zudem nutzte er sieben bis zehn Praktiken in den einzelnen Szenarien. Der Studienteilnehmer schätzt Up-Front Anforderungserhebungen für wichtig, zudem ist er der Meinung, dass es ein Grob- und Feinentwurf für die Architektur geben muss. Das Testen und Ausliefern sollte seines Erachtens separat erfolgen.

Proband 5
Der Proband hat die Übung für die Vorlesung Software Process Engineering geleitet (halbes Jahr Vorerfahrung). Zudem hat er durch das Labor *Intensivübung Agile Software-Entwicklung* weitere Kenntnisse in den agilen Methoden und Praktiken erworben. Als weitere Kenntnisse hat der Proband einen Vortrag über DevOps gesehen.
Auffällig ist, dass der Proband nie das iterative Development auswählte. Zudem wählte der Proband Coding Standards, Code Reviews und Retrospectives in allen drei Szenarien aus. Die Basispraktik Release Planning wurde hingegen kein einziges Mal ausgewählt.
Auffällig ist, dass der Proband Wasserfall im zweiten Szenario nutzen wollte, allerdings nicht im klassischen Sinn, sondern eine abgewandelte Variante, ähnlich dem Water-Scrum-Fall aus der Vorlesung Software Process Engineering.
Die Abwandlung der Methode deckt sich mit den Aussagen von Proband 3, der in den Methoden und Praktiken die Feingranularität vermisste.
Ein weiterer auffälliger Punkt ist die Auswahl der Methoden im dritten Szenario. Hier wählte der Proband sieben Methoden aus.
Der Befragte würde zudem die Anforderungserhebungen und Architektur Up-Front durchführen. Außerdem würde er immer versuchen, das Testen

und die Integration in die Entwicklung einzubinden.

Zusammenfassung der Ergebnisse
Die Probanden sind sich darin einig, dass die Anforderungen und die Architektur zunächst als Up-front umgesetzt werden müssen. Dies deckt sich auch mit den Aussagen von Prenner et al. [29], die immer zu einem Mindestmaß die Anforderungserhebung und die Architektur festlegen würden.
Uneinig sind sich die Studienteilnehmer bei der Auswahl der Methoden und Praktiken. Ebenso in der Anzahl der Methoden und Praktiken gibt es erhebliche Unterschiede. Allerdings wurden von allen Studienteilnehmern immer die Code Reviews und die Coding Standards genutzt, welche zwei der drei Basispraktiken aus der HELENA-Studie aus Tell et al. [42] bilden. Die dritte Basispraktik, das Release Planning wurde von zwei der Studienteilnehmer nie genutzt, außerdem benutzten zwei weitere Studienteilnehmer das Release Planning immer.
Einig sind sich die Studienteilnehmer darin, dass die Anzahl der Methoden und Praktiken keinen Einfluss auf die Agilität haben. Das spiegelt sich auch in ihrer Auswahl wider, da sie in den einzelnen Szenarien keine großen Abweichungen in der Anzahl der Methoden und Praktiken haben.
Am Wichtigsten für die Auswertung ist das zweite Szenario, da es sich hier um eine hybride Entwicklung handelt, weshalb diese nachfolgend genauer betrachtet wird.
Bei der Betrachtung der einzelnen Methoden und Praktiken lässt sich bereits anhand ihrer Anzahl erkennen, dass die Auswahl individuell ist. Die Teilnehmer wählten hier 3, 1, 6, 4 bzw. 3 Methoden aus. Bei der Anzahl der Praktiken sind es 15, 10, 21, 10 bzw. 13 Praktiken.
Um die Studienergebnisse genauer zu betrachten, werden noch einmal die Methoden und Praktiken in der Tabelle 6.1 zusammengefasst, die in mindestens der Hälfte der Durchläufe ausgewählt wurden, somit von mindestens drei der fünf Probanden.

6.6 Auswertung der Studienfragen

Bevor die einzelnen Ergebnisse diskutiert werden, wird auf die Studienfragen eingegangen.

Studienfrage 1
Im zweiten Szenario zeigt sich eine Tendenz Richtung Wasserfall, Scrum und iterative Development (je dreimal gewählt). Auffällig ist, dass Kanban immer am häufigsten genutzt wird. Fest steht, dass die gewählten Methoden des Tools häufig genutzt werden.

Bezeichnung	Häufigkeit	Methode/Praktik
Kanban	4	Methode
Iterative Development	3	Methode
Scrum	3	Methode
Wasserfall	3	Methode
Code Reviews	5	Praktik
Coding Standards	5	Praktik
User Stories	5	Praktik
Burn-Down Charts	4	Praktik
Continuous Deployment	4	Praktik
Daily Standup	4	Praktik
Iteration Planning	4	Praktik
Iteration / Sprint Reviews	4	Praktik
Retrospectives	4	Praktik
Automated Unit Testing	3	Praktik
Backlog Management	3	Praktik
Continuous Integration	3	Praktik
Formal Specification	3	Praktik
Release Planning	3	Praktik

Tabelle 6.1: Szenario 2: Alle Methoden und Praktiken, die mindestens dreimal ($\geq 60\%$) ausgewählt wurden

Studienfrage 2
Code Reviews und Coding Standards werden von jedem Probanden in allen Szenarien immer genutzt. In diesem Punkt sind sich alle Studienteilnehmer einig, mit der Begründung, dass sauberer Code wichtig ist. Von daher lässt sich die Frage mit ja beantworten. Dadurch ist davon auszugehen, dass zwei immer genutzt werden sollten.
Allerdings schneidet die dritte Basispraktik (Release Planning) deutlich schlechter ab. Im hybriden Szenario wird die Praktik immerhin von drei Personen Szenario gewählt. Fest steht, dass die Basispraktik weniger als in der HELENA-Studie genutzt wird.

Studienfrage 3
Es konnte keine Abhängigkeit zwischen Anzahl der Praktiken und der Agilität eines Projektes festgestellt werden.

Studienfrage 4
Um die Frage zu beantworten, wird die Menge der weiteren Praktiken mit denen aus dem zweiten Szenario verglichen.
Mindestens dreimal wurden die Praktiken Automated Unit Testing (3), Iteration Planning (4), Backlog Management (3), Continuous Integration

(3), Daily Standup (4) und User Stories (5) gewählt. Prototyping (1), Refactoring (1) und Design Reviews (0) wurden weniger gewählt.
Neben den Praktiken wurden noch Burn-Down Charts (4), Continuous Deployment (4), Retrospectives (3) und Formal Specification (3) häufig gewählt.
Den Daten wird entnommen, dass sechs Praktiken wiederzufinden sind, drei Praktiken findet sich eher nicht wieder und vier weitere Praktiken werden zusätzlich zu den Praktiken der HELENA-Studie häufig in der Studie ausgewählt.

Zusammenfassung und Interpretation der Studienergebnisse
Die Auswahl der Methoden und Praktiken stellte sich als sehr individuell heraus. Dennoch konnten Zusammenhänge festgestellt werden.
In der Methodenauswahl gibt es eine Tendenz in Richtung Iterative Development, Scrum und Wasserfall, die jeweils dreimal genutzt werden. Vor allem Kanban wurde sehr häufig genutzt. Dies spielte in dem Konzept für die Software keine Rolle. Ebenso nutzte vor allem der Proband mit dem meisten Vorwissen sechs Methoden, was in der Diskussion in 7 noch eine Rolle spielen wird.
Die Nutzung der Basispraktiken Code Reviews und Coding Standards hat sich sowohl in der Software als auch in der Studie als sinnvoll herausgestellt. Die Basispraktik Release Planning konnte sich nicht ganz so stark durchsetzen, wurde allerdings immerhin dreimal genutzt. Dies widerspricht allerdings auch nicht Tell et al. [42], da diese nur zwei der drei Basispraktiken als Nutzung ausgewiesen haben.
Die weiteren Praktiken, die häufig genutzt werden, bestehen zu einem großen Teil aus der HELENA-Studie, da sechs der neun zusätzlichen Praktiken auch von den Studienteilnehmern gewählt worden sind. Allerdings muss bedacht werden, dass diese auf der Auswahl von Wasserfall, Scrum und iterativer Entwicklung basieren. Auf Grundlage der Auswahl der Methoden müssten gemäß der HELENA-Studie andere Praktiken verwendet werden. Ein neuer Vergleich ist hier allerdings nicht möglich, da es keine Kombination aus Wasserfall, Scrum, iterativer Entwicklung und Kanban gibt.
Dass im Tool nur ein gewisser Anteil der weiteren Praktiken verwendet werden, lassen sich inhaltlich nicht in einen logischen Zusammenhang bringen. Es konnte kein Zusammenhang zwischen der Anzahl der Praktiken und der Agilität festgestellt werden. Zudem schätzten die Experten dies auch so ein. Warum dies auch nicht sein kann, wird in der Diskussion in Kapitel 7 erörtert.
Neben den bewährten Praktiken gibt es noch eine kleine Menge an weiteren Praktiken, die häufig genutzt werden. Es handelt sich hierbei um vier weitere Praktiken, die sich zusätzlich zu der Software in der Studie bewähren.

Insgesamt konnten Zusammenhänge zwischen Software und Studie identifiziert werden. Außerdem gibt es noch offene Fragen und Widersprüche, die weiter untersucht werden müssen.

Kapitel 7

Diskussion

In diesem Kapitel werden die Ergebnisse des Prototyps mit denen der Studie untereinander verglichen und ausführlich diskutiert. Deshalb werden zunächst einmal die Ergebnisse aus der Studie diskutiert, bevor die Forschungsfrage konkret beantwortet wird.

7.1 Diskussion der Studienergebnisse

Im Vorfeld der Studie gab es keine konkreten Vorstellungen, wie die Studienergebnisse aussehen könnten. Es wurden unabhängig Experten dazu befragt, wie sie unterschiedliche Entwicklungsansätze mit Methoden und Praktiken anhand von konkreten Szenarien umsetzen würden.
Trotzdem konnten anhand der Analyse und der Studienfragen einige Erkenntnisse über die Auswahl der Methoden und Praktiken gewonnen werden. Dabei konnten folgende Eindrücke des Prototyps bestätigt werden:

- Die Tendenz der Studienteilnehmer war es, Wasserfall, Scrum und iterative Development als Methoden auszuwählen. Diese wurden ebenfalls im Prototyp verwendet.
- Code Reviews und Coding Standards wurde von allen Studienteilnehmern immer verwendet. Dies sind in der HELENA-Studie zwei von drei Basispraktiken, die auch im Prototyp integriert sind.
- Sechs von neun Praktiken, die im Prototyp mithilfe der HELENA-Studie verwendet werden sollten, wurden von der Mehrheit der Studienteilnehmer eingesetzt.

Dem entgegen steht, dass es neben den Zustimmungen auch weitere Erkenntnisse gibt:

- Kanban war die meist präferierte Methode der Studienteilnehmer. Im Prototyp wurde Kanban nicht verwendet.

- Release Planning wurden als dritte Basispraktik von allen Studienteilnehmern nicht immer verwendet. Im hybriden Ansatz gab es allerdings eine Tendenz, sie zu nutzen.
- Drei von neun Praktiken, die im Prototyp mithilfe der HELENA-Studie verwendet werden sollten, wurden von der Mehrheit der Studienteilnehmer nicht eingesetzt.
- In dem Prototyp wurden mehr Praktiken bei mehr Agilität verwendet. Dieser Ansatz war falsch.

Anhand dieser Ergebnisse ist es möglich, dass der Prototyp evaluiert werden kann. Denn der Ansatz war es, dass eine Basis von Methoden genommen wurde und mithilfe der HELENA-Studie diese Methoden mit Praktiken angereichert werden.
Die Auswahl der Methoden hat im Vergleich zur Studie gut funktioniert. Es stellte sich heraus, dass die häufigst eingesetzten Methoden in der Studie Wasserfall, Scrum und iterative Development waren. Dennoch wurde Kanban noch häufiger eingesetzt. Dies deckt sich nicht ganz mit der Software.
Bei der Auswahl der Praktiken gibt es eine klare Deckung mit den Code Reviews und den Coding Standards. Diese Praktiken haben sich sowohl in dem Prototyp als auch in der Studie klar durchgesetzt. Die dritte Basispraktik (Release Planning) aus dem Prototyp ist kein Favorit in der Studie, wird aber häufig eingesetzt. Bei den weiteren Praktiken gibt es durchaus einen Unterschied zwischen Prototyp und Studie. Sechs von neun Praktiken des Prototyps überdecken sich mit der Mehrheit der Studienteilnehmer. Die Studienteilnehmer haben noch vier weitere Praktiken, die von der Mehrheit verwendet werden. Dadurch gibt es durchaus eine Tendenz in Richtung der Software bei der Auswahl der Praktiken mit kleinen Abweichungen.

7.2 Beantwortung der Forschungsfrage

Anhand der Erstellung des Konzeptes und des Prototyps sowie der einhergehenden Evaluation mithilfe der Studie wird die Forschungsfrage aus Kapitel 1 beantwortet. Um diese zu beantworten, wird diese zunächst einmal aufgegriffen.

Forschungsfrage

Inwiefern ist es möglich, auf Grundlage der Organisation Methoden und Praktiken für hybride Entwicklungsansätze zu finden?

Mithilfe des Konzeptes wurden Methoden und Praktiken aufgestellt, die ein Prototyp vorschlagen soll. Dafür wurden drei Methoden verwendet, die immer in der hybriden Entwicklung genutzt werden. Anschließend wurden

diese mithilfe der HELENA-Studie kombiniert. Durch die anschließenden Experteninterviews konnten diese dann evaluiert werden.
Wie bereits im vorherigen Teilkapitel erwähnt, konnte zwischen dem Prototyp und der Studie eine Deckung erzielt werden.
Die Wahl der Methoden zeigt, dass Wasserfall, Scrum und iterative Development eine Basis für hybride Entwicklung sein können. Dennoch wurde im Konzept Kanban nicht berücksichtigt, weshalb dort eine Abweichung festzustellen ist.
Die Wahl der Praktiken in der Studie decken sich überwiegend mit denen des Prototyps. Es stellt sich heraus, dass zwei Basispraktiken sich komplett durchsetzen, die dritte Basispraktik wird bei der Mehrheit eingesetzt und weitere zwei Drittel des Prototyps werden auch in der Studie häufig verwendet. Lediglich drei von neun Praktiken der Software wurden in der Studie nicht berücksichtigt, stattdessen allerdings vier andere Praktiken.
Durch diese Deckung lassen sich durchaus Tendenzen zeigen, dass geeignete Methoden und Praktiken für hybride Entwicklungsansätze gefunden werden können. Zwar ist in dieser Arbeit nicht gelungen, dass anhand konkreter Kriterien Methoden und Praktiken entwickelt werden können. Dennoch konnte herausgefunden werden, dass Methoden und Praktiken sehr häufig verwendet werden. Demnach gibt es für die Konstruktion zwei Möglichkeiten:

- Es werden die Methoden und Praktiken des Prototyps verwendet. Das sind die Methoden Wasserfall, Scrum und iterative Development. Zudem würden die Praktiken Code Reviews, Coding Standards, Release Planning, Prototyping, Refactoring, Automated Unit Testing, Iteration Planning, Backlog Management, Continuous Integration, Daily Standup, User Stories und Design Reviews) verwendet werden.

- Es wird die Schnittmenge der Methoden und Praktiken des Prototyps und der Studie verwendet. Das wären die Methoden Wasserfall, Scrum und iterative Development. Zudem würden die Praktiken Code Reviews, Coding Standards, Release Planning, Automated Unit Testing, Iteration Planning, Backlog Management, Continuous Integration, Daily Standup, User Stories) verwendet werden.

Lediglich die Nutzung der Methoden und Praktiken aus der Studie erscheint nicht sinnvoll, da die Validität nicht ausreichend hoch ist. Bei der Erstellung des Projekts mit einem hybriden Entwicklungsansatz liegt es nahe, dass die Basis von Methoden und Praktik mithilfe eines der zwei Ansätze gebildet wird. Wie dann weiter vorgegangen wird, konnte in der Arbeit nicht festgestellt werden. Die Kernaussage dieser Untersuchungen ist es, dass in der Konstruktion von hybriden Entwicklungsansätzen auf ähnliche Methoden und Praktiken zurückgegriffen wird. Das bedeutet zwar, dass es keine Vereinfachung mithilfe eines Prototyps gibt, allerdings kann mithilfe einer Basis ein hybrider Entwicklungsansatz balanciert werden und somit

auch unerfahrenen Praktikern geholfen werden.

7.3 Diskussion des Prototyps

Bereits vor der Erstellung des Prototyps gab es einige Unklarheiten, die mithilfe der Studie geklärt worden sind. Es gab keine klaren Kriterien für einzelne Methoden und Praktiken. Die Konstruktion der Methoden und Praktiken beruhte nur auf Basismethoden und der Kombination von den häufigsten Praktiken aus der HELENA-Studie. In der Studie konnte dieser Eindruck bestätigt werden. Dies geschah dadurch, dass die Studienteilnehmer die Verwendung der Methoden und Praktiken individuell bewerteten und es eine Deckung mit den ausgewählten Methoden und Praktiken zwischen Prototyp und Studie festgestellt werden.
Demzufolge ist das einfache Konzept kein Fehler, sondern wurde mithilfe der Studie bestätigt. Zudem wurde der Prototyp mit einer weiteren Funktionalität erweitert. Es könnten in der Zukunft mithilfe der Herausforderungen und Ziele aus Prenner et al. [31] Methoden und Praktiken empfohlen werden, die mit der Klärung der Herausforderungen einhergehen. Jedoch, mit den Ergebnissen der Arbeit hat der Prototyp aktuell keine Möglichkeit in der Praxis Verwendung zu finden.

7.4 Validität

In diesem Kapitel wird auf die Aussagekraft der Studie und der Ergebnisse dieser Arbeit eingegangen. Um die Validität zu überprüfen, werden die vier Typen der Validität von Wohlin et al. [45] verwendet. Das sind die Conclusion Valitidy, Internal Validity, Construct Validity und die External Validity.

Conclusion Valitidy
Die Anzahl der Studienteilnehmer ist äußerst gering, denn es wurden lediglich fünf Personen interviewt. Daher ist die Aussagekraft der Studie stark eingeschränkt. Um eine höhere Conclusion Validity zu erreichen, würde eine höhere Anzahl von Studienteilnehmern benötigt werden. Demzufolge handelt es sich in diesem Fall nur um eine vorläufige Evaluation.

Internal Valitidy
Für den hybriden Entwicklungsansatz gibt es nur ein Szenario. Mit mehreren hybriden Szenarien hätte festgestellt werden können, ob in diesen die Herangehensweise, Methoden und Praktiken gleich sind. Zudem sind die Szenarien eigenständig gewählt. Es ist nicht auszuschließen, dass unbekannte Störfaktoren einen Einfluss auf die Auswahl der Methoden und Praktiken haben. Beispielsweise gab ein Studienteilnehmer an, dass er Automobilkenntnisse habe und mit diesen Vorkenntnissen gewisse Praktiken auswählte.

Die Vorkenntnisse von Autos wurden bei der Erstellung des Szenarios nicht bedacht. Ein anderer Studienteilnehmer gab an, dass er manche Methoden und Praktiken aus persönlichen Präferenzen nicht nutzen würde, obwohl diese vielleicht besser geeignet wären.

Construct Valitidy
Die Methoden und Praktiken wurden abgehakt, wenn diese genutzt werden sollen. Optionale Methoden und Praktiken wurden bei der Auswertung nicht betrachtet. Wenn diese betrachtet werden, dann könnten die Messergebnisse anders ausfallen und die damit einhergehende Bewertung. Dazu gehört auch, dass die Wahl sowie Evaluation der Methoden und Praktiken keine Abwandlungen zulässt. Dies ist auch eine Schwachstelle in der HELENA-Studie. Es ist unklar, inwiefern eine Methode genutzt und diese umgesetzt wird. Die benannten Fälle waren allerdings Einzelfälle und würden von daher keine großen Änderungen bedeuten. Des Weiteren kannten die Studienteilnehmer nicht alle Methoden und Praktiken. Das bedeutet, dass die Möglichkeit besteht, dass andere Methoden und Praktiken ausgewählt würden, wenn diese den Studienteilnehmern bekannt wären.

External Valitidy
Die Studienteilnehmer kommen alle aus der Leibniz Universität Hannover. Dementsprechend weisen alle ähnliche Vorkenntnisse auf. Durch ähnliche Vorlesungen, Dozenten und den einhergehenden Inhalten ist eine ähnliche Auswahl von Methoden und Praktiken zu erwarten. Bei der Betrachtung der Spezifikation fällt auf, dass die in der Studie mehr Beachtung findet als in der HELENA-Studie. Die Spezifikation wird in einer Grundlagenvorlesung gelehrt und zudem in einem großen Projekt und einer weiterführenden Vorlesung behandelt, weshalb dies ein gutes Beispiel für den Einfluss von ähnlichem Vorwissen sein könnte. Fraglich ist allerdings, ob die Vorkenntnisse mithilfe der Vorlesung Software Process Engineering ausreichend sind, um hybride Entwicklungsansätze zu konstruieren. Denn die Herangehensweise des Doktoranden in dem Themengebiet war deutlich anders als die der anderen Studienteilnehmer. Zudem sind die Studienteilnehmer nicht in der freien Wirtschaft tätig, also fehlt ihnen die Erfahrung in der aktuellen Entwicklung.
Aufgrund der eben genannten Einflüsse ist es schwierig, die Studienergebnisse auf andere Projekte anzuwenden. Sicherlich können die Methoden und Praktiken für Projekte innerhalb der Leibniz Universität Hannover gut verwendet werden. Doch ein Praxisbezug für die Wirtschaft, aber auch für andere Universitäten fehlt. Deswegen sollte die vorläufige Evaluation nicht nur auf eine größere Stichprobe ausgeweitet werden, sondern auch weitere Experten in der Wirtschaft und an den Hochschulen befragt werden, um die

Ergebnisse auch für andere Projekte zu validieren.

Kapitel 8

Zusammenfassung und Ausblick

In diesem abschließenden Kapitel werden die bedeutendsten Ergebnisse dieser Arbeit vorgestellt.
Des Weiteren wird ein Ausblick auf zukünftige Forschungsfragen gegeben, die dazu beitragen sollen, die Forschung in dem Gebiet der hybriden Entwicklung voranzutreiben.

8.1 Zusammenfassung

In dieser Arbeit wurde untersucht, inwiefern hybride Entwicklungsansätze mit Methoden und Praktiken automatisch gefüllt werden können. Dabei wurde zunächst einmal die Literatur zu diesem Thema erfasst und mithilfe dessen Hilfe ein Konzept und Prototyp entwickelt, der das Problem lösen soll. Das Konzept basiert einerseits auf den festgestellten Basismethoden Wasserfall, Scrum und iterative Development sowie andererseits mit den häufigst kombinierten Praktiken aus der HELENA-Studie, die sich bei Kuhrmann et al. [19] zeigten.
Bereits vor der Evaluation des Prototyps gab es einige offene Fragen. Diese bestätigten sich bei der Auswertung der Studie mithilfe von Experteninterviews.
Es wurde festgestellt, dass bei der Konstruktion von hybriden Entwicklungsansätzen ähnliche Methoden und Praktiken verwendet werden. Die Studie zeigt, dass die Methoden und Praktiken von den Prototypen und der Studie eine große Schnittmenge haben. Die Validität der Studie ist eingeschränkt, da sie auf einen sehr eingeschränkten kleinen Personenkreis angewandt wurde, stimmt aber mit den Vorüberlegungen und dem Konzept überein.
Als Ergebnis steht ein Prototyp, der so konstruiert wurde, dass die Methoden Wasserfall, iterative Development und Scrum empfohlen werden. Als Praktiken werden Code Reviews, Release Planning, Coding Standards, Pro-

totyping, Refactoring, Automated Unit Testing, Iteration Planning, Backlog Management, Continuous Integration, Daily Standup, User Stories und Design Review vorgeschlagen. Da dies nicht sonderlich hilfreich ist, ist eine Anpassbarkeit bereits gegeben. Beispielsweise sind drei Herausforderungen und neun einhergehende Ziele aus Prenner et al. [31] an den Prototypen gebunden. Kann anhand dieser auf Methoden oder Praktiken geschlossen werden, könnten diese mit den Prototypen verknüpft werden. Allerdings bietet der aktuelle Prototyp keinen Mehrwert und hat in der aktuellen Implementierung keine Verwendung.
Offen ist, wie weitere Praktiken ausgewählt werden können und ob mit den Methoden und Praktiken ein kompletter hybrider Entwicklungsansatz gebildet werden kann. Dennoch ist eine klare Tendenz zu erkennen, dass die Auswahl der Methoden und Praktiken bei hybriden Entwicklungsansätzen sehr ähnlich sind.

8.2 Ausblick

In dieser Arbeit wurden einige Grenzen festgestellt. Zwar ist es möglich, dass hybride Entwicklungsansätze einfacher konstruiert und somit balanciert werden. Trotzdem gibt es noch einige offene Fragen, die geklärt werden können.
Der Prototypen beruht darauf, dass drei Methoden und zwölf weitere Praktiken genutzt werden. Unklar ist, ob diese ausreichen. Aufgrund der HELENA-Studie gab es nicht die Möglichkeit, mehr als drei Methoden zu kombinieren. Des Weiteren ist die Anzahl der Praktiken äußerst unklar. Auch in der Studie konnte sich keine Tendenz abzeichnen, wie viele Praktiken genutzt werden.
Bei Prenner et al. [30] wird die Organisation von hybriden Entwicklungsansätzen angedeutet. Es ist eindeutig, welche Phasen es in einem Projekt geben sollte. Allerdings ist unklar, wie die einzelnen Phasen und Ansätze genau miteinander kombiniert werden. Ebenso ist auch nicht bewusst, in welchen Phasen welche Methoden oder Praktiken genutzt werden. Scrum kann in jeder Phase verwendet werden, doch tut sie das auch?
Eine große Schwachstelle in dieser Arbeit ist, dass keine konkreten Kriterien für einzelne Methoden und Praktiken gefunden werden konnten. Daher empfiehlt es sich, dass die einzelnen Methoden und Praktiken einer Studie unterzogen werden sollten, mit dem Hinblick auf Kriterien in der Auswahl. Eine Möglichkeit wäre es auch, dass die Herausforderungen und Ziele aus Prenner et al. [31] auf Methoden und Praktiken gemappt werden.
Im Optimalfall ist es in Zukunft möglich, dass mithilfe von Natural Language Processing Anforderungen ausgelesen werden und anhand dessen ein Projekt mit den Methoden und Praktiken konstruiert wird. Dies ist allerdings eine Idealvorstellung.

Auf diesem Weg sollte auch noch einmal geprüft werden, inwiefern neue Methoden und Praktiken konzipiert werden müssen. Dies erwähnten Prenner et. al [29] auch bereits. Neue Methoden und Praktiken könnten die hybride Entwicklung noch einmal weiter voranbringen.

Anhang A

Appendix

A.1 Kombinationen in der HELENA-Studie

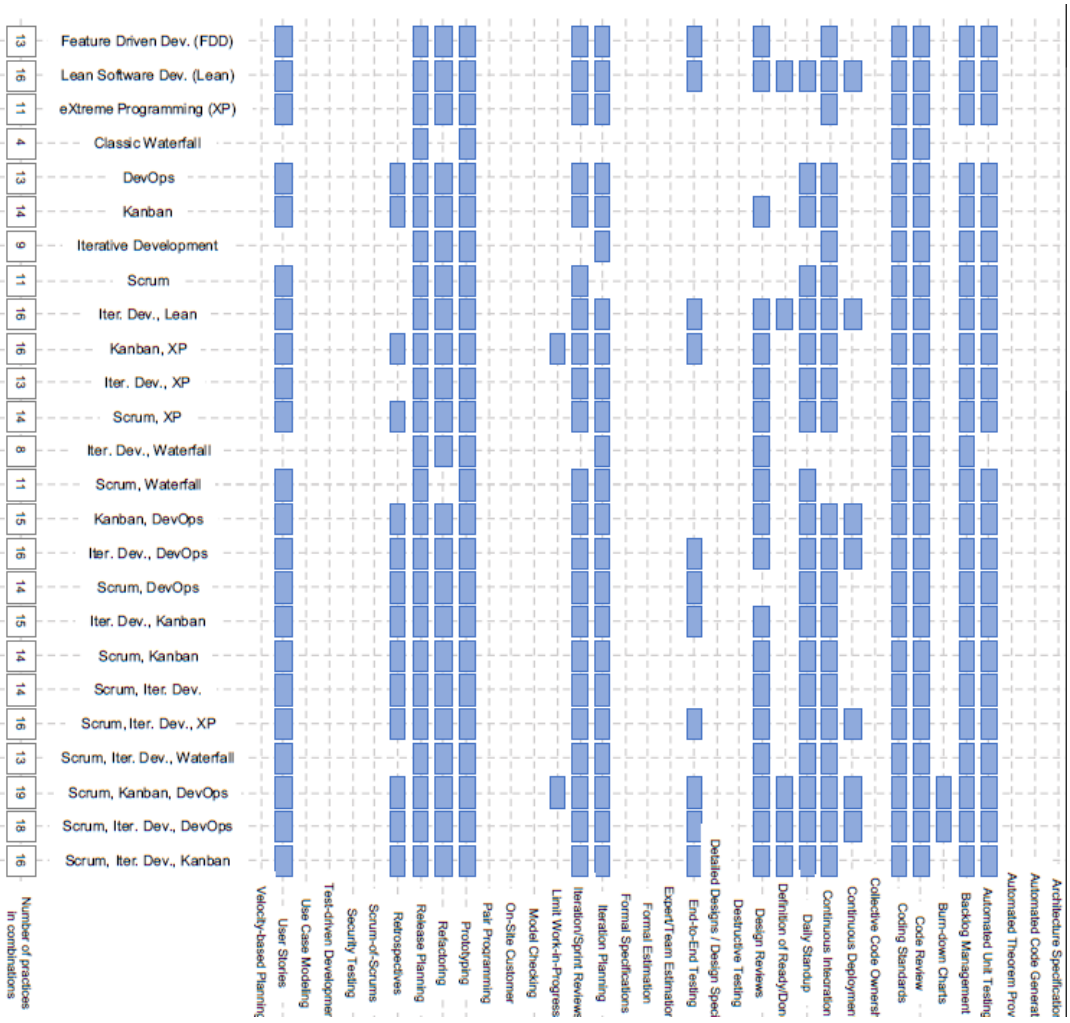

Abbildung A.1: Die Kombinationen der Praktiken, die zu mindestens 85% genutzt wurden, wo angegeben wurde, dass sie Entwicklungsansätze nicht vermischen aus Tell et al. [42]

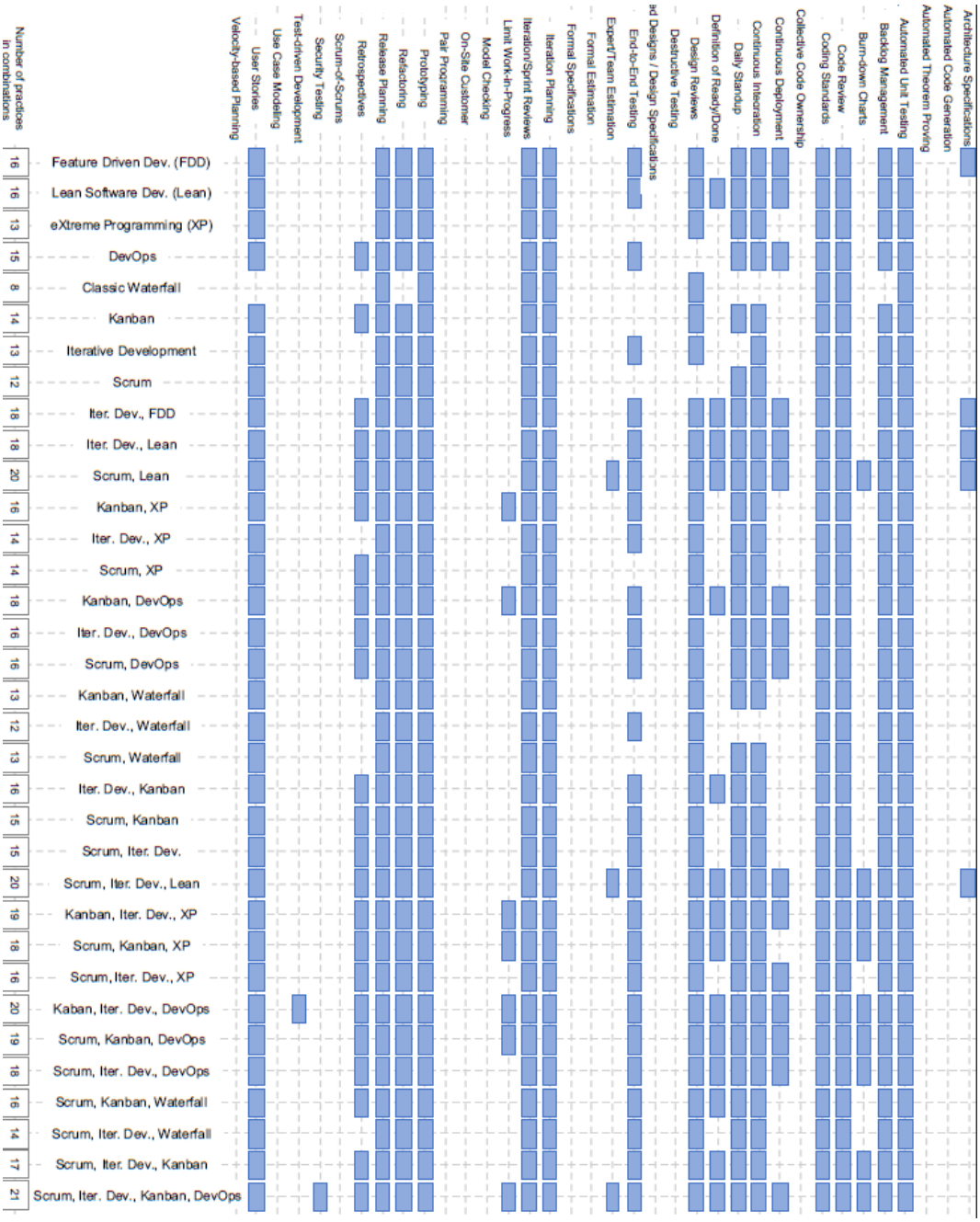

Abbildung A.2: Die Kombinationen der Praktiken, die zu mindestens 85% genutzt wurden, wo angegeben wurde, dass sie Entwicklungsansätze vermischen aus Tell et al. [42]

A.2 Agilitätsgrad

		Project Management	Quality Management	Risk Management	Configuration Management	Change Management	Requirements Analysis/ Engineering	Architecture and Design	Implementation and Coding	Integration and Testing	Transition and Operation	Maintenance and Evolution
Methods	*Waterfall*	0.00E+00	8.00E-13	1.42E-08	1.92E-05	5.71E-13	1.44E-13	2.87E-13	3.24E-16	1.74E-15	3.92E-08	1.85E-12
	Crystal											
	DevOps	1.67E-08	4.71E-06	5.90E-05	5.65E-10	2.51E-07	1.44E-04	4.61E-08	8.75E-08	2.51E-08	3.57E-10	1.06E-04
	Domain-Driven Design							2.23E-02	3.53E-02			3.62E-02
	DSDM											
	Extreme Programming	5.40E-05	1.66E-05	1.19E-03	4.83E-05	2.98E-05	5.76E-03	1.08E-02	1.93E-06	8.68E-10	1.26E-08	2.74E-05
	Feature-driven Development	3.45E-02										
	Iterative Development	4.34E-03	2.66E-02	3.37E-02	1.69E-03	3.18E-04			1.84E-05	2.62E-04	1.96E-02	8.71E-03
	Kanban	2.49E-02	4.66E-02	1.19E-02			4.55E-02		9.74E-03	1.03E-02	5.17E-04	5.95E-03
	Large-scale Scrumm (LeSS)		3.41E-02							9.82E-03	4.00E-02	
	Lean Software Development	1.36E-02		3.84E-02	6.87E-03	1.27E-02	1.30E-02	9.56E-05	1.03E-03	1.59E-04	8.48E-05	5.19E-03
	Model-Driven Architecture (MDA)								3.10E-02			
	Nexus											
	Personal Software Process (PSP)											
	Phase / Stage-gate model	9.78E-04	9.85E-04	9.85E-03						2.53E-02	3.81E-02	9.19E-03
	PRINCE2											
	Rational Unified Process (RUP)											
	Scaled Agile Framework (SAFe)				9.91E-03							
	Scrum	3.10E-10	4.90E-05	5.23E-05	4.73E-08	3.73E-08	2.85E-08	4.80E-07	2.97E-14	8.03E-12	3.52E-07	6.36E-07
	ScrumBan	5.13E-03	4.01E-02	4.51E-02	1.08E-03		9.20E-04	3.10E-02	4.32E-02	1.43E-03		
	Spiral Model							1.76E-02	1.11E-02			
	SSADM	4.69E-02	2.69E-02				2.01E-03	4.10E-03	2.48E-03	2.71E-03	2.68E-02	
	Team Software Process (TSP)						1.77E-02		1.43E-02			
	V-Shaped Process (V-Model)	5.50E-05	6.03E-09	1.28E-05	1.44E-04	3.58E-04	1.66E-05	1.01E-05	1.22E-04	4.54E-03	4.00E-04	1.36E-05
Practices	Architecture Specifications	1.09E-02	3.24E-02	3.28E-02	2.22E-02	7.75E-03	2.30E-02	6.83E-04			1.03E-02	3.61E-02
	Automated Code Generation											
	Automated Theorem Proving								1.75E-02			
	Automated Unit Testing	4.09E-04	2.87E-03		3.12E-03			2.74E-02	2.83E-06	2.18E-06	7.31E-03	1.10E-02
	Backlog Management	7.13E-10	3.28E-06	7.84E-05	2.00E-04	1.87E-07	4.52E-07	5.43E-09	8.26E-12	8.91E-09	1.00E-05	1.41E-07
	Burn-Down Charts								1.59E-03			
	Code Reviews	1.47E-03	2.15E-02				7.17E-03	8.74E-03	6.06E-04	4.31E-04		8.13E-03
	Coding Standards			2.69E-02			8.38E-03					4.16E-02
	Collective Code Ownership	8.35E-05	2.24E-04	2.07E-03	1.26E-06	9.39E-08	5.29E-04	5.44E-06	6.58E-07	1.15E-04	2.41E-03	5.65E-05
	Continuous Deployment	2.75E-04	5.35E-05	1.32E-04	6.23E-06	1.10E-05	6.31E-05	1.71E-03	9.09E-05	5.34E-09	2.24E-09	1.66E-04
	Continuous Integration	5.36E-06	6.97E-06	1.77E-04	2.47E-05	8.70E-06	1.52E-07	1.63E-06	1.05E-07	6.26E-11	2.20E-05	6.86E-06
	Daily Standup	8.02E-07	7.42E-04	2.72E-02	2.13E-03	9.96E-06	1.86E-06	3.48E-05	6.49E-11	2.41E-09	4.68E-05	3.37E-04
	Def. of Ready/Done	1.17E-05	2.01E-04	2.17E-04	7.57E-06	1.27E-06	1.22E-05	1.28E-04	3.67E-09	2.00E-07	4.38E-07	2.50E-06
	Design Reviews			3.59E-03					3.14E-02			
	Destructive Testing								1.16E-02			
	Detailed Designs	6.22E-03	3.80E-04	2.06E-03	5.71E-03	4.30E-03	3.78E-04	1.67E-07	1.23E-07	2.07E-05	4.26E-02	1.17E-03
	Limit Work-in-Progress	1.59E-02	1.17E-02				1.66E-02	5.11E-03	1.08E-02	3.38E-03	1.45E-03	6.92E-04
	End-to-End (System) Testing											
	Expert/Team based estimation	1.77E-02	2.12E-03	6.53E-03	3.96E-05	2.80E-03	2.02E-03	5.41E-04	1.66E-05	2.81E-03	1.18E-03	3.34E-02
	Formal estimation		1.49E-02			2.44E-02	3.88E-03	4.49E-05	2.70E-04	8.89E-03	1.44E-02	
	Formal Specification	8.68E-07	9.36E-07	4.34E-05	2.57E-06	7.12E-08	8.41E-09	1.40E-09	4.42E-07	7.47E-07	6.91E-05	1.69E-08
	Iteration Planning	1.27E-02					4.53E-02	3.30E-02	6.19E-05	1.08E-02	3.78E-02	
	Iteration / Sprint Reviews	4.31E-09	2.06E-04	1.85E-04	4.82E-04	4.49E-05	4.95E-08	8.32E-07	1.13E-10	3.16E-07	8.82E-06	3.80E-06
	Model Checking					2.86E-02	4.39E-02					
	On-Site Customer	3.59E-02					1.15E-02		2.63E-02			
	Pair Programming	1.42E-02			2.95E-03	1.51E-02	2.11E-02	5.65E-04	1.01E-02		4.01E-03	
	Prototyping					1.19E-02	8.35E-03					
	Refactoring	9.36E-05	2.12E-05	5.91E-03	1.97E-06	8.01E-06	7.56E-06	1.30E-07	1.55E-09	1.87E-12	5.19E-05	1.80E-04
	Release planning											
	Retrospectives	3.54E-06	2.36E-04	3.05E-02	2.08E-04	2.11E-06	1.19E-06	2.92E-05	3.92E-12	1.37E-09	4.89E-04	4.59E-05
	Scrum of Scrums			1.33E-02	4.79E-02					5.60E-03	1.27E-02	
	Security Testing											
	Test-driven Development	1.06E-02	4.56E-03	5.93E-03	5.94E-04	1.64E-02	3.77E-02	4.66E-04	2.03E-02	5.11E-06	7.49E-07	2.62E-03
	User Stories	5.17E-04	4.39E-04	1.04E-03	3.21E-03	1.03E-03	4.15E-07	2.07E-04	2.71E-05	2.43E-05	2.35E-02	1.89E-03
	Velocity-based Planning	1.16E-04	6.35E-06	1.26E-05	1.63E-05	8.37E-04	1.05E-06	4.03E-06	7.84E-10	3.14E-06	2.99E-04	1.56E-03
	Use Case Modeling	1.91E-02					1.71E-02	4.09E-03				

Abbildung A.3: Agilitätsgrade der Methoden und Praktiken in den einzelnen Projektdisziplinen aus Kuhrmann et al. [19]

	Project Management	Quality Management	Risk Management	Configuration Management	Change Management	Requirements Analysis/ Engineering	Architecture and Design	Implementation and Coding	Integration and Testing	Transition and Operation	Maintenance and Evoution
Waterfall											
Crystal											
DevOps						ag					ag
Domain-Driven Design							ag	n			ag
DSDM											
Extreme Programming			ag			ag	ag				
Feature-driven Development	ag										
Iterative Development	ag	n	n	n	ag				ag	n	ag
Kanban	ag	n	n			ag		n	n	n	ag
Large-scale Scrumm (LeSS)		ag							n	n	
Lean Software Development	ag		n	ag	ag	ag	ag	n	n	ag	ag
Model-Driven Architecture (MDA)								n			
Nexus											
Personal Software Process (PSP)											
Phase / Stage-gate model	n	tr	tr						tr	tr	n
PRINCE2											
Rational Unified Process (RUP)											
Scaled Agile Framework (SAFe)				ag							
Scrum											
ScrumBan	ag	ag	ag	ag		ag	ag	n	n		
Spiral Model							n	tr			
SSADM	tr	tr				tr	tr	tr	tr	tr	
Team Software Process (TSP)						n		tr			
V-Shaped Process (V-Model)				tr	tr			tr	tr	tr	
Architecture Specifications	tr	n	tr	n	tr	tr	tr			n	tr
Automated Code Generation											
Automated Theorem Proving								tr			
Automated Unit Testing	ag	n		n			n			n	ag
Backlog Management			ag	n							
Burn-Down Charts								n			
Code Reviews	ag	n				ag	n	n	ag		ag
Coding Standards			n			ag					ag
Collective Code Ownership	ag	n	n			ag			n	n	
Continuous Deployment	ag		ag				ag	n			ag
Continuous Integration			ag								
Daily Standup		n	ag	n							ag
Def. of Ready/Done		n	n				ag				
Design Reviews			tr					n			
Destructive Testing								tr			
Detailed Designs	tr	n	tr	tr	tr	tr				n	tr
Limit Work-in-Progress	ag	n				ag	ag	n	n	n	ag
End-to-End (System) Testing											
Expert/Team based estimation	ag	n	ag		ag	ag	n		n	n	ag
Formal estimation		n			tr	n		n	tr	n	
Formal Specification											
Iteration Planning	ag					ag	n		ag	n	
Iteration / Sprint Reviews		n	ag	n							
Model Checking					tr	n					
On-Site Customer	ag					ag		n			
Pair Programming	ag			ag	ag	ag	ag	n		ag	
Prototyping					n	n					
Refactoring	ag		ag								ag
Release planning											
Retrospectives		n	ag	n						n	
Scrum of Scrums			n	n					n	n	
Security Testing											
Test-driven Development	ag	ag	n	ag	ag	ag	ag	n			ag
User Stories	ag	n	ag	n	ag		ag			n	ag
Velocity-based Planning	ag				ag					n	ag
Use Case Modeling	n					n	n				

- fully traditional
- mainly tradtional
- balanced
- fully agile
- mainly agile

Abbildung A.4: Die Methoden und Praktiken in den einzelnen Projektdisziplinen aus Kuhrmann et al. [19], in denen ein Zusammenhang zwischen Agilität und Nutzung festgestellt wurde.

A.3 Studienergebnisse

Bezeichnung	Häufigkeit	Methode/Praktik
Kanban	4	Methode
Scrum	3	Methode
Code Reviews	5	Praktik
Coding Standards	5	Praktik
Test-driven Development	5	Praktik
Continuous Deployment	4	Praktik
Continuous Integration	4	Praktik
Iteration Planning	4	Praktik
Iteration / Sprint Reviews	4	Praktik
Retrospectives	4	Praktik
Daily Standup	3	Praktik
On-Site Customer	3	Praktik
Release Planning	3	Praktik
User Stories	3	Praktik

Tabelle A.1: Szenario 1: Alle Methoden und Praktiken, die mindestens drei Mal ($\geq 60\%$) ausgewählt wurden

Bezeichnung	Häufigkeit	Methode/Praktik
Code Reviews	5	Praktik
Coding Standards	5	Praktik
Formal Specification	4	Praktik
Continuous Deployment	3	Praktik
Continuous Integration	3	Praktik
Retrospectives	3	Praktik

Tabelle A.2: Szenario 3: Alle Methoden und Praktiken, die mindestens drei Mal ($\geq 60\%$) ausgewählt wurden

A.4 Prototyp Installation

Der Prototyp ist eine Webanwendung, welchem mit dem Python Framework Flask umgesetzt wurde. Die Architektur ist nach dem vorgeschlagenen Schema von Flask[1] umgesetzt. Nachfolgend befindet sich eine Anleitung, um das Projekt lokal und auf einem Server zu initialisieren.

[1] https://flask.palletsprojects.com/en/2.2.x/tutorial/layout/

Lokal

1. Navigation auf der CD in den Ordner Prototyp\website.
2. Aktivierung der virtuellen Umgebung. (Windows: .\.venv\Scripts\activate)
3. Installation der Abhängigkeiten dependencies:
   pip install -r requirements.txt
4. Erstellung einer .env Datei, mit folgendem Inhalt:
   FLASK_APP=“project“
   FLASK_ENV=“TRUE“
5. Erstellung der Datenbank (kann auch genutzt werden, um alle Daten zu löschen): flask init-db
6. Lokalen Server starten: flask run

Server
Hierfür wird WSGI und ein Apache-Server genutzt.

1. Apache auf Server installieren.
2. WSGI auf Server installieren.
3. Konfigurationsdatei im Apache-Ordner \server-available ändern, damit WSGI auf der richtigen IP läuft.
4. Server neu starten.

Falls etwas nicht funktioniert, sollte in die Log Files von Apache geschaut werden.

A.5 Prototyp Benutzeroberfläche

Hier sind die Übersichtsseiten abgebildet:

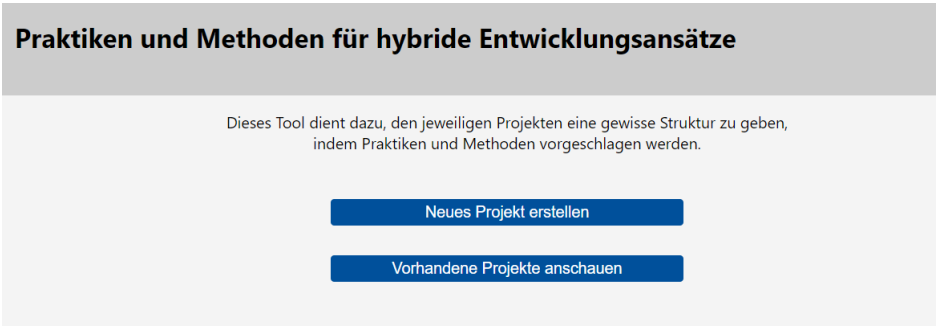

Abbildung A.5: Startseite und Navigationsseite für den Prototyp.

Abbildung A.6: Übersichtsseite für die Verwaltung der vorhandenen Projekte.

A.6 Studie

Sehr geehrter Herr Mustermann,

ich arbeite aktuell an meiner Masterarbeit zum Thema „Unterstützung der Balance in hybriden Entwicklungsansätzen mit geeigneten Methoden und Praktiken“. Dabei geht es darum Projekte mit geeigneten Methoden und Praktiken zu unterstützen. Dafür habe ich ein Tool entwickelt, das anhand von unterschiedlichen Kriterien konkrete Methoden und Praktiken empfiehlt.

Um mein Tool evaluieren zu können, benötige ich die Expertise von Ihnen. Ich würde Sie gerne bitten, sich für die folgenden drei Projektbeschreibungen Methoden und Praktiken mit dem Hinblick auf hybrider Entwicklung zu überlegen, die aus ihrer Sicht geeignet sind. Lassen Sie sich dabei so viel Zeit, wie sie benötigen. Als Hilfe gebe ich Ihnen zusätzlich die Methoden und Praktiken aus der HELENA Studie, aus denen Sie wählen sollen.
Im Anschluss daran würde ich gerne die Ergebnisse meines Tools mit ihrer Auswahl vergleichen und diskutieren. Gerne gebe ich Ihnen im Anschluss weitere Informationen über meine Arbeit und das dazugehörige Tool.

Bevor Sie allerdings loslegen, denken Sie bitte daran die Einverständniserklärung abzugeben. Ich danke Ihnen bereits im Voraus für die Teilnahme.

Mit freundlichen Grüßen
Fabian Natusch

Abbildung A.7: Anschreiben an die Studienteilnehmer.

Einverständniserklärung zur Mitwirkung an der Studie zum Thema: „Unterstützung der Balance in hybriden Entwicklungsansätzen mit geeigneten Methoden und Praktiken“

Die Studie wird von Fabian Natusch durchgeführt im Rahmen seiner Masterarbeit an der Leibniz Universität Hannover.

Die Teilnahme an der Studie ist freiwillig. Die Studienteilnahme wird nicht vergütet. Es besteht jederzeit die Möglichkeit die Studie ohne Angabe von Gründen abzubrechen.

Die Einwilligung ist freiwillig und kann jederzeit ohne Angabe von Gründen verweigert werden. Hierbei sind keine Nachteile zu befürchten.

Datenerfassung-, speicherung und -verwendung

Die Daten werden im Rahmen der Masterarbeit und dessen Archivierung gespeichert. Die erfassten Daten werden ausschließlich für wissenschaftliche Zwecke genutzt und pseudonomysiert ausgewertet. Zudem sind die Daten nur für das Fachgebiet Software Engineering der Leibniz Universität Hannover zugänglich. Die Daten werden im Rahmen der Masterarbeit pseudonomysiert veröffentlicht, können aber auch für wissenschaftliche Publikationen des Fachgebiets Software Engineering genutzt werden.

Es besteht jederzeit das Recht auf Auskunft, Berichtigung oder Löschung der perzonenbezogenen Daten. Die Einwilligungserklärung kann jederzeit schriftlich wiederrufen werden. Nach dem Widerruf werden die personenbezogenen Daten gelöscht und ab diesem Zeitpunkt für keine weitere Publikation verwendet.

Hiermit erkläre ich meine Zustimmung für die Einverständniserklärung.

_____

Name, Vorname selbstständig gewählte 4-stellige-Zahl

_____

Ort, Datum, Unterschrift

Abbildung A.8: Die Einverständniserklärung der Studie zum Verarbeiten der Daten.

Methoden

Waterfall
Crystal
DevOps
Domain-Driven Design
DSDM
Extreme Programming
Feature-driven Development
Iterative Development
Kanban
Large-scale Scrum (LeSS)
Lean Software Development
Mode-Driven Architecture (MDA)
Nexus
Personal Software Process(PSP)
Phase / Stage-gate model
PRINCE2
Rational Unified Process (RUP)
Scaled Agile Framework (SAFe)
Scrum
ScrumBan
Spiral Model
ssadm
Team Sofware Process (TSP)
V-Shaped Process (V-Modell)

Praktiken

Architecture Specifications
Automated Code Generation
Automated Theorem Proving
Automated Unit Testing
Backlog Management
Burn-Down Charts
Code Reviews
Coding Standards
Collective Code Ownership
Continuous Deployment
Continuous Integration
Daily Standup
Def. Of Ready/Done
Design Reviews
Destructive Testing
Detailed Designs
Limit Work-in-Progress
End-to-End (System) Testing
Expert/Team based estimation
Formal estimation
Formal Specification
Iteration Planning
Iteration / Sprint Reviews
Model Checking
On-Site Customer
Pair Programming
Prototyping
Refactoring
Release Planning
Retrospectives
Scrum of Scrums
Security Testing
Test-driven Development
User Stories
Velocity-based Planning
Use Case Modeling

Abbildung A.9: Die Methoden und Praktiken der Studie, die den Studienteilnehmern vorgelegt wurden.

A.7 CD

Die beigefügte CD im Anhang enthält folgende Dateien:

1. Die Arbeit als PDF.
2. Die Vorbereitungen (als PDF) und Auswertungen der Studie (im .xlsx-Format).
3. Der entwickelte Prototyp.

Literaturverzeichnis

[1] 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)*, 2(3):126–130, 2014.

[2] B. Aymerich, I. Díaz-Oreiro, J. C. Guzmán, G. López, and D. Garbanzo. Software development practices in costa rica: A survey. In *Advances in Artificial Intelligence, Software and Systems Engineering: Joint Proceedings of the AHFE 2018 International Conference on Human Factors in Artificial Intelligence and Social Computing, Software and Systems Engineering, The Human Side of Service Engineering and Human Factors in Energy, July 21–25, 2018, Loews Sapphire Falls Resort at Universal Studios, Orlando, Florida, USA 9*, pages 122–132. Springer, 2019.

[3] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. Manifesto for agile software development. 2001.

[4] S. Bick, K. Spohrer, R. Hoda, A. Scheerer, and A. Heinzl. Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. *IEEE Transactions on Software Engineering*, 44(10):932–950, 2017.

[5] 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*, 119:182–191, 2014.

[6] 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*, 4(1), 2013.

[7] P. Bourque and R. E. Fairley. *SWEBOK: guide to the software engineering body of knowledge*. IEEE Computer Society, 2014.

[8] L. Cao, K. Mohan, P. Xu, and B. Ramesh. 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. Proceedings of the*, pages 10–pp. IEEE, 2004.

[9] J. Cho. A hybrid software development method for large-scale projects: rational unified process with scrum. *Issues in Information Systems*, 10(2):340–348, 2009.

[10] M. Felderer, D. Winkler, and S. Biffl. Hybrid software and system development in practice: initial results from austria. In *Product-Focused Software Process Improvement: 18th International Conference, PROFES 2017, Innsbruck, Austria, November 29–December 1, 2017, Proceedings 18*, pages 435–442. Springer, 2017.

[11] M. Fowler, J. Highsmith, et al. The agile manifesto. *20th Australasian Conference on Information Systems Compatibility of Agile and Document-Driven Approaches*, 9(8):28–35, 2001.

[12] L. T. Heeager and P. A. Nielsen. Agile software development and its compatibility with a document-driven approach? a case study. *20th Australasian Conference on Information System Compatibility of Agile and Document-Driven Approaches, Melbourne.*, 2009.

[13] L. T. Heeager and P. A. Nielsen. *Meshing agile and plan-driven development in safety-critical software: a case study*, volume 25. Springer, 2020.

[14] 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*, 22(6):2892–2936, 2017.

[15] M. Hron and N. Obwegeser. Scrum in practice: an overview of scrum adaptations. *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[16] J. Klünder, P. Hohl, M. Fazal-Baqaie, S. Krusche, S. Küpper, O. Linssen, and C. R. Prause. Helena study: Reasons for combining agile and traditional software development approaches in german companies. In *Product-Focused Software Process Improvement: 18th International Conference, PROFES 2017, Innsbruck, Austria, November 29–December 1, 2017, Proceedings 18*, pages 428–434. Springer, 2017.

[17] J. Klünder, D. Karajic, P. Tell, O. Karras, C. Münkel, J. Münch, S. 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*, pages 61–70, 10 2020.

[18] 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. ICSSP 2017, page 30–39, New York, NY, USA, 2017. Association for Computing Machinery.

[19] M. Kuhrmann, P. Tell, R. Hebig, J. Klünder, J. Münch, O. Linssen, D. Pfahl, M. Felderer, C. R. Prause, S. G. MacDonell, J. Nakatumba-Nabende, D. Raffo, S. Beecham, E. Tüzün, G. López, N. Paez, D. Fontdevila, S. A. Licorish, S. Küpper, G. Ruhe, E. Knauss, Özcan Top, P. Clarke, F. McCaffery, M. Genero, A. Vizcaino, M. Piattini, M. Kalinowski, T. Conte, R. Prikladnicki, S. Krusche, A. Coşkunçay, E. Scott, F. Calefato, S. Pimonova, R.-H. Pfeiffer, U. P. Schultz, R. Heldal, M. Fazal-Baqaie, C. Anslow, M. Nayebi, K. Schneider, S. Sauer, D. Winkler, S. Biffl, M. C. Bastarrica, and I. Richardson. What makes agile software development agile? *IEEE Transactions on Software Engineering*, 48(9):3523–3539, 2022.

[20] 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 (ICTKE)*, pages 1–6, 2016.

[21] G. Lozo and S. Jovanovic. *A flexible hybrid method for IT project management*, volume 3. Citeseer, 2012.

[22] W. Millard, D. Johnson, J. Henderson, N. Lombardo, R. Bass, and J. Smith. Embedding agile practices within a plan-driven hierarchical project life cycle. *INCOSE International Symposium*, 24, July 2014.

[23] J. Nakatumba-Nabende, B. Kanagwa, R. Hebig, R. Heldal, and E. Knauss. Hybrid software and systems development in practice: perspectives from sweden and uganda. In *Product-Focused Software Process Improvement: 18th International Conference, PROFES 2017, Innsbruck, Austria, November 29–December 1, 2017, Proceedings 18*, pages 413–419. Springer, 2017.

[24] M. Niederberger and O. Renn. *Delphi-Verfahren in den Sozial-und Gesundheitswissenschaften: Konzept, Varianten und Anwendungsbeispiele*. Springer, 2019.

[25] S. Nisa, M. Rizwan, and M. R. Qureshi. Empirical estimation of hybrid model: A controlled case study. *International Journal of Information Technology and Computer Science*, 4, 07 2012.

[26] N. Paez, D. Fontdevila, and A. Oliveros. Helena study: initial observations of software development practices in argentina. In *Product-Focused Software Process Improvement: 18th International Conference, PROFES 2017, Innsbruck, Austria, November 29–December 1, 2017, Proceedings 18*, pages 443–449. Springer, 2017.

[27] R. F. Paige, R. Charalambous, X. Ge, and P. J. Brooke. Towards agile engineering of high-integrity systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 30–43. Springer, 2008.

[28] L. T. Portela and G. Borrego. Scrumconix: Agile and documented method to agsd. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, pages 195–196, 2016.

[29] N. Prenner, J. Klünder, and K. Schneider. Defining frames to structure agile development in hybrid settings-a multi-case interview study. In *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering*, pages 34–44, 2022.

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

[31] N. Prenner, C. Unger-Windeler, and K. Schneider. Goals and challenges in hybrid software development approaches. *Journal of Software: Evolution and Process*, 33(11):e2382, 2021.

[32] 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*, pages 5319–5328. IEEE, 2012.

[33] G. Rong, D. Shao, and H. Zhang. Scrum-psp: Embracing process agility and discipline. In *2010 Asia Pacific Software Engineering Conference*, pages 316–325, 2010.

[34] E. Scott, D. Pfahl, R. Hebig, R. Heldal, and E. Knauss. Initial results of the helena survey conducted in estonia with comparison to results from sweden and worldwide. In *Product-Focused Software Process Improvement: 18th International Conference, PROFES 2017,*

Innsbruck, Austria, November 29–December 1, 2017, Proceedings 18, pages 404–412. Springer, 2017.

[35] N. Sharma and M. Wadhwa. exsrup: Hybrid software development model integrating extreme programing, scrum & rational unified process. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 16(2):377–388, 2015.

[36] 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)*, pages 301–306, July 2017.

[37] 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*, pages 147–153. IEEE, 2009.

[38] K. R. Stoesser et al. *Prozessoptimierung für produzierende Unternehmen*. Springer, 2017.

[39] 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)*, pages 256–262, 2014.

[40] M. Tanveer. Agile for large scale projects—a hybrid approach. In *2015 National Software Engineering Conference (NSEC)*, pages 14–18. IEEE, 2015.

[41] L. Taxén and U. Pettersson. Agile and incremental development of large systems. In *7th European Systems Engineering Conference (EuSEC 2010)*, Stockholm, Sweden, May 2010.

[42] P. Tell, J. Klünder, S. Küpper, D. Raffo, S. G. MacDonell, J. Münch, D. Pfahl, O. Linssen, and M. Kuhrmann. What are hybrid development methods made of? an evidence-based characterization. In *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, pages 105–114. IEEE, 2019.

[43] P. Tell, R.-H. Pfeiffer, and U. P. Schultz. Helena stage 2—danish overview. In *Product-Focused Software Process Improvement: 18th International Conference, PROFES 2017, Innsbruck, Austria, November 29–December 1, 2017, Proceedings 18*, pages 420–427. Springer, 2017.

[44] G. Theocharis, M. Kuhrmann, J. Münch, and P. Diebold. Is water-scrum-fall reality? on the use of agile and traditional development practices. In *Product-Focused Software Process Improvement: 16th*

International Conference, PROFES 2015, Bolzano, Italy, December 2-4, 2015, Proceedings 16, pages 149–166. Springer, 2015.

[45] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[46] K. M. Zaki and R. Moawad. A hybrid disciplined agile software process model. In *The 7th International Conference on Informatics and Systems (INFOS)*, pages 1–8, 2010.