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

Vergleich von entwicklerzentrierten Softwareprozessattributen in Theorie und Praxis

Comparing developer-centered software process attributes in theory and industry

Masterarbeit

im Studiengang Informatik

von

Lukas Köhler

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüferin: Dr. Jil Klünder
Betreuerin: Dr. Jil Klünder

Hannover, 11.03.2024

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 11.03.2024

Lukas Köhler

Zusammenfassung

Vorgehensmodelle stellen den Rahmen für die Durchführung und das Management von Softwareprojekten dar. Plan-getriebene Vorgehensmodelle schreiben dabei ein ingenieurmäßiges Vorgehen vor. Dem gegenüber existieren agile Methoden, welche den Menschen in den Fokus der Softwareentwicklung rücken. Obwohl durch agile Methoden erfolgreicher Software entwickelt wird, scheitern dennoch regelmäßig Projekte. Ein Grund für dieses Scheitern sind die in das Projekt involvierten Entwickler. Dabei beeinflussen verschiedene Aspekte, wie beispielsweise eine starke Arbeitsbelastung, die Produktivität der involvierten Personen. Aus diesem Grund ist die Berücksichtigung der Bedürfnisse von Entwicklern von höchster Wichtigkeit.

Im Rahmen dieser Masterarbeit wird ein erster Einblick in die sogenannten *entwicklerzentrierten Softwareprozessattribute* geschaffen, welche im Zuge dieser Arbeit definiert werden. Es wird eine Literaturstudie durchgeführt, um die entwicklerzentrierten Softwareprozessattribute in den Lehrbüchern weit verbreiteter Methoden zu identifizieren. Darüber hinaus wird mithilfe einer Interviewstudie mit 22 Praktikern überprüft, welche Attribute sich in realen Softwareprozessen finden lassen. Abschließend wird ein Vergleich zwischen Theorie und Praxis gezogen. Dieser zeigt ein Vorhandensein vieler, der in der Literatur identifizierten, entwicklerzentrierten Softwareprozessattribute in der Praxis. Allerdings bieten die in den Interviews geäußerten Attribute, welche nicht durch die Prozesse erfüllt werden, Raum für Verbesserung. Diese Arbeit stellt den ersten Schritt dar, um die entwicklerzentrierten Softwareprozessattribute bereits bei der Prozesskonstruktion zu berücksichtigen.

Abstract

Process models provide the framework for the implementation and management of software projects. Plan-driven process models prescribe an engineering approach. In contrast, there are agile methods that focus on people in software development. Although software is developed more successfully using agile methods, projects still regularly fail. The developers involved in the project could be a significant reason for this failure. Various aspects, such as a heavy workload, influence the productivity of the people involved. For this reason, considering the needs of developers is of utmost importance.

This master's thesis provides an initial insight into the so-called *developer-centered software process attributes*, which are defined in the course of this work. A literature study is conducted to identify the developer-centered software process attributes in the textbooks of widely used methods. In addition, an interview study with 22 practitioners is used to examine which attributes can be found in real software processes. Finally, a comparison is drawn between theory and industry. This shows the existence of many of the developer-centered software process attributes identified in the literature in practice. However, the attributes expressed in the interviews, which are not fulfilled by the processes, offer room for improvement. This thesis represents the first step towards taking developer-centered software process attributes into account as early as the process design stage.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Vorgehensmodelle bilden die Basis für ein zweckbestimmtes Vorgehen in der Softwareentwicklung [35]. Sie definieren eine Abfolge von Schritten, welche benötigt werden, um ein festgelegtes Ziel zu erreichen. Die ersten Prozesse, die so genannten plan-getriebenen Vorgehensmodelle, orientieren sich dabei an dem ingenieurmäßigen Vorgehen anderer Disziplinen [2]. Im Jahr 2001 beschrieben Beck et al. [4] mit dem *agilen Manifest* einen neuen Blickwinkel an die Herangehensweisen der Softwareentwicklung. Die auf den Prinzipien des Manifests basierenden agilen Methoden verschieben den Fokus weg von einer vollständigen, stringenten Planung des Projekts hin zu einem iterativen Vorgehen [13]. Dabei stehen die involvierten Menschen im Mittelpunkt des Prozesses [13].

1.1 Motivation

Die Nutzung der agilen Methoden zeigt einen positiven Einfluss auf den erfolgreichen Abschluss von Softwareprojekten [49]. Dennoch scheitern Projekte häufig [21]. Einen Grund für dieses Scheitern stellen die involvierten Menschen und, unter anderem, auch das Entwicklerteam dar [55]. Verschiedene Aspekte, wie beispielsweise die Atmosphäre und Zusammenarbeit im Team, beeinflussen die Arbeitsweise von Entwicklern [60]. Außerdem zeigten Graziotin et al. [18], dass unglückliche Entwickler weniger produktiv sind. Zudem begünstigen berufliche Überlastung und Anspannungen bei der Arbeit ein Burnout der Entwickler [58]. Mit dem Ziel, Mitarbeiter im schnelllebigen Markt der Softwareentwicklung langfristig anzustellen, müssen Unternehmen dafür Sorge tragen, dass die Entwickler gerne bei ihnen arbeiten. Ein wichtiger Aspekt ist die positive Gestaltung der Arbeit und ihres Arbeitsalltags durch den Prozess. Dabei ist die Berücksichtigung der Vorstellungen der Entwickler von zentraler Bedeutung. Die Zufriedenheit mit ihrem Prozess kann lediglich subjektiv, unter Einbeziehung der betroffenen Entwickler, beurteilt werden.

1.2 Zielsetzung

Im Rahmen dieser Arbeit soll die Grundlage der entwicklerzentrierten Softwareprozessattribute (EZSPA) geschaffen werden. Die EZSPA, welche im Rahmen dieser Arbeit definiert und beschrieben werden, stellen einen wichtigen Aspekt der entwicklerzentrierten Prozessgestaltung dar. Im Gegensatz zu den Pflichten, welche den Entwicklern im Rahmen ihres Entwicklungsprozesses auferlegt werden, beschreiben diese die Rechte der Entwickler. Sie umfassen dabei verschiedene Aspekte, die im Verlauf dieser Arbeit erläutert werden. Ein Beispiel für solch ein entwicklerzentriertes Softwareprozessattribut stellen Weiterbildungsmöglichkeiten im Arbeitsalltag oder der Wissensaustausch dar. Die genaue Definition der EZSPA ist in Abschnitt 6.1 zu finden.

Das abschließende Ziel dieser Masterarbeit ist ein Vergleich des Auftretens entwicklerzentrierter Softwareprozessattribute in Theorie und Praxis. Auf diese Weise soll ein erster Schritt getan werden, um die Attribute langfristig bei der Konstruktion von Prozessen berücksichtigen zu können. Dafür wird eine Literaturstudie durchgeführt, um die theoretische Basis der EZSPA zu schaffen. Auf Grundlage der Ergebnisse findet eine Definition der entwicklerzentrierten Softwareprozessattribute statt. Es wird eine Interviewstudie durchgeführt, um auch die Ansichten der Praktiker berücksichtigen zu können, und um final den beabsichtigten Vergleich zwischen Theorie und Praxis vollziehen zu können. Das Erreichen des definierten Ziels wird durch den in Abbildung 1.1 skizzierten Verlauf dieser Arbeit angestrebt.

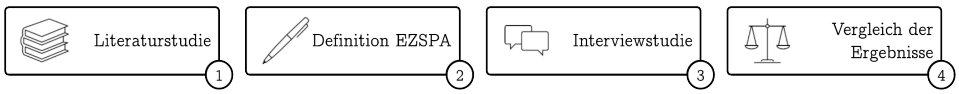

Abbildung 1.1: Der Ablauf dieser Arbeit

Der Fokus dieser Arbeit liegt bewusst auf einem tiefgehenden Vergleich der theoretischen Beschreibung und dem praktischem Vorkommen der EZSPA. Weiterführende Schritte, wie etwa die Konstruktion eines Vorgehensmodells mit expliziter Berücksichtigung der EZSPA oder die Aussprache von Empfehlungen an die Industrie, sind Gegenstand künftiger Forschung.

1.3 Struktur der Arbeit

Diese Arbeit ist folgendermaßen strukturiert. In Kapitel 2 werden die Grundlagen erläutert, welche für das Verständnis dieser Arbeit notwendig sind. Es werden verschiedene Methoden bzw. Vorgehensmodelle, wie beispielsweise *Scrum*, vorgestellt. Im anschließenden Kapitel 3 folgt ein Ausblick auf die Publikationen, welche ein mit dieser Arbeit verwandtes Ziel verfolgt haben. Daraufhin wird in Kapitel 4 die Methodik der durchgeführten, mehrstufigen

Studie dargestellt, die dieser Arbeit zugrunde liegt. Dafür wird zu Beginn erläutert, auf welche Weise die Literaturstudie durchgeführt worden ist. Anschließend wird die Planung und Durchführung der Interviewstudie beschrieben. Zudem wird beschrieben, wie der Vergleich zwischen Theorie und Praxis stattgefunden hat. Nachfolgend werden in Kapitel 5 die Ergebnisse der Studie erläutert, bevor diese in Kapitel 6 diskutiert werden. Außerdem findet in diesem Kapitel auch eine Interpretation der Ergebnisse statt und die vorher definierten Forschungsfragen werden beantwortet. Abschließend wird in Kapitel 7 eine Zusammenfassung der Arbeit präsentiert und zudem ein Ausblick darauf gegeben, wie sich die Forschung zukünftig in diesem Bereich gestalten könnte.

Kapitel 2

Grundlagen

In diesem Kapitel werden die Grundlagen dargestellt, die für das Verständnis der Arbeit notwendig sind. Zu Beginn wird die relevante Terminologie erläutert, welche im Rahmen dieser Arbeit verwendet wird. Anschließend werden die verschiedenen Vorgehensmodelle beschrieben, die im weiteren Verlauf eine Rolle spielen werden.

2.1 Terminologie

Seit den Anfängen der Softwareentwicklung haben sich unterschiedliche Herangehensweisen an die Durchführung und das Management von Softwareprojekten herausgebildet. Die sogenannten *plan-getriebenen Methoden* orientieren sich an dem ingenieurmäßigen Vorgehen anderer Disziplinen [2]. Es werden Phasen definiert, welche vollständig nacheinander durchlaufen werden müssen. Die Anforderungen an die Software werden im Vorfeld der Entwicklung definiert, was Änderungen im Nachhinein erschwert [13]. Eine klassische Implementation dieses Phasenmodells ist das in Unterabschnitt 2.2.4 beschriebene Wasserfallmodell nach Royce [47].

Mit dem *Agilen Manifest* [4] rückten ab dem Jahr 2001 die sogenannten *agilen Methoden* immer mehr in den Fokus. Es existieren unterschiedliche Herangehensweisen an die agile Entwicklung, allerdings teilen sie sich alle ähnliche Werte [13]. Der Schwerpunkt agiler Methoden liegt auf der Interaktion und Kommunikation beteiligter Akteure und einem iterativen Vorgehen, um schnell auf Änderungen reagieren zu können [13].

Zwischen diesen beiden Vorgehensweisen existieren hybride Methoden. Diese kombinieren unterschiedliche Herangehensweisen aus beiden Bereichen in einen hybriden Ansatz für die Softwareentwicklung [25].

2.1.1 Vorgehensmodelle

Die erfolgreiche Entwicklung von Software in einem Unternehmen wird maßgeblich durch die Verwendung geeigneter Vorgehensmodelle beeinflusst [10]. Sie stellen die Brücke zwischen der Projektleitung und -durchführung dar und dienen als Grundlage für die Planung, Überwachung und Steuerung von Softwareprojekten [10]. Broy und Kuhrmann [10] definieren ein Vorgehensmodell als eine „*systematische, ingenieurmäßige und quantifizierbare Vorgehensweise, um Aufgaben einer bestimmen Klasse wiederholbar zu lösen*“.

2.1.2 Methoden

Vorgehensmodelle bieten außerdem die Möglichkeit, sogenannte *Methoden* zu integrieren, um bestimmte Aufgaben im Softwareprojekt lösen zu können [10]. Brinkkemper [8] definiert eine Methode als einen auf Regeln und Vorgaben basierenden Ansatz zur Lösung eines vorgegebenen Problems. Broy und Kuhrmann [10] geben außerdem an, dass das Vorgehen bei der Problemlösung durch eine Methode strukturiert wird. Darüber hinaus geben Methoden Hinweise zur Erstellung von Artefakten[1] an [10].

2.2 Vorgehensmodelle

Im Folgenden werden einige Vorgehensmodelle beschrieben, welche laut der HELENA-Studie[2] [28] in der Praxis eine wichtige Rolle spielen und aus diesem Grund auch für diese Arbeit analysiert wurden.

2.2.1 Scrum

Bei Scrum handelt es sich um „[...] *ein leichtgewichtiges Rahmenwerk, welches Menschen, Teams und Organisationen hilft, Wert durch adaptive Lösungen für komplexe Probleme zu generieren*“ [51]. Es wurde im Jahr 1993 das erste Mal verwendet [54] und stellt ein Vorgehensmodell im Projektmanagement dar [42]. Es findet vor allem, aber nicht ausschließlich, in der agilen Softwareentwicklung Anwendung und definiert selbst nur wenige Regeln [42]. Scrum repräsentiert dabei die Werte des agilen Manifests von Beck et al. [4].

In kurzen Iterationen, den sogenannten *Sprints*, werden neue Versionen bzw. Inkremente der Software mit Features aus dem *Product-*

[1] Ein Artefakt beschreibt jegliche Elemente, die während eines Softwareprojekts entstehen können. Dabei kann es sich um Quellcodedateien handeln, aber auch um textuelle Beschreibungen von Softwareanforderungen [32].

[2] Im Rahmen der HELENA-Studie [28] wurde untersucht, welche Methoden und Praktiken in Unternehmen genutzt und miteinander kombiniert werden. Dafür wurden in der zweiten von drei Phasen Daten von ca. 1500 Unternehmen gesammelt.

und Sprintbacklog erarbeitet [42]. Im Productbacklog werden jegliche, noch umzusetzende Anforderungen abgelegt. Im Sprintbacklog finden sich hingegen nur jene wieder, welche im *Sprint Planning* Meeting für den kommenden Sprint geplant werden.

Das Framework definiert die Rollen *Scrum Master*, *Product Owner* und das *Scrum Team*, welche alle eng miteinander arbeiten müssen, um funktioniere Software zu generieren [42]. Das Scrum Team ist dabei für die eigentliche Entwicklung der Software verantwortlich und arbeitet weitestgehend selbstbestimmt [38]. Der Product Owner repräsentiert die Stakeholder[3] der Software und stellt auf diese Weise die Verbindung zwischen ihnen und der Entwicklung her [38]. Zusätzlich verantwortet er die Erreichung wirtschaftlicher Ziele und priorisiert aus diesem Grund die nächsten Entwicklungsschritte des Projekts [42]. Der Scrum Master stellt sicher, dass das Entwicklungsteam alle zur Verfügung stehenden Mittel nutzt und versucht alle Hindernisse zu beseitigen, auf die das Team stößt [50]. Außerdem ist er dafür verantwortlich, dass das Team sich an den agilen Werten und Praktiken orientiert [38].

Neben dem *Daily Scrum*, einem täglichen Austauschmeeting, gibt es noch das *Sprint Review*. Hier werden Fortschritte besprochen, die im Sprint erreicht wurden und es wird Feedback von allen Stakeholdern eingeholt [42]. Da es sich bei Scrum um einen empirischen Prozess handelt, kann nach dem Sprint in der *Sprint Retrospective* die Arbeitsweise überprüft und angepasst werden. Auf diese Weise kann eine kontinuierliche Verbesserung ermöglicht werden [42].

2.2.2 Extreme Programming

Extreme Programming (XP) stellt ein Vorgehensgehensmodell in der Softwareentwicklung dar, welches auf kurzen Iterationen basiert [52]. Es gründet auf der Annahme, dass der Kunde die eigentlichen Anforderungen an das Projekt selbst nicht von Beginn an kennt und setzt daher eine kontinuierliche Zusammenarbeit zwischen Stakeholdern und dem Projektteam voraus [52]. Aus diesem Grund wird auch eine kurzfristige Anpassung der Kundenwünsche oder Prioritäten einzelner Features ermöglicht [3, 52].

Das Modell basiert auf verschiedenen Werten und Praktiken, welche mithilfe von Prinzipien verbunden werden sollen [3], wie in Abbildung 2.1 symbolisch dargestellt wird.

Ohne Berücksichtigung dieser könnte eine Softwareentwicklung nach XP nicht erfolgreich sein [3]. Die fünf elementaren Werte werden dabei von *Kommunikation, Einfachheit, Feedback, Mut* und *Respekt* verkörpert [3, 52]. Zu den 14 Prinzipien zählen Punkte wie *Menschlichkeit* oder *Wirtschaftlichkeit* [3]. Diese stellen eine Brücke zu den Praktiken her,

[3]Stakeholder sind Personen(gruppen) mit einem berechtigten Interesse an einem Projekt oder dessen Ergebnis [40].

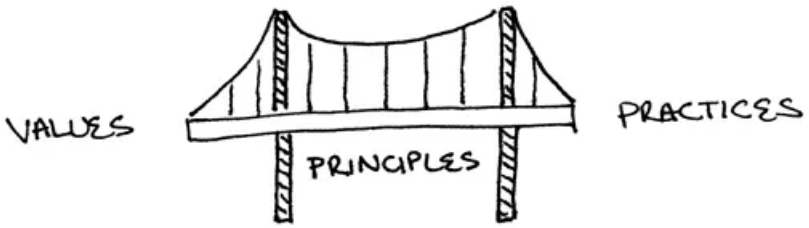

Abbildung 2.1: Nach XP verbinden Prinzipien Werte und Praktiken [3]

welche XP ausmachen. Zu diesen gehören Vorgänge wie *Pair Programming*, *kontinuierliche Integration* oder *Refactoring*.

Mit dem Ziel, eine effektive Kommunikation zu ermöglichen, spielt die räumliche Nähe der Mitglieder des Entwicklungsteams eine wichtige Rolle [3, 52]. Hierdurch soll gleichzeitig eine gute Zusammenarbeit mit dem Kunden ermöglicht werden, da er sich auf diese Weise als Teil des Teams fühlen kann [52]. Damit soll die Zeit verringert werden, welche die Entwickler mit dem Lesen von Dokumentation verbringen, sodass mehr Zeit für die eigentliche Entwicklung genutzt werden kann [52].

2.2.3 DevOps

DevOps ist ein Vorgehensmodell, welches zwei grundlegende Komponenten eines Softwareprojekts miteinander kombiniert. Das Wort DevOps selbst ist ein Kofferwort[4] und verbindet die beiden integralen Bestandteile der Entwicklung (engl. **Dev**elopment) und des Betriebs (engl. **Op**eration**s**) miteinander, ebenso wie das Modell selbst. Dabei wird die Silostruktur der Teams in der Entwicklung und dem Betrieb aufgebrochen [16]. Die beiden Abteilungen werden zu einem sogenannten *crossfunktionalen Team* verschmolzen, welches die Aufgaben beider Bereiche übernimmt [16]. Auf diese Weise lässt sich die Misskommunikation im Team verringern, wodurch die Software schneller weiterentwickelt und neue Versionen kontinuierlich bereitgestellt werden können [16]. Dafür existiert ein fortlaufender Fluss zwischen den Aufgaben der Entwicklung, wie beispielsweise der Implementierung oder dem Testen, und den Aufgaben des Betriebs, wie dem Bereitstellen der entwickelten Software.

2.2.4 Wasserfallmodell

Das Wasserfallmodell nach Royce [47] beschreibt ein klassisches Vorgehensmodell in der Softwareentwicklung. Die grundlegende Version des Modells ist in Abbildung 2.2 dargestellt.

[4]Ein Kofferwort beschreibt eine inhaltliche Wortneuschöpfung aus Teilen von mind. zwei anderen Wörtern.

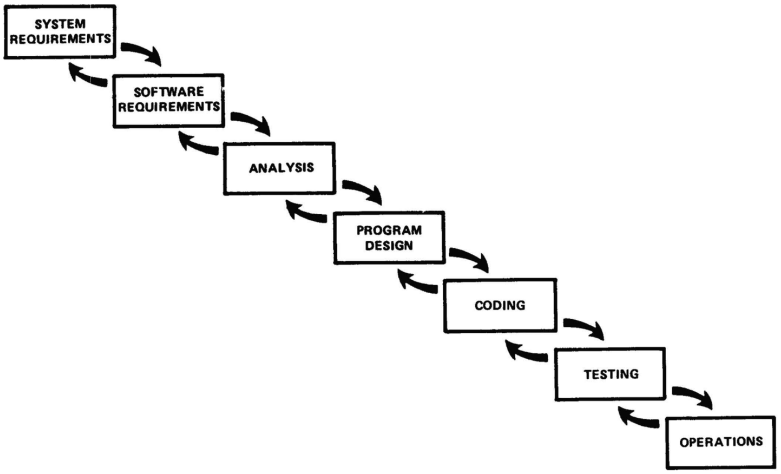

Abbildung 2.2: Das grundlegende Wasserfallmodell nach [47]

Die Phasen werden streng nacheinander durchlaufen, wobei ein Rücksprung nur in den direkt vorhergehenden Schritt möglich ist [47]. Zu Beginn werden die Anforderungen an das System und die Software identifiziert. Anschließend folgt der Entwurf der Architektur. Als Nächstes wird das System implementiert und getestet. Nach erfolgreichem Testen wird die Software in Betrieb genommen.

Jede Phase endet mit einem Meilenstein, wenn ein festgelegtes Ergebnis vorliegt [10]. Für den Beginn des nächsten Schritts sind sowohl die Vollständigkeit aller definierten Artefakte als auch die ordnungsgemäße Durchführung sämtlicher vorangegangener Aktivitäten erforderlich [10].

2.2.5 Spiralmodell

Das in Abbildung 2.3 dargestellte Spiralmodell nach Boehm [5] beschreibt ein grundlegendes Beispiel für die iterative Softwareentwicklung [10].

Dabei werden die Funktionalitäten der Software stufenweise aufeinander aufbauend entwickelt, um auf diese Weise Risiken zu minimieren und den Fortschritt des Projekts besser messen zu können [10]. Das Spiralmodell definiert dabei vier Phasen, welche den Sektoren des Koordinatensystems in Abbildung 2.3 zugeordnet werden können [10]. Die Iterationen starten im zweiten Quadranten mit der Analyse. Hier werden die Rahmenbedingungen des Projekts und verschiedene Lösungsvorschläge analysiert. Dabei spielt die Entwicklung von Prototypen eine wichtige Rolle. Diese können im anschließenden Schritt, der Evaluierung umgesetzter Lösungen, überprüft werden, um Risiken zu identifizieren und zu vermeiden. Im dritten Schritt wird die Realisierung entsprechend geplant und umgesetzt. Zuletzt folgt die Planung. Es werden die vorangegangenen Schritte überprüft und im Anschluss mit der Planung der nächsten Iteration begonnen.

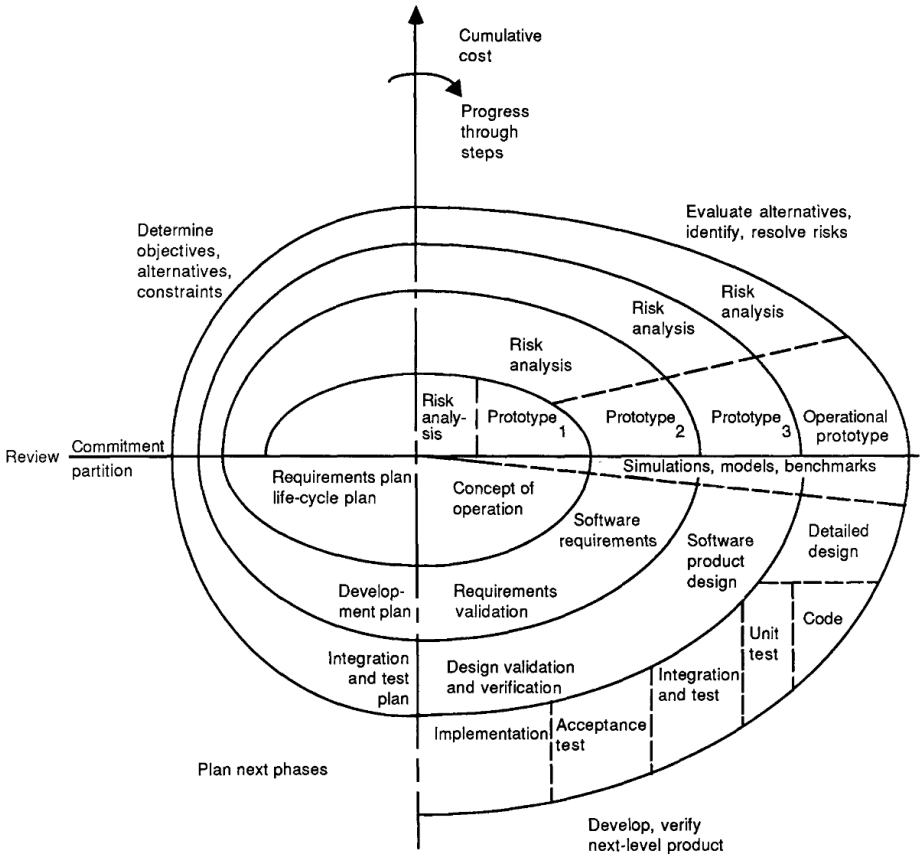

Abbildung 2.3: Das Spiralmodell nach Boehm [5]

Kapitel 3

Verwandte Arbeiten

Diese Arbeit befindet sich an der Schnittstelle zwischen der Forschung an Softwareprozessen und der entwicklerzentrierten Forschung. Nach bestem Wissen und Gewissen stellt sie den ersten Schritt für die Kombination dieser beiden Bereiche dar. Daher werden an dieser Stelle verwandte Arbeiten aus diesen beiden Forschungsfeldern vorgestellt.

3.1 Softwareprozesse

Kuhrmann et al. [26] haben in ihrer Arbeit untersucht, welche Methoden und Praktiken reale Softwareprozesse umfassen. Außerdem wurde überprüft, wie diese Ansätze miteinander kombiniert werden und ob externe Einflüsse für die Implementation von hybriden Ansätzen verantwortlich sind. Die Erhebung basiert dabei auf den Ergebnissen der HELENA-Studie [28].

Die Untersuchungen [26] zeigten, dass eine Vielzahl verschiedener Entwicklungsansätze von den Probanden verwendet werden. Darunter fallen sowohl agile als auch plan-getriebene Ansätze. Beispielsweise gaben über 50% an, dass sie nach Scrum arbeiten würden. Außerdem stellte sich heraus, dass Unternehmen sowohl agile als auch plan-getriebene Ansätze kombinieren würden, um Vorteile beider Herangehensweisen ausnutzen zu können. Es konnte kein Einfluss auf die Implementation hybrider Ansätze durch externe Ursachen, wie beispielsweise der Erfüllung von Standards, nachgewiesen werden.

In einer weiteren Arbeit haben Kuhrmann et al. [27] auf Basis der HELENA-Studie [28] untersucht, welche Methoden und Praktiken den wahrgenommenen Grad der Agilität eines Projekts beeinflussen. Außerdem wurde untersucht, wie agil die einzelnen Disziplinen innerhalb von Projekten, beispielsweise das Architekturdesign oder die Implementierung, durchgeführt werden.

Kuhrmann et al. [27] haben gezeigt, dass ein Großteil der Projektdisziplinen agil durchgeführt werden. Ca. 15% aller Probanden gaben an, ein Projekt vollständig agil oder vollständig plan-getrieben durchzuführen. Es wurde gezeigt, dass sowohl jede Methode als auch jede Praktik in agiler und plan-getriebener Entwicklung gefunden werden kann. Allerdings wurden verschiedene Methoden und Praktiken identifiziert, welche den Grad der Agilität eines Projekts beeinflussen. Hierunter fällt beispielsweise Scrum als Methode oder das Daily-Standup als Praktik.

Tell et al. [57] haben in ihrer Publikation untersucht, wie sich hybride Methoden in der Praxis zusammensetzen. Die Ergebnisse basieren dabei ebenso auf den Daten, welche im Rahmen der HELENA-Studie [28] gesammelt wurden.

Es wurde gezeigt, dass in der Industrie eine Vielzahl verschiedener Prozesse existieren, die auf unterschiedlichen Methoden und Praktiken basieren und zu einem hybriden Modell kombiniert werden. Dabei werden nur wenige dieser Methoden und Praktiken als Basis für diese Kombinationen genutzt. Des Weiteren wurde ein Verfahren entwickelt, um hybride Prozesse evidenzbasiert zu charakterisieren und zu beschreiben.

Die Arbeit von Klünder et al. [25] basiert ebenso auf den Ergebnissen der HELENA-Studie [28]. Die Forscher haben untersucht, welche hybriden Entwicklungsansätze in der Praxis verwendet werden, wie diese Ansätze entwickelt werden und inwiefern unterschiedliche Verfahren zur Prozesskonstruktion mehr oder weniger erfolgreich im Bezug auf die Erfüllung selbstgesetzter Ziele sind.

Dabei wurde festgestellt, dass über 75% aller Probanden verschiedene Ansätze zu einem, für sie angepassten, Hybriden kombinieren würden. Scrum und iterative Entwicklung gehören zu den am häufigsten verwendeten Methoden. Hybride Ansätze entstehen in erster Linie durch eine Weiterentwicklung des aktuellen Vorgehens und werden aufgrund verschiedener Ziele entwickelt. Unter anderem wurden hier eine höhere Produktivität oder eine verbesserte Planung als Ziel genannt. Des Weiteren gaben die Autoren an, dass verschiedene Strategien für die Prozessentwicklung existieren. Die systematische Verbesserung eines Softwareprozesses erhöht die Wahrscheinlichkeit, gesetzte Ziele zu erreichen.

Vijayasarathy et al. [59] haben in ihrer Arbeit ebenfalls untersucht, inwiefern verschiedene Entwicklungsansätze in der Industrie zu finden sind. Ein besonderer Blick wurde dabei auf die Zusammenhänge zwischen den genutzten Ansätzen und den Charakteristika des Unternehmens, des Projekts und des Teams gelegt. Für die Datenerfassung wurde eine Onlineumfrage mit insgesamt 153 Probanden durchgeführt.

Es stellte sich, im Einklang mit der anfangs beschriebenen Arbeit von Kuhrmann et al. [26], heraus, dass agile und hybride Methoden häufig Anwendung in der Praxis finden. Außerdem konnten die Forscher zeigen, dass die oben genannten Eigenschaften in Verbindung mit der Wahl der Entwicklungsmethode stehen. Beispielsweise ließen sich traditionelle Vorgehensweisen mit einem hohen Projektbudget in Verbindung setzen, während agile Methoden häufig mit einem geringen Budget arbeiten müssten.

3.2 Entwicklerzentrierte Forschung

Das Ziel der Arbeit von Storey et al. [53] war es, herauszufinden, auf welche Weise technische und soziale Faktoren die Zufriedenheit von Softwareentwicklern beeinflussen. Basierend auf einer Umfrage mit 465 Entwicklern wurde eine Theorie entwickelt, welche beschreibt, wie diese Faktoren die Verbindung zwischen Zufriedenheit und wahrgenommener Produktivität beeinflussen.

Sie zeigten, dass sowohl soziale als auch technische Faktoren die Produktivität und die Zufriedenheit von Entwicklern beeinflussen. Als relevante Faktoren für die Zufriedenheit wurden unter anderem die Arbeitskultur und eine Wertschätzung und Belohnung der Arbeit genannt. Für die Produktivität stellten sich insbesondere die Aspekte der Autonomie und das Erfüllen von Aufgaben als wichtig heraus. Es bestehe ein bidirektionaler Zusammenhang zwischen der Produktivität und Zufriedenheit, da sich diese beiden Aspekte gegenseitig beeinflussen würden.

Mit der Frage, durch welche Faktoren die selbst bewertete Produktivität von Softwareentwicklern beeinflusst und vorhergesagt werden kann, haben sich Murphy-Hill et al. [36] beschäftigt. Dafür haben sie Umfragen mit insgesamt 622 Entwicklern aus drei verschiedenen Unternehmen durchgeführt. Außerdem wurde an dieser Stelle untersucht, inwiefern sich diese Faktoren über die verschiedenen Unternehmen hinweg unterscheiden.

Im Rahmen der Studie zeigte sich, dass die zehn wichtigsten Produktivitätsfaktoren nicht technischer Natur waren. Die drei Wichtigsten stellten die Begeisterung für den Beruf, eine gegenseitige Unterstützung für neue Ideen im Team und Feedback für die eigene Arbeit dar. Diese Aspekte wurden in allen untersuchten Unternehmen als wichtig erachtet. Außerdem wurde Homeoffice als Produktivitätsfaktor identifiziert.

Salido O. et al. [48] haben in ihrer Arbeit eine *Systematic Mapping Study* durchgeführt, um auf Basis von Literatur herauszufinden, welche affektiven Zustände die Leistung agiler Entwickler und die Qualität der entwickelten Software beeinflussen.

Sie gaben an, dass die affektiven Zustände der Entwickler Auswirkungen auf verschiedene Aspekte des Entwicklungsprozesses haben würden. Unter anderem könnten sie sowohl positive als auch negative Auswirkungen auf die Qualität der entwickelten Software oder die Entwicklungszeit haben. Außerdem führten sie an, dass menschliche Faktoren, wie beispielsweise die Motivation oder das Verhalten von Personen, bei der Bewertung von Mitarbeitern im agilen Umfeld berücksichtigt werden müssen. Sie unterstützen dabei Ziele zu erreichen und Beziehungen im Team aufzubauen. Daher sei die Forschung in diesem Bereich von höchster Wichtigkeit, um das Wohlbefinden der Entwickler zu gewährleisten.

Johnson et al. [22] haben einen mehrstufigen *Mixed-Methods-Ansatz* gewählt, um zu überprüfen, welche Einflüsse die Arbeitsumgebung auf die Produktivität und Zufriedenheit von Entwicklern hat. Sie haben mithilfe einer initialen Umfrage Teilnehmer für eine Interviewstudie akquiriert, an welcher 19 Personen teilgenommen haben. Anschließend folgte eine weitere Umfrage mit 843 Teilnehmern, mit dem Ziel tiefergehende Informationen über die Arbeitsumgebung der Entwickler zu erhalten.

Die Ergebnisse zeigen, dass die Produktivität von Entwicklern durch unterschiedliche Faktoren beeinflusst wird. Die Zufriedenheit mit der Arbeitsumgebung stellte dabei einen wichtigen Aspekt dar. Zudem sei es wichtig, offen und schnell mit Teammitgliedern kommunizieren zu können. Private Büroräume wurden mit einer erhöhten Produktivität in Verbindung gesetzt, was zeigte, dass Rückzugsorte für konzentrierte Arbeitsphasen ebenso relevant sind.

Graziotin et al. [17] haben sich mit der Frage beschäftigt, aus welchen Gründen Softwareentwickler im Rahmen ihrer Arbeit (un)glücklich sind. Das Ziel der Forscher war es, die Unzufriedenheit der Entwickler zu limitieren. Nach Angaben von ihnen ausgewählter Literatur stelle dies einen kostengünstigen Weg dar, die Zufriedenheit und Produktivität zu fördern. Dafür wurde eine Umfrage mit insgesamt 1318 vollständigen Antworten durchgeführt.

Auf Basis der SPANE-B Skala [15] zeigten die Forscher, dass Softwareentwickler einen leicht glücklichen Bevölkerungsanteil darstellen. Mithilfe dieser Skala lässt sich einzelnen Probanden ein Glücklichkeitswert zuordnen [17]. Außerdem führten sie unterschiedliche Gründe für die Unzufriedenheit von Entwicklern an, die sich in zwei Kategorien einordnen lassen. Zum einen existieren innere Ursachen, wie beispielsweise das Gefühl für die eigene Arbeit unzulänglich zu sein. Zum anderen sind es äußere Aspekte. Zu denen durch den Prozess vorgegebene Merkmale, wie etwa Zeitdruck, zählen.

3.3 Abgrenzung dieser Arbeit

Viele der zuvor beschriebenen verwandten und entwicklerzentrierten Publikationen haben sich mit unterschiedlichen Aspekten der Arbeit von Softwareentwicklern beschäftigt und bilden daher eine gute Basis für die vorliegende Arbeit. Erkenntnisse aus der Forschung sowohl mit Fokus auf die Entwickler als auch auf die Prozesse bilden den Grundstein zur Analyse von entwicklerzentrierten Softwareprozessattributen.

Die Frage, wie durch die Ausrichtung und Ausgestaltung von Softwareprozessen die Zufriedenheit und die Produktivität der Entwickler erhöht werden können, wurde bislang noch nicht untersucht. Die vorliegende Arbeit liefert einen ersten Schritt in diese Richtung, indem untersucht wird, welche Rechte Softwareprozesse ihren Entwicklern theoretisch als auch in der praktischen Umsetzung zusprechen. Auf der Basis von einer Kombination zwischen entwicklerzentrierter Forschung und Forschung an Softwareprozessen sollen eben diese Rechte herausgearbeitet werden. Hierdurch sollen die Entwickler langfristig bereits bei der Prozesskonstruktion berücksichtigt werden, um durch den Prozess ihre Zufriedenheit zu stärken.

Kapitel 4

Methodik

In diesem Kapitel wird die Methodik der durchgeführten Studie beschrieben. Das Ziel dieser Arbeit ist es, einen Vergleich zwischen Theorie und Praxis der EZSPA zu ermöglichen. Aus diesem Grund wurde eine mehrstufige Studie durchgeführt. Dessen Ablauf ist in Abbildung 4.1 dargestellt.

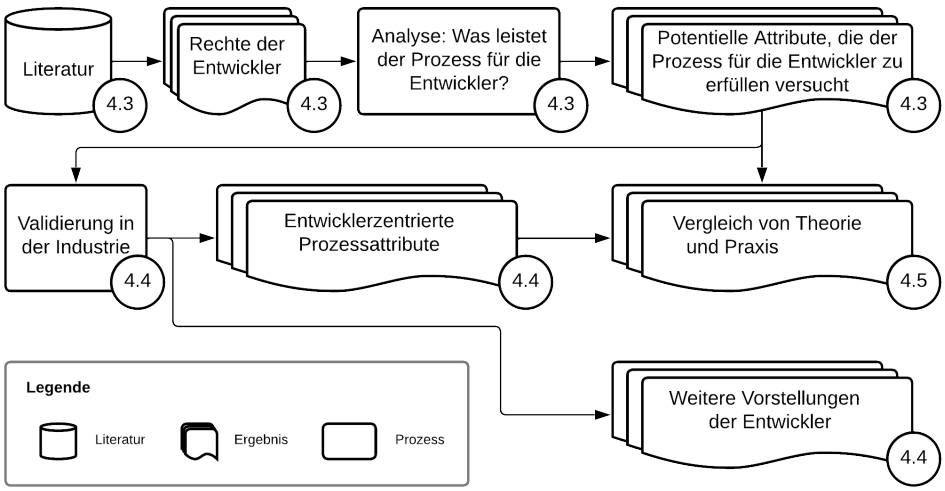

Abbildung 4.1: Der Ablauf der durchgeführten Studien. Die Zahlen verweisen auf die zugehörigen Abschnitte in diesem Kapitel.

Zu Beginn wurden Rechte der Entwickler aus Prozessbeschreibungen gängiger Vorgehensmodelle herausgearbeitet. Daraus wurde im Anschluss extrahiert, was die Vorgehensmodelle für die Entwickler versprechen zu leisten. Auf diese Weise entstanden potentielle EZSPA, welche anschließend durch eine Interviewstudie mit Praktikern im Hinblick auf ihr Vorkommen evaluiert wurden. Das Ergebnis dieses Schrittes stellen die eigentlichen EZSPA und weitere Vorstellungen der Entwickler über EZSPA dar.

4.1 Zielsetzung

Das übergeordnete Ziel der Studie ist ein Vergleich der EZSPA zwischen der theoretischen Beschreibung in der untersuchten Literatur und der praktischen Anwendung der Methoden. Aus diesem Grund wurde für die Studie folgende Zielsetzung in Anlehnung an die Vorlage von Wohlin et al. [61] entwickelt:

Zielsetzung

Analysiere Lehrbücher und reale Softwareprozesse
aus Sicht von Entwicklern
im Rahmen von einer Studie
in Bezug auf entwicklerzentrierte Softwareprozessattribute
mit dem Ziel die in der Praxis vorhandenen Attribute mit den in der Literatur beschriebenen zu vergleichen.

4.2 Forschungsfragen

Die im Folgenden beschriebenen Forschungsfragen (engl. Research Questions (RQs)) wurden gestellt, um das in Abschnitt 4.1 definierte Forschungsziel zu erreichen. Mithilfe der ersten Forschungsfrage sollen die EZSPA der Vorgehensmodelle festgehalten werden, wenn sie nach ihren Lehrbüchern ausgeführt werden. Durch Forschungsfrage 2 soll ein Einblick in die EZSPA realer Softwareprojekte ermöglicht werden. Schließlich soll durch die dritte Forschungsfrage ein Vergleich zwischen Theorie und Praxis der EZSPA ermöglicht werden.

RQ1: Entwicklerzentrierte Prozessattribute nach Lehrbuch

In der Primärliteratur über Vorgehensmodelle werden unterschiedliche Aspekte beleuchtet. Neben typischen Prozessanforderungen, wie beispielsweise einem pünktlichen Projektabschluss, lassen sich hier auch entwicklerzentrierte Prozessattribute identifizieren. Forschungsfrage 1 zielt darauf ab, einen Überblick über die vorhandenen Attribute aus der Literatur zu schaffen. Auf diese Weise soll festgehalten werden, welche EZSPA von welchen Vorgehensmodellen erfüllt werden. Dabei wurde sich in dieser Arbeit auf fünf häufig verwendete Vorgehensmodelle [28] beschränkt:

RQ1

Welche entwicklerzentrierten Prozessattribute werden von weit verbreiteten Entwicklungsmethoden (Scrum, XP, DevOps, Spiralmodell und Wasserfall) erfüllt, wenn diese nach Lehrbuch ausgeführt werden?

RQ2: Entwicklerzentrierte Prozessattribute in realen Prozessen

Aufgrund unterschiedlicher Gegebenheiten werden Prozesse nur selten ausgeführt, wie sie in der Literatur beschrieben werden [56]. Außerdem existiert keine Forschung, welche sich mit EZSPA und ihrer Anwendung in realen Softwareprozessen beschäftigt. Aus diesem Grund soll mithilfe einer Beantwortung der zweiten Forschungsfrage auf der Basis von Interviews mit Entwicklern untersucht werden, wie die EZSPA in der Praxis gestaltet sind:

RQ2

Welche entwicklerzentrierten Prozessattribute finden sich in realen Softwareprozessen?

RQ3: Vergleich zwischen Theorie und Praxis

Auf der Basis der Antworten auf die ersten beiden Forschungsfragen ist es möglich, einen Vergleich zwischen Theorie und Praxis zu ziehen. Dabei werden die aus der Literatur extrahierten EZSPA mit den Vorstellungen und Aussagen der Interviewprobanden verglichen. Auf diese Weise sollen Übereinstimmungen ebenso wie Unterschiede identifiziert werden. Außerdem können weitere Vorstellungen der Entwickler aufgenommen werden, welche die Literatur nicht berücksichtigt hat:

RQ3

Inwiefern decken sich die aus der Literatur extrahierten Prozessattribute mit den Vorstellungen der Entwickler?

4.3 Literaturstudie

Der erste Teil der Studie umfasst eine Literaturstudie, die Primärliteratur verschiedener Vorgehensmodelle untersucht. Dabei wurde kein Literaturreview im eigentlichen Sinne durchgeführt, wie sie beispielsweise in der Arbeit von Boland et al. [6] beschrieben wird. Das Ziel dieses Studienteils war es, zu analysieren, welche EZSPA etablierte Softwareprozesse ihren Entwicklern nach Lehrbuch bieten sollen.

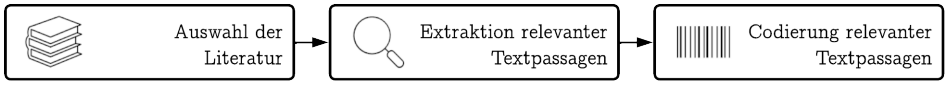

Abbildung 4.2: Ablauf der Literaturstudie

In Abbildung 4.2 ist der Ablauf dieser Teilstudie schematisch dargestellt. Zu Beginn wurde die zu analysierende Literatur auf Basis der Ergebnisse

der 2017 durchgeführten HELENA-Studie [28] ausgewählt. Hierbei wurden Methoden und Vorgehensmodelle selektiert, welche eine hohe Relevanz in der Praxis aufweisen und häufig zum Einsatz kommen. Sekundärliteratur wurde im Rahmen dieser Arbeit explizit nicht untersucht, da diese Interpretationen beinhalten und somit die ursprünglichen Ideen der Autoren verändern könnten. Mit dem Ziel, einen Vergleich zwischen *traditionellen* und *agilen* Methoden zu ermöglichen, wurde sich für die Analyse folgender Primärliteratur entschieden:

- *Agile Software Development with Scrum*, Schwaber et al. [50]
- *Extreme Programming Explained*, Beck [3]
- *Das DevOps-Handbuch*, Gene et al. [24]
- *A spiral model of software development and enhancement*, Boehm [5]
- *Managing the development of large software systems: concepts and techniques*, Royce [47]

Im Rahmen der Literaturstudie wurden die folgenden Inklusions- und Exklusionskriterien definiert. Sie wurden auf jede Passage der oben genannten Primärliteratur angewendet. Erfüllte eine Passage mindestens ein Inklusions- aber kein Exklusionskriterium, wurde die Aussage als EZSPA gewertet.

Inklusionskriterien:

(i) Die Aussage beschreibt ein Recht des Entwicklers im Prozess.

(ii) Die Aussage beschreibt eine Eigenschaft des Prozesses, die dem Entwickler dient.

(iii) Die Aussage beschreibt eine Prozesseigenschaft, die Vorteile für den Entwickler bringt.

Exklusionskriterien:

(i) Die Aussage beschreibt eine Pflicht des Entwicklers im Prozess.

(ii) Die Aussage ist im Konjunktiv formuliert.

(iii) Die Aussage beschreibt eine methodische Prozesseigenschaft.

(iv) Die Aussage beschreibt Abweichungen vom regulären Prozess.

Insgesamt wurden 118 Abschnitte extrahiert, von denen sich im Laufe des Prozesses 35 als unpassend erwiesen haben. Die selektierten Textpassagen wurden von einer Forscherin mit mehrjähriger Erfahrung im Bereich der Softwareprozesse und dem Autor dieser Arbeit getrennt voneinander codiert. Hierbei handelte es sich um 83 Abschnitte, welche codiert wurden. Im nächsten Schritt wurden die entstandenen Codes verglichen. Bei einer Divergenz der Codierung wurde über die Auswahl des zugehörigen Codes diskutiert, bis ein gemeinsamer Konsens gefunden wurde. Eine solche Diskussion fand bei 45 Codes statt. Allerdings handelte es sich bei 38 Punkten lediglich um sprachliche Differenzen und nur bei sieben um inhaltliche Unterschiede. Im Folgenden ist der Codierungsprozess anhand eines Beispiels dargestellt.

Literaturzitat:

> *„There are other models of teamwork besides ‚every man for himself'. The team members can collectively assume responsibility not just for the quality of what they deliver to users but also for the pride they take in their work along the way.“* [3, S. 66]

Zugewiesener Code durch den Autor dieser Arbeit:

- Zusammenarbeit

Zugewiesener Code durch die Forscherin:

- Zusammenarbeit

Konsens:

- Zusammenarbeit

4.4 Interviewstudie

Im nächsten Schritt wurde eine Interviewstudie mit Softwareentwicklern durchgeführt, um praxisbezogene Erfahrungen zu erfassen. Hierfür ist eine ausführliche Planung und Dokumentation der Studie wichtig, um im Nachhinein die RQs beantworten und um eine Reproduzierbarkeit der Studie gewährleisten zu können [37]. Der Ablauf der Interviewstudie ist in Abbildung 4.3 dargestellt.

Zu Beginn werden die Rahmenbedingungen der Studie festgelegt. Anschließend wird beschrieben, wie der Interviewleitfaden entworfen wurde. Daraufhin wird die Zielgruppe definiert, die Teilnehmerakquise beschrieben und die Demografie der Probanden vorgestellt. Als Nächstes wird beschrieben, wie die Daten mittels Interviews erfasst wurden und abschließend wird geschildert, wie die gesammelten Daten zuerst codiert und anschließend mittels *thematischer Analyse* ausgewertet worden sind.

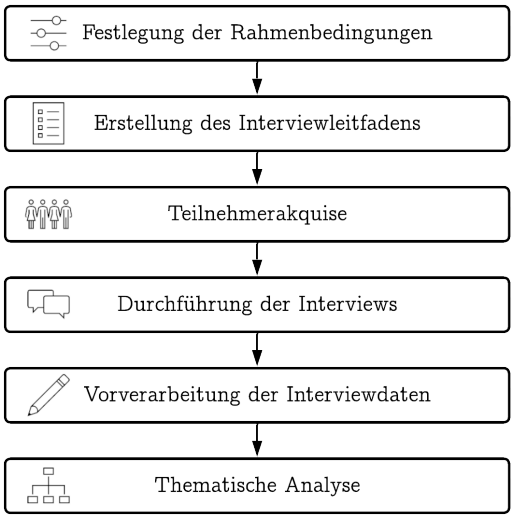

Abbildung 4.3: Ablauf der Interviewstudie

4.4.1 Rahmenbedingungen

Das Ziel quantitativer Forschung ist in vielen Fällen die Generalisierbarkeit gemessener Effekte [11]. Im Gegensatz dazu steht die qualitative Forschung, dessen Absicht es häufig ist, einen umfassenden Einblick in die Einstellungen und Erfahrungen von Menschen zu erlangen [11].

In dieser Arbeit wurde sich für einen qualitativen Ansatz mithilfe von Interviews entschieden, um einen tiefgehenden Eindruck der EZSPA in der Praxis zu erlangen. Mit dem Ziel, den Teilnehmenden größtmögliche Flexibilität zu ermöglichen, konnten sie entweder in Präsenz oder virtuell an der Studie teilzunehmen. Für die Durchführung von Interviews online wurde das Konferenztool BigBlueButton (BBB)[1] genutzt, welches im Browser lauffähig ist. Die Aufnahme der Tonspur wurde mithilfe von OBS[2], einem Programm zur Aufnahme von Ton- und Videodaten, realisiert.

Interviews lassen sich in verschiedene Kategorien einordnen. Das in dieser Studie verwendete sogenannte *semistrukturierte Interview* stellt eine Kombination aus unstrukturierten und strukturierten Interviews dar [9]. Die vorhandene Struktur eines semistrukturierten Interviews sorgt dafür, dass alle relevanten Fragen angesprochen werden können [45]. Dennoch dürfen auch Fragen gestellt werden, welche nicht im Vorfeld des Interviews festgelegt worden sind, um tiefergehende Einblicke zu erhalten [1].

[1] Zu finden auf: https://bigbluebutton.org/
[2] Zu finden auf: https://obsproject.com/de

4.4.2 Entwicklung des Interviewleitfadens

Der Leitfaden eines semistrukturierten Interviews soll den Interviewer dabei unterstützen, das Interview durchzuführen. Es werden Fragen definiert, welche im Laufe des Interviews auf jeden Fall gestellt werden sollen [45]. Zudem wird die Reihenfolge der Fragen definiert und, wie für ein semistrukturiertes Interview üblich, mögliche, optionale Rückfragen festgelegt [1]. Auf diese Weise wird eine Struktur des Interviews sichergestellt, welche eine Vergleichbarkeit der Interviews und ebenso eine vergleichende Auswertung zwischen einzelnen Interviews ermöglicht [45]. Wie bereits in Unterabschnitt 4.4.1 dargelegt, erlaubt ein semistrukturiertes Interview auch Nachfragen, um weitere Details vom Probanden zu erfahren [1].

Für diese Arbeit wurde die erste Version des Interviewleitfadens auf Basis der in Abschnitt 4.2 vorgestellten Forschungsfragen entworfen. Zudem wurden die Ergebnisse der in Abschnitt 4.3 beschriebenen Literaturstudie berücksichtigt, um einen Vergleich zwischen Theorie und Praxis zu ermöglichen. Außerdem wurden die Empfehlungen von Renner und Jacob [45] für die Erstellung des Interviewleitfadens berücksichtigt, wie etwa die Formulierung von offenen Fragen sowie potentiellen Rückfragen, um so viele Details wie möglich von den Probanden zu erfahren.

Der Interviewleitfaden wurde zuerst mit einem wissenschaftlichen Mitarbeiter des Fachgebiets für Software Engineering pilotiert. Hierbei wurde zum einen überprüft, ob das technische Setup und die Tonaufnahme mithilfe von OBS korrekt funktioniert. Zum anderen wurde kontrolliert, ob die Fragen von den Probanden wie gewünscht verstanden werden und von ihnen einfach zu beantworten sind. Es zeigte sich, dass einige Fragestellungen ohne weitere Hinweise nicht eindeutig zu verstehen waren. Nach einer Anpassung des Leitfadens wurde das Interview noch einmal mit zwei Bachelorstudenten der Informatik pilotiert, die im Software Engineering einen Interessenschwerpunkt haben. Diese Interviews haben gezeigt, dass die aktualisierte Version des Leitfadens sich zur Führung der Interviews als effektiv erwiesen hat. Die finale Version des Interviewleitfadens ist in Anhang A zu finden. Der Ablauf der Interviews ist in Abbildung 4.4 schematisch dargestellt.

Abbildung 4.4: Ablauf des Interviews

Zu Beginn eines Interviews gab es eine kurze Einführung, in welcher die Probanden begrüßt wurden. Zudem stellte der Leiter des Interviews sich und das Thema des Gesprächs vor. Als Nächstes wurden die Probanden gefragt, ob zu diesem Punkt bereits Unklarheiten aufgetreten sind. Im Anschluss wurde der Ablauf des Interviews vorgestellt, explizit auf die

Pseudoanonymität des Interviews hingewiesen und darum gebeten, ehrliche Antworten zu geben. Daraufhin wurde noch einmal auf die Einverständniserklärung der Probanden eingegangen. Bevor das eigentliche Interview startete, wurden die Probanden erneut gefragt, ob noch Unklarheiten bestehen würden. Abschließend wurden die Probanden gefragt, ob sie einer Aufnahme des Interviews zustimmen. Nach Erhalt der Zustimmung wurde die Aufzeichnung gestartet und die Teilnehmer wurden erneut um ihre Zustimmung zur Aufnahme gebeten.

Im ersten Teil des Interviews wurden den Probanden demografische Fragen gestellt, um nähere Informationen über sie und ihr Arbeitsumfeld zu erhalten. Die Probanden wurden beispielsweise gefragt, welche Rolle sie in der Softwareentwicklung innehaben und wie lange sie bereits in ihrem Feld arbeiten.

Im Anschluss an die demografischen Fragen folgte der Hauptteil des Interviews. Dieser wurde in zwei Teile aufgeteilt, um die verschiedenen Forschungsfragen beantworten zu können und trotzdem eine kohärente Struktur zu bewahren. Die erste Hälfte des Hauptteils drehte sich um die EZSPA, die der Prozess im Unternehmen des Probanden aufweist. Dabei wurden die EZSPA jedoch nur implizit erhoben, indem die Probanden gebeten wurden, den Prozess mit seinen Eigenschaften zu beschreiben. Mit diesen Fragen sollte eine Antwort auf die zweite RQ gefunden werden. In der zweiten Hälfte wurden die Probanden darüber befragt, welche Vorstellung von EZSPA sie haben und wie sie sich die Gestaltung dieser vorstellen würden. Mithilfe dieser Fragen sollte die dritte RQ beantwortet werden. In Tabelle 4.1 ist eine Auflistung der Interviewfragen dargestellt. Hier wird auch die Forschungsfrage aufgezeigt, welche mithilfe der jeweiligen Frage beantwortet werden soll. Nicht aufgelistet sind die demografischen Fragen zu Beginn des Interviews sowie mögliche Rückfragen auf die Aussagen der Probanden.

Zum Ende des Interviews wurden die Probanden gefragt, ob sie noch etwas auf der Aufnahme ergänzen wollen. Daraufhin wurde sich erkundigt, ob die Teilnehmer noch etwas äußern wollen, was nicht auf der Tonspur zu finden sein soll. Nachfolgend wurde gefragt, ob die Person noch weitere, potentielle Teilnehmer für die Studie kennt. Im Anschluss wurde auf die Verlosung eines 20€-Gutscheins hingewiesen. Letztlich wurde sich für die Teilnahme bedankt und im Anschluss wurden die Teilnehmer verabschiedet.

4.4.3 Teilnehmer

Mit dem Ziel, die in Abschnitt 4.2 definierten RQ mithilfe der geplanten Interviewstudie beantworten zu können, ist es von höchster Relevanz, von vornherein den möglichen Teilnehmerkreis der Studie genau festzulegen. Die Probanden sollten ausreichend Erfahrung im Themenbereich der Studie aufweisen, um die gestellten Fragen beantworten zu können [46].

Nr.	Frage	RQ
8.	Bitte beschreiben Sie mir so ausführlich wie möglich Ihren Alltag im Entwicklungsprozess.	2
9.	Was ist das vorherrschende Entwicklungs- bzw. Prozessframework?	2
10.	Wie sieht Ihr typischer Arbeitsalltag aus?	2
11.	Wie läuft ein typischer Sprint/Meilenstein/QG ab?	2
12.	Wie erhalten Sie Ihre Aufgaben?	2
13.	Wie wird darüber entschieden, woran als Nächstes gearbeitet wird?	2
14.	Wie erhalten Sie Feedback zu Ihrer Arbeit/ der Aufgabe?	2
15.	Wie wird Ihrer Erfahrung nach in Ihrem Unternehmen mit Fehlern umgegangen?	2
16.	Wie arbeiten Sie mit Teammitgliedern zusammen?	2
17.	Welche (regelmäßigen) Meetings haben Sie?	2
18.	Wie können Sie sich fort- und weiterbilden?	2
19.	Wie sieht Ihr Büro/Ihre Arbeitsumgebung aus?	2
20.	Welche Vorteile bietet der momentane Prozess für Sie als Entwickler?	2
21.	Was würden Sie sich zusätzlich wünschen, dass der Prozess für Sie als Entwickler tun sollte?	3
22.	Was für Arbeitsbedingungen wünschen Sie sich?	3
23.	Was sollte der Prozess in Bezug auf gute Zusammenarbeit für Sie ermöglichen?	3
24.	Wie sollte der Prozess eine klare Kommunikation ermöglichen?	3
25.	Welche Feedbackmechanismen würden Sie sich wünschen?	3
26.	Wie sollte der Prozess die Weitergabe von Wissen im Team unterstützen?	3
27.	Welche Freiheiten wünschen Sie sich in Ihrem Arbeitsalltag?	3

Tabelle 4.1: Die Interviewfragen des Hauptteils

Zielgruppe

Das Ziel der Studie ist die Aufnahme des gegenwärtigen Stands der EZSPA in der Praxis, sowie die Erfassung der Vorstellung dieser von den Studienteilnehmern. Aus diesem Grund steht die Perspektive von Softwareentwicklern im Vordergrund. Außerdem sollten die Teilnehmer zum Zeitpunkt des Interviews in einem Softwareprojekt beschäftigt sein, um vom Prozess nicht aus ihrer Erinnerung berichten zu müssen. Demografische Daten wie das Alter oder Geschlecht sind für diese Studie nicht relevant, da speziell die persönlichen Ansichten und Erfahrungen der Probanden von Interesse sind. Aus diesem Grund wurden sie nicht erfasst. Dennoch ist eine große Diversität der Probanden wünschenswert, um möglichst viele, facettenreiche Antworten zu erhalten. Das ist relevant, da es sich bei den EZSPA um einen relativ neuen Forschungsbereich handelt.

Teilnehmerakquise

Insgesamt nahmen 22 Personen an der Interviewstudie teil. Die Teilnehmerakquise fußte dabei in erster Linie auf der Ansprache von persönlichen Kontakten in Unternehmen, welche Software entwickeln. Viele der angesprochenen Personen konnten zwar nicht selbst an der Studie teilnehmen, allerdings wurde auf diesem Weg der Kontakt zu anderen Personen hergestellt, welche in der Softwareentwicklung tätig sind.

Mit dem Ziel, einen Anreiz zur Teilnahme zu schaffen, wurde unter allen Teilnehmern ein 20€-Gutschein des Onlineversandhändlers Amazon[3] verlost.

Im Rahmen der Teilnehmerakquise wurde das in Anhang B beigefügte Informationsblatt potentiellen Probanden zugeschickt. In diesem sind die Hintergründe und das Ziel der Studie beschrieben, sowie weitere, organisatorische Informationen bereitgestellt.

Vor dem Interview wurden die Teilnehmer gebeten, die in Anhang C befindliche Einverständniserklärung auszufüllen. Durch ihre Unterschrift zeigten die Probanden ihre Zustimmung zur Aufnahme des Interviews sowie zur Verwendung der dabei erfassten und im Anschluss pseudonymisierten Daten für die vorliegende wissenschaftliche Arbeit und eventuelle zukünftige Forschungsarbeiten. Zusätzlich wurden die Teilnehmer darüber informiert, dass sie ihre Zustimmung jederzeit und ohne Angabe von Gründen widerrufen können.

Teilnehmerdemografie

Die zu Beginn der Interviews aufgenommenen demografischen Daten werden im Folgenden vorgestellt. Dabei zieht sich die, den Probanden zugewiesene,

[3] Zu finden auf: https://www.amazon.de/

eindeutige ID durch alle Tabellen. In Tabelle 4.2 sind die demografischen Daten der Probanden zu finden.

ID	Rolle	Berufserf. (Rolle) [in Jahren]		Teamgröße	Homeoffice
1	Softwareentwickler	≤ 2	(≤ 1)	10	60%
2	Softwareentwickler	≤ 2	(≤ 2)	9	100%
3	IT-Engineer	≤ 1	(≤ 1)	7	100%
4	Softwareentwickler	≤ 1	(≤ 1)	15 - 20	40%
5	Softwareentwickler	≤ 5	(≤ 1)	5	20%
6	Projektleiter	> 10	(> 10)	5 - 6	100%
7	Softwareentwickler	≤ 5	(≤ 1)	3	100%
8	Softwareentwickler	≤ 1	(≤ 1)	15	90%
9	Softwareentwickler	≤ 2	(≤ 2)	13	50%
10	DevOps-Engineer	≤ 10	(≤ 1)	5	60%
11	Softwareentwickler	≤ 5	(≤ 1)	6	80%
12	Softwareentwickler	≤ 2	(≤ 2)	2	80%
13	Softwareentwickler	≤ 1	(≤ 1)	10	100%*
14	Softwareentwickler	≤ 2	(≤ 2)	6	60%
15	Softwareentwickler	≤ 2	(≤ 2)	10	80%
16	Softwareentwickler	≤ 5	(≤ 1)	10	50%
17	Softwareentwickler	≤ 2	(≤ 2)	7	100%
18	Softwareentwickler	≤ 2	(≤ 1)	5	100%*
19	Softwareentwickler	≤ 5	(≤ 5)	9	100%
20	Softwareentwickler	≤ 5	(≤ 5)	10	100%
21	Softwareentwickler	≤ 5	(≤ 5)	13	60%
22	Softwareentwickler	≤ 2	(≤ 2)	10	40%*

* Diese Teilnehmenden gaben an, das Homeofficeangebot nicht vollständig zu nutzen.

Tabelle 4.2: Die demografischen Daten der Probanden

Viele der Teilnehmer gaben spezifische Rollenbeschreibungen, wie beispielsweise *Fullstack-Entwickler*, an. Eine derartig feine Untergliederung ist im Kontext dieser Arbeit nicht relevant, daher werden entsprechend spezifische Rollenbeschreibungen zusammenfassend als *Softwareentwickler* codiert. Lediglich drei Probanden gaben an, dass ihre Rolle nicht der eines Softwareentwicklers entspreche. Neben ihrer Hauptaufgabe in verwandten Bereichen der Softwareentwicklung sind sie weiterhin direkt an der Softwareentwicklung beteiligt oder haben mehrjährige Erfahrung darin sammeln können. Ferner ist neben der aktuell ausgeführten Rolle auch die Berufserfahrung sowie die Erfahrung in der aktuellen Rolle angegeben. Dabei haben 20 der 22 Teilnehmer weniger als fünf Jahre Berufserfahrung. Zusätzlich ist die Größe des Teams angegeben, in welchem die Probanden ihre Arbeit verrichten. Jede Person, die an der Interviewstudie teilgenommen

ID	Mitarbeiter	Sektor
1	4.000	IT-Beratung
2	360.000	IT-Beratung
3	6	Werbeagentur
4	130	Versicherungsbranche
5	100 - 120	IT-Beratung
6	220	IT-Beratung
7	200 - 500	Logistik
8	500	Mobilität
9	300 - 400	Hausautomatisierung
10	200.000	Automobilindustrie
11	9.000	Automobilindustrie
12	1.000 - 2.000	Versicherungsbranche
13	400 - 500	Bahntechnik
14	200 - 220	Industrie
15	250 - 300	IT-Beratung
16	100	IT-Beratung
17	8	Finanzsektor
18	500	Mobilität
19	320.000	IT-Beratung
20	50	Enterprise Ressource Planning
21	1.000	Hausautomatisierung
22	60 - 65	Funk- und Sicherheitselektronik

Tabelle 4.3: Die demografischen Daten der Unternehmen

hat, gab an, im Homeoffice arbeiten zu können. Einige der Probanden gaben indes an, das Homeofficeangebot ihres Arbeitgebers nicht vollständig auszunutzen und häufiger als notwendig ins Büro zu kommen.

In Tabelle 4.3 sind die demografischen Daten der Unternehmen dargestellt. In der zweiten Spalte ist die von den Teilnehmern geschätzte Anzahl der Mitarbeiter im Unternehmen zu finden. Es wurden Interviews mit Personen aus Kleinstunternehmen bis hin zu Großunternehmen durchgeführt. Die Teilnehmer mit der ID 3 und 17 arbeiteten in Unternehmen mit weniger als 10 Mitarbeitern, während etwa der Proband mit der ID 2 in einem Großunternehmen mit ca. 360.000 Mitarbeitern mitwirkte. Des Weiteren wurde der Sektor angegeben, in welchem das Unternehmen agiert. Am häufigsten repräsentiert war der IT-Beratungssektor mit sieben Teilnehmern. Die Sektoren Automobilindustrie, Hausautomatisierung, Mobilität und Versicherungsbranche hatten jeweils zwei Teilnehmer. Alle sieben übriggebliebenen Teilnehmer kam aus jeweils unterschiedlichen Sektoren, wie beispielsweise dem Finanzsektor. Insgesamt ließen sich die 22 Interviewteilnehmer zwölf verschiedenen Sektoren zuordnen.

ID	Mitarbeiter	Dauer [in Jahren]	Kritikalität	Prozess
1	20	Wartungsprojekt	Fin. Schaden	Wasserfall
2	40 - 50	5	Unkritisch	Scrum
3	7	Keine Angabe	Fin. Schaden	Scrum
4	Keine Angabe	Produktentw.	Fin. Schaden	Scrum
5	10 - 11	1	Rep. Schaden	Lean Principles
6	10 - 15	Keine Angabe	Unkritisch	Scrum
7	3	Keine Angabe	Fin. Schaden	Agil
8	15	Produktentw.	Fin. Schaden	Scrum
9	30 - 40	1,5	Fin. Schaden	Keine Angabe
10	4 -5	Keine Angabe	Fin. Schaden	Scrum
11	6	Produktentw.	Unkritisch	Scrum
12	Keine Angabe	10	Fin. Schaden	Traditionell
13	150	3	Unkritisch	Scrum
14	6	Wartungsprojekt	Unkritisch	Agil
15	30 - 40	Wartungsprojekt	Fin. Schaden	Scrum
16	30 - 40	1	Unkritisch	Scrum
17	25	4	Unkritisch	Scrum
18	80 - 100	Produktentw.	Fin. Schaden	Scrum
19	50 - 70	4	Fin. Schaden	Scrum
20	2 - 3	3	Fin. Schaden	Scrum
21	20	0,66	Fin. Schaden	Scrum
22	20	10	Lebensgefahr	V-Modell

Tabelle 4.4: Die demografischen Daten der Projekte

In Tabelle 4.4 sind die demografischen Daten erfasst, welche die Probanden zu ihren Projekten angaben. In der zweiten Spalte ist die Anzahl der Mitarbeiter erfasst, welche laut Angaben der Teilnehmer am Projekt mitwirken. In der dritten Spalte ist die (geschätzte) Dauer des Projekts dargestellt. Dabei gaben einige Probanden an, dauerhaft an einem Produkt zu arbeiten, welches ihre Firma vertreibt. Diese sind mit „Produktentw." (Produktentwicklung) gekennzeichnet. Andere gaben an, an einem Wartungsprojekt zu arbeiten, welches ebenso nicht in den Rahmen eines klassischen Projektes fallen würde. Zudem ist die Kritikalität der Projekte festgestellt worden. Die meisten der Probanden sagten aus, dass bei einem Ausfall der entwickelten Software finanzielle Schäden für das Unternehmen entstehen würden. Andere sahen den Ausfall der im Projekt entwickelten Software als unkritisch, da es sich beispielsweise um ein *Proof of Concept* handeln würde. Lediglich eine Person sah Reputationsschäden für das Unternehmen im Falle eines Ausfalls der Software. Außerdem gab ein Proband an, dass Lebensgefahr für die Benutzer der Software

entstehen würde, sollte sie ausfallen. In der letzten Spalte dieser Tabelle ist der Prozess angegeben, nachdem, laut des Probanden, im Projekt gearbeitet werden würde. Die Mehrheit der Teilnehmer gab hierbei an, nach einer Form von Scrum zu arbeiten. Drei Probanden sagten, sie würden einen eher traditionellen Prozess nutzen, wie etwa das V-Modell oder das Wasserfallmodell. Lediglich eine Person konnte keinen Namen für den Prozess finden, gab aber an, nach Meilensteinen zu arbeiten.

4.4.4 Datenerfassung

Die Interviews wurden im November und Dezember 2023 unter Zuhilfenahme des im Vorfeld entworfenen Interviewleitfadens durchgeführt. Dabei konnten die Teilnehmenden frei entscheiden, ob sie online oder in Präsenz an der Studie teilnehmen wollten. 21 der 22 Probanden haben sich für eine Teilnahme online, unter Verwendung von BBB, entschieden. Lediglich eine Person führte das Interview in Präsenz durch.

Bei einem Probanden traten während des Interviews Verbindungsprobleme zu BBB auf. Diese konnten allerdings gelöst werden, sodass das Interview ordnungsgemäß zu Ende geführt werden konnte. Bei einem weiteren Probanden traten technische Probleme auf, welche dafür sorgten, dass das Interview schneller zu Ende geführt werden musste. Ein dritter Proband musste das Interview zwischenzeitlich unterbrechen. Es konnte jedoch nach einer kurzen Pause zu Ende geführt werden.

4.4.5 Datenanalyse

Im Folgenden wird beschrieben, wie die erhobenen Daten ausgewertet wurden. Zu Beginn fand eine Vorverarbeitung der in den Interviews aufgenommenen Daten statt. Auf dieser Grundlage konnte im nächsten Schritt eine thematische Analyse durchgeführt werden.

Vorverarbeitung der Daten

Die im Rahmen der Interviews aufgenommenen Audiodaten wurden im ersten Schritt mithilfe von dem neuronalen Netz *Whisper*[4] [44] von *OpenAI*[5] transkribiert. In dieser Arbeit wurde eine vollständige Transkription des Gesprochenen durchgeführt, wobei lediglich Fülllaute wie beispielsweise „Ähm“ nicht transkribiert wurden, um eine gute Lesbarkeit des Transkripts sicherzustellen [33]. Im Anschluss an die automatisch durchgeführte Transkription wurden die generierten Transkripte manuell mit der Tonspur der Interviews verglichen, um eventuell noch vorhandene Transkriptionsfehler zu korrigieren. Dies geschah mit dem Ziel, eine hohe Qualität der Transkriptionen

[4]Zu finden auf: https://openai.com/research/whisper
[5]Zu finden auf: https://openai.com/

sicherzustellen, da diese die Grundlage für die weitere Analyse darstellen und von ihnen die Zuverlässigkeit der Ergebnisse direkt abhängt [33]. Während dieses Vorgangs wurden identifizierende Daten zudem pseudonymisiert, um einen Rückschluss von Dritten auf die interviewte Person unmöglich zu machen. Hatte der Proband beispielsweise einen Firmennamen genannt, wurde dieser im Transkript mit „[Unternehmen]“ ersetzt.

In den Transkripten finden sich zusätzliche Metainformationen zu den jeweiligen darin beschriebenen Interviews. Dazu gehört die ID des Probanden, die ihm zur Unterscheidung zwischen einzelnen Interviews zugeordnet worden ist, sowie das Datum, an dem das Interview stattgefunden hat. Außerdem ist vermerkt, ob das Interview online oder in Präsenz stattgefunden hat, wer das Interview geführt und wer es transkribiert hat. Das Schriftbild der Transkripte wurde einheitlich und nach den Richtlinien von McLellan et al. [31] gestaltet. Bei einem Sprecherwechsel wurde ein neuer Absatz eingefügt und der Beginn mit dem Kürzel des Sprechers versehen. Hierbei steht ein „I“ für den Interviewer und ein „P“ für den Probanden. Zudem wurde nach den Sprecherkürzeln eine Nummerierung für den Absatz eingefügt, um bei der Analyse Aussagen im Transkript schneller wiederfinden zu können.

Thematische Analyse

Ursprünglich war eine Auswertung der Interviewstudie mittels *Grounded Theory* geplant. Dabei handelt es sich um eine qualitative Forschungstechnik, mit der *Theorien* formuliert werden können [14, 39], welche auf den gesammelten empirischen Daten basieren [7]. Dabei bringt diese Form der Auswertung einige Vorteile mit sich. So bietet Grounded Theory dem Forscher eine systematische Herangehensweise an die Datenanalyse und kann auf diese Weise für tiefgreifende und reichhaltige Ergebnisse sorgen [20]. Das Ziel dieser Arbeit ist es allerdings nicht, eine Theorie aus den gewonnenen Daten zu erstellen. Es sollen die EZSPA in der Praxis erfasst und mit der Theorie verglichen werden, wofür eine *thematische Analyse* besser geeignet ist.

Bei der thematischen Analyse handelt es sich um eine Methode für die Identifizierung, Analyse und Interpretation von Mustern, sogenannten *Themen*, in qualitativen Daten [12]. Dabei beschreiben Braun und Clarke in ihrer Arbeit [7] eine Herangehensweise mit sechs Schritten, welche in Abbildung 4.5 dargestellt ist und im Folgenden beschrieben wird.

Zu Beginn ist es von höchster Wichtigkeit, sich mit den vorliegenden Daten vertraut zu machen. Es empfiehlt sich, die Daten mindestens einmal vollständig gelesen zu haben, damit bereits zu diesem Zeitpunkt Ideen für potentielle Themen entstehen können. Die Autoren schreiben, dass der Prozess der Transkription hierfür bereits eine sehr gute Gelegenheit bietet [7].

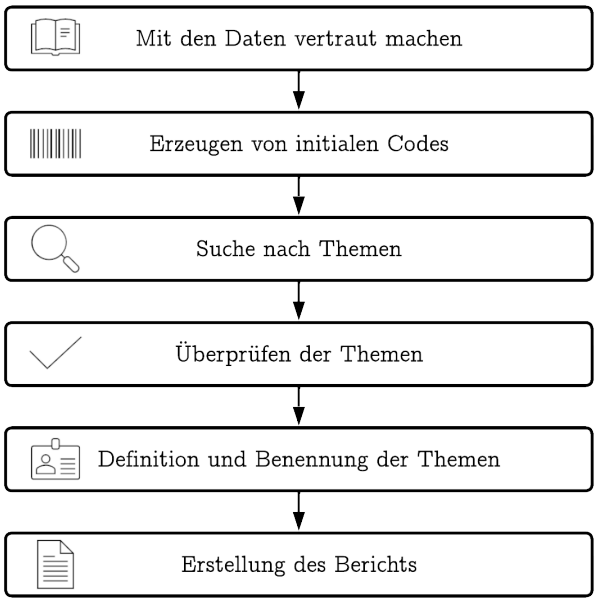

Abbildung 4.5: Ablauf einer thematischen Analyse nach [7]

Im nächsten Schritt folgt die initiale Codierung der Daten. Es werden Merkmale der Daten identifiziert, welche für die Arbeit interessant sein könnten. In dieser Phase findet noch keine Interpretation der Äußerungen statt. In dieser Arbeit wird die *offene Codierung* (*in vivo* [30]) genutzt, bei der Kernaussagen in wenigen Worten als Code zusammengefasst werden [23].

Anschließend folgt die Suche nach Themen in den vorhandenen Codes. In diesem Schritt wird damit begonnen, Codes zu abstrahieren und zu potentiellen Subthemen zusammenzufassen. Im Anschluss werden diese Subthemen wiederum zu Themen zusammengefasst.

Im vierten Schritt werden die bereits definierten Themen noch einmal überprüft. Themen ohne hinreichende Datenbasis werden entfernt und ursprünglich unterschiedliche Themen werden zusammengeführt, wenn sie sich inhaltlich entsprechen. Außerdem ist es möglich, Themen aufzubrechen und aus einem mehrere zu generieren, solange die Datenbasis das zulässt. Dabei ist es wichtig darauf zu achten, dass die Daten innerhalb eines Themas in einem sinnvollen Zusammenhang zueinander stehen, während zwischen den Themen klare und erkennbare Unterschiede bestehen sollten.

In der fünften Phase werden die finalen Themen verfeinert, definiert und benannt. Hierbei soll die *Essenz* eines jeden Themas erfasst werden. Themen sollten weder zu komplex, noch inhaltlich zu weitreichend sein. Sie sollten ineinander kohärent sein und eine „Geschichte“ in Bezug auf die Forschungsfragen erzählen, sowohl innerhalb eines Themas als auch themenübergreifend.

Den Abschluss der thematischen Analyse bildet die Erstellung des Berichts über die Auswertung.

Braun und Clarke [7] geben außerdem einige Fragestellungen zu bedenken, welche im Vorfeld einer thematischen Analyse betrachtet werden sollten. Diese sollen im Folgenden beantwortet werden:

- **Was gilt als Thema?**
  Ein Thema wird in dieser Arbeit erfasst, wenn es für die Beantwortung der Forschungsfragen als wichtig erscheint und von mehreren Probanden genannt wird.

- **Eine ausführliche Beschreibung des Datensatzes oder ein detaillierter Bericht über einen bestimmten Aspekt?**
  Mit dem Ziel einen Vergleich zwischen Theorie und Praxis der EZSPA zu schaffen, wird der Datensatz in erster Linie ausführlich beschrieben.

- **Induktive oder theoretische thematische Analyse?**
  In dieser Arbeit wird eine *theoretische thematische Analyse* durchgeführt, da ein analytisches Interesse am Forschungsbereich besteht und die Forschungsfragen bereits im Vorfeld definiert worden sind.

- **Semantische oder latente Themen?**
  Codes und Themen werden für diese Arbeit primär auf semantischer Ebene erfasst. Das bedeutet, dass Aussagen von Probanden nicht interpretiert, sondern in Themen organisiert und zusammengefasst werden. Eine Interpretation findet im Anschluss in Form einer Einschätzung der Bedeutung der gefundenen Muster und ihrer Implikationen statt.

- **Realistische oder konstruktivistische thematische Analyse?**
  In einer *realistischen thematischen Analyse* sollen die Daten so genau wie möglich und unvoreingenommen analysiert werden [7]. Auf diese Weise sollen objektiv Muster in den Daten identifiziert werden. In einer *konstruktivistischen thematischen Analyse* sollen die vorhandenen Daten subjektiv interpretiert werden, um soziale und kulturelle Hintergründe berücksichtigen zu können [7]. In dieser Arbeit wird eine realistische thematische Analyse durchgeführt, um auf diese Weise einen objektiven Überblick über die EZSPA schaffen zu können.

4.5 Vergleich der Ergebnisse

Nachdem die benötigen Daten für einen Vergleich zwischen Theorie und Praxis erfasst worden sind, kann dieser im nächsten Schritt stattfinden. In dieser Arbeit findet er mithilfe der Themen und Codes statt, welche in den beiden Studienteilen gesammelt worden sind. Es werden die

unterschiedlichen Themen analysiert und gegenübergestellt. Dazu wird überprüft, ob und inwiefern die Ausführungen der Interviewprobanden mit den Beschreibungen in der Literatur übereinstimmen. In der Interviewstudie wird ein Attribut als gegeben betrachtet, wenn die Mehrheit der Teilnehmer es oder damit verbundene Aspekte positiv oder als erfüllt bewertet hat. Wenn unterschiedliche Antworten vorliegen und einige Teilnehmer ein Merkmal als gegeben betrachten, während andere dies nicht tun, wird dieser Aspekt gesondert markiert.

Kapitel 5

Ergebnisse

In diesem Kapitel werden die Ergebnisse beider Studienteile vorgestellt. Zu Beginn werden die Ergebnisse der Literaturstudie beschrieben. Daraufhin werden die Ergebnisse der Interviewstudie dargestellt und erläutert. Abschließend werden die Ergebnisse beider Studien miteinander verglichen, wie Abbildung 5.1 schematisch zeigt.

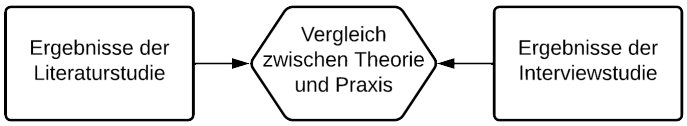

Abbildung 5.1: Eine schematische Darstellung des Vorgehens in der Studie

5.1 Ergebnisse der Literaturstudie

In Tabelle 5.1 ist ein Überblick über die Ergebnisse der Literaturstudie abgebildet. Diese werden im Folgenden erläutert. Sie sind für eine bessere Lesbarkeit der Daten in Kategorien strukturiert, ohne eine weiterführende Interpretation darzustellen. Die insgesamt 29 erfassten übergeordneten Codes befanden sich in 83 Passagen der analysierten Primärliteratur. Einer einzelnen Passage wurden dabei in vielen Fällen mehrere unterschiedliche Codes zugeordnet. Insgesamt ergeben sich 57 Codes für *„Extreme Programming Explained“* [3], 51 für *„Das DevOps-Handbuch“* [24] und 28 für *„Agile Software Development with Scrum“* [50]. Weder in *„A Spiral Model of Software Development and Enhancement“* [5] noch in *„Managing the Development of large Software Systems: Concepts and Techniques“* [47] konnten den in Abschnitt 4.3 definierten Inklusions- und Exklusionskriterien genügende Passagen gefunden werden. Eine genaue Auflistung aller Codes mit der dazugehörigen Anzahl an Befunden und der jeweiligen Quelle ist in Anhang D zu finden.

Weiterbildung	Team	Prozess	Persönliche Aspekte
Weiterentwicklung	Zusammenarbeit	Entscheidungsfreiheit	Persönliche Erfüllung
Feedback	Transparenz	Eigene Zieldefinition	Gute Arbeitsbedingungen
Weiterbildung	Unterstützung	Selbstorganisiertes Team	
Kontinuierliche Verbesserung	Klare Kommunikation	Wenig Teamabhängigkeiten	
Wissenstransfer	Interesse an den Anderen	Entscheidungsbefugnis	
Mentoring	Respekt	Optimaler Informationsfluss	
Fortbildung	Wertschätzung	Gute Arbeitsbedingungen	
Unternehmensweites Lernen	Crossfunktionale Teams		
Kontinuierliches Lernen	Umgang		
	Fehlerkultur		
	Mitarbeiterkontinuität		
	Gute Arbeitsbedingungen		

Tabelle 5.1: Überblick über die Codes, welche durch die Literaturstudie entstanden sind

5.1.1 Weiterbildung

In der Literatur [3, 24, 50] spielt Weiterbildung insgesamt eine bedeutende Rolle. Unternehmen können die Weiterbildung und -entwicklung von Mitarbeitern unterstützen, indem sie ihnen die Möglichkeit bieten, zusammen neue Dinge zu lernen und auszuprobieren. XP beschreibt diesen Aspekt beispielsweise folgendermaßen:

> *„Communities can also be a place to study together. XP includes many skills that improve with practice.“* [3, S. 158]

Auch Feedback über die getane Arbeit ist von Bedeutung, um als Entwickler zu wachsen. Dadurch wissen diese, an welchen Punkten noch Verbesserungsbedarf besteht und können so gezielt daran arbeiten:

> *„Statt Projektteams zu bilden, in denen die Entwickler nach jedem Release neu zugewiesen und verschoben werden und dabei niemals Feedback zu ihrer Arbeit erhalten, lassen wir die Teams intakt, sodass sie weiter iterieren und sich verbessern können und dabei die gewonnene Erfahrung nutzen, um ihre Ziele besser zu erreichen.“* [24, S. 34]

Den Willen, eine kontinuierliche Verbesserung zu zeigen, aber dennoch mit seiner Arbeit zufrieden zu sein, ist insbesondere für XP wichtig:

> *„XP is fully appreciating yourself for total effort today. XP is striving to do better tomorrow.“* [3, S. 6]

Dazu gehört auch, Fehler nicht zu verstecken, sondern das eigene Vorgehen zu analysieren und zu evaluieren, um auf diesem Weg besser zu werden.

XP stellt zudem den Wissenstransfer im Team in den Mittelpunkt. Dieser kann beispielsweise durch *Pair Programming* geschehen:

> *„By mostly keeping teams together and yet encouraging a reasonable amount of rotation, the organization gets the benefits of both stable teams and consistently spread knowledge and experience.“* [3, S. 64]

Scrum schlägt auch explizit einen Wissenstransfer durch Mentoring vor. Personen, die sich in einem Bereich sehr gut auskennen, können ihr Wissen im Team teilen und es weiterbilden.

Fortbildung in Scrum und ein unternehmensweites Lernen in DevOps stellen weitere wichtige Aspekte der Weiterbildung dar. Das Team muss herausfinden, welche Ideen funktionieren und welche nicht. Im Anschluss kann sich das Team auf die Erfolgreichen fokussieren und diese in einem größeren Rahmen umsetzen:

> *„Jeder lokale Wissensgewinn wird schnell in globale Verbesserungen umgesetzt, sodass sich die gesamte Firma neue Techniken und Praktiken, die die Technologie-Wertkette in einem Bereich verbessern, zunutze machen kann.“* [24, S. 84]

Hinzu kommt im DevOps-Prozess noch ein kontinuierliches Lernen der Mitarbeiter. Unter Verwendung von wissenschaftlichen Methoden und einer

von Hypothesen getriebenen Kultur wird bei DevOps sichergestellt, dass kein Wissen als gegeben betrachtet wird. Außerdem ermöglichen interne Fortbildungen vorhandenes Wissen zu teilen und jeder Mitarbeiter bekommt die Möglichkeit, Neues kennenzulernen. Mithilfe dieser Aspekte kann eine angenehme Arbeitsumgebung geschaffen werden:

> *„Kontinuierliches Lernen und Experimentieren verbessern aber nicht nur die Performance unserer Systeme. Diese Praktiken schaffen auch eine inspirierende und lohnenswerte Arbeitsumgebung, in der wir gern tätig sind und mit unseren Kolleginnen und Kollegen zusammenarbeiten.“* [24, S. 84]

5.1.2 Team

Teambezogene Aspekte nehmen den größten Raum in der Literatur [3, 24, 50] ein. Insbesondere die Zusammenarbeit wurde aus vielen Perspektiven und in allen drei Vorgehensmodellen beleuchtet. Der Scrum-Prozess sieht vor, dass jedes Teammitglied weiß, woran die anderen arbeiten, um somit Möglichkeiten zur Zusammenarbeit und zum Mentoring zu schaffen. Für XP sind sichere soziale Interaktionen genauso wichtig für die Entwicklung wie technische Fähigkeiten:

> *„Good, safe social interaction is as necessary to successful XP development as good technical skills.“* [3, S. 4]

XP geht davon aus, dass das Team *eine Einheit* ist, zusammenarbeitet und sich gegenseitig unterstützt. Daher ist neben der Zusammenarbeit auch die gegenseitige Unterstützung in allen Prozessen zentral:

> *„Community is important because everyone needs support sometimes.“* [3, S. 157]

Zu dieser Unterstützung zählt auch eine klare Kommunikation im Team. Für viele auftretende Probleme kennt ein Teammitglied bereits die Lösung. Daher ist es von Bedeutung sich mit seinem Team auszutauschen, um durch die gegenseitige Hilfe schneller die passende Lösung zu finden. Gegenseitige Unterstützung stärkt in langfristigen Teams zudem die Weiterbildung, da die Mitglieder voneinander lernen.

Weiter beschreibt XP die Wichtigkeit von Transparenz über die Projektentwicklung. Auf diese Weise verursachen Deadlines weniger Stress, da jede im Projekt involvierte Person zu jedem Zeitpunkt über den aktuellen Stand Bescheid weiß und es so keine unerwarteten Ereignisse gibt.

Im Rahmen von XP ist eine klare, häufige und zielgeführte Kommunikation ein Zeichen von einem produktiven Team:

> *„People evaluating XP teams should understand what an effective team looks like. This may differ from other teams they have seen. For example, talking and working go together on an XP team. The hum*

> *of conversation is a sign of health. Silence is the sound of risk piling up.“* [3, S. 79]

Zudem hat sie das Potential, Spannungen unter Teammitgliedern zu vermindern. Des Weiteren ist in XP gegenseitiger Respekt, Interesse an den eigenen Teammitgliedern und eine Wertschätzung im Umgang mit diesen von hoher Relevanz:

> *„In XP, valuable employees: Act respectful, play well with others, [...]“* [3, S. 82]

Jede Person ist gleich viel wert und kann relevante Ideen zur Entwicklung beisteuern. Hinzu kommt, dass Personen sich selber weiterentwickeln, um sich den ihnen entgegengebrachten Respekt auch zu verdienen.

Einen weiteren Aspekt für DevOps und XP stellt die Fehlerkultur im Unternehmen dar:

> *„Hiding errors to protect yourself, while sometimes seemingly necessary, is a tremendous waste of time and energy. Trust energizes participants. We feel good when things work smoothly. We need to be safe to experiment and make mistakes.“* [3, S. 98]

Es ist wichtig, Mitarbeitern ein Gefühl von Sicherheit und Vertrauen zu vermitteln. Auf diese Weise müssen sie geschehene Fehler nicht verstecken, sondern können diese offen kommunizieren und an einer Lösung arbeiten. Somit werden möglichst viele Erkenntnisse aus einem Vorfall gewonnen und keine Schuldigen gesucht. Dadurch kann die Qualität und Sicherheit des Systems verbessert und die Beziehungen der Entwickler untereinander verstärkt werden.

Hinzu kommen verschiedene Ausprägungen guter Arbeitsbedingungen, persönliche aber auch teambezogene. Zum einen strebt DevOps eine Kontinuität der Mitarbeiter an. Hierdurch können Teams sich durch erhaltenes Feedback iterativ verbessern und ihre Zusammenarbeit und das Vertrauen untereinander stärken. Zum anderen ist es wichtig, Aufstiegsmöglichkeiten zu bieten. Mitarbeiter mit einem eigenen Interesse an Weiterbildung sollen uneingeschränkte Unterstützung dabei erhalten und daran wachsen:

> *„Dadurch dass wir jeden beim Lernen unterstützen, [...], sorgen wir für einen sehr nachhaltigen und günstigen Weg, unseren Teams zu Großem zu verhelfen - weil wir in die Entwicklung der Menschen investieren, die wir schon haben.“* [24, S. 145]

5.1.3 Prozess

Die Lehrbücher [3, 24, 50] beschreiben verschiedene Aspekte des Prozesses, welche den Entwicklern dienlich sein sollen. Zu Beginn steht hier die Entscheidungsfreiheit, welche in Scrum von größter Wichtigkeit ist, aber auch in XP eine Rolle spielt. Im Scrum-Prozess haben Entwickler die volle Entscheidungsfreiheit, um ihr gesetztes Ziel zu erreichen:

> *„During a Sprint, no one external to the team tells the team what to do.“* [50, S. 147]

Das Team kann entscheiden, auf welche Weise sie ihre Aufgaben bearbeiten und niemand kann ihnen ihre Arbeitsweise vorschreiben.

Auch die eigene Zieldefinition durch das Team ist in beiden Vorgehensmodellen wichtig. Auf Basis dessen können die Entwickler selbstständig einschätzen, welche Aufgaben sie in welchem Zeitrahmen erledigen können. Sie sind weniger frustriert, da es seltener vorkommt, dass sie Aufgaben nicht schaffen.

Einen weiteren Aspekt für Scrum stellt das selbstorganisierte, crossfunktionale Team dar:

> *„The team self-organizes not only at the beginning of a Sprint, but continually as work progresses throughout the Sprint. Since it is the team as a whole that commits to the Sprint goal, the team is going to sink or swim together.“* [50, S. 117]

In Scrum gibt es keine Rollenbeschreibungen. Das Team entscheidet selbstständig, welches Mitglied welche Aufgabe übernimmt und die Teammitglieder arbeiten kontinuierlich zusammen, um ihr Ziel zu erreichen. Allerdings müssen sie sich dabei an existierende Standards und Technologien halten. Sie teilen ihre Arbeitszeiten zudem frei ein, sofern sie nicht mit den Vorgaben des Unternehmens kollidieren.

Im Rahmen von DevOps ist es relevant, dass möglichst wenig Abhängigkeiten zwischen einzelnen Teams existieren. Auf diese Weise arbeiten Entwickler eigenständig und unabhängig voneinander und müssen nicht auf Ergebnisse anderer Teams warten. Nach DevOps wird so die Produktivität erhöht und der Kunde erhält häufiger neue Features. Des Weiteren sorgt dieser Vorgang auch für ein unternehmensweites Lernen und die Zufriedenheit der Mitarbeiter steigt dadurch an:

> *„So [Durch unabhängige Teams] maximieren Firmen die Produktivität der Entwickler, ermöglichen ein unternehmensweites Lernen, sorgen für eine hohe Mitarbeiterzufriedenheit und sind im Markt erfolgreich.“* [24, S. 25]

Im DevOps-Prozess spielt auch die Entscheidungsbefugnis einzelner Mitarbeiter eine Rolle. Wenn sie die für ihren Arbeitsalltag notwendigen Entscheidungen selbstständig treffen können, ohne einen Vorgesetzten zu involvieren, werden Aufgaben schneller und mit weniger Frust erledigt.

Außerdem ist es wichtig, Teamgrößen im DevOps-Prozess klein zu halten, um einen optimalen Informationsfluss zu gewährleisten. Hierdurch wird die Kommunikation auf einem Minimum gehalten und das Team hat jederzeit ein klares und gemeinsames Verständnis von dem System, welches es entwickelt. Auf diesem Weg wird den Teams eine autonome Arbeitsweise ermöglicht, da sie kaum mit anderen interagieren müssen.

Hinzu kommen gute Arbeitsbedingungen, welche durch den Prozess ermöglicht werden. XP verspricht seinen Entwicklern, dass sie keine geteilten Aufgaben erhalten. Das bedeutet, dass sie ausschließlich an einem Projekt arbeiten und sich daher vollständig auf dieses konzentrieren können. Wenn Menschen konstant zusammenarbeiten, entwickelt sich zudem ein Teamgefühl:

> *„Some organizations try tro have teams with fractional people [...]. In this case, so much time is wasted on task-switching that you can see immediate improvement by grouping the programmers into teams. [...] This frees the programmers from fractured thinking. [...] People need acceptance and belonging. [...], without having other programmers to identify with, destroys the sense of „team“ and is counterproductive.“* [3, S. 39]

Im DevOps-Prozess ist es die Aufgabe von Vorgesetzten, Bedingungen zu schaffen, unter welchen die Entwickler ihr volles Potential entfalten können. Daher ist es wichtig, dass Vorgesetzte ihre Entwickler bestmöglich unterstützen, um gute Arbeitsbedingungen für sie zu schaffen. Außerdem dürfen Entwickler nicht dafür bestraft werden, Risiken einzugehen:

> *„Statt einer Kultur der Angst bilden wir eine Kultur des Vertrauens und der Zusammenarbeit, bei der Mitarbeiter dafür belohnt werden, dass sie Risiken eingehen.“* [24, S. 35]

5.1.4 Persönliche Aspekte

Es wurden zwei Facetten persönlicher Aspekte in der Literatur [3, 24, 50] identifiziert. Zum einen gibt XP an, dem Entwickler in einem gewissen Rahmen eine persönliche Erfüllung zu bieten:

> *„XP is fully appreciating yourself for total effort today. XP is striving to do better tomorrow. XP is evaluating yourself by your contribution to the team's shared goals. XP is asking to get some of your human needs met through software development.“* [3, S. 6]

Zum anderen fallen hierunter gute Arbeitsbedingungen für die involvierten Entwickler. XP beschreibt hier beispielsweise den Arbeitsschutz im Rahmen einer nachhaltigen Arbeitsweise. Dazu gehört auch die Negierung von Überstunden und Wochenendarbeit. Beides sei auf längere Zeit nicht verkraftbar. DevOps gibt an, für weniger stressbehaftete Releasephasen zu sorgen. Des Weiteren gehört zu den persönlichen Aspekten auch die Work-Life-Balance, welche im Rahmen von DevOps und XP genannt wird:

> *„Statt mit den Deployments freitags um Mitternacht zu beginnen und das ganze Wochenende damit zu verbringen, sie über die Bühne zu bringen, werden sie an normalen Arbeitstagen zu den normalen Arbeitszeiten ausgeführt, wenn sowieso jeder im Büro ist.“* [24, S. 33]

XP legt zudem Wert auf die Gesundheit und das Wohlbefinden der Entwickler. Personen sollen aus Respekt gegenüber dem eigenen Team Zuhause bleiben, wenn sie krank sind und anschließend erholt zurückkommen. Außerdem muss den Entwicklern genug Raum für Privatsphäre geschaffen werden, um diesen die Möglichkeit zu geben, sich bei Bedarf zurückzuziehen.

Sicherheit und Verständnis im Team spielen für XP auch eine zentrale Rolle. Diese sind für eine gute Kommunikation im Team sehr wichtig:

> *„For open, honest communication to take place in a community, the participants must feel safe and understood.“* [3, S. 157]

Allerdings ist ein Gefühl von Sicherheit auch im Rahmen von DevOps von Relevanz. Es ist wichtig, den Mitarbeitern zu vermitteln, dass sie angstfrei über Probleme sprechen können, um diese überhaupt sehen und auch lösen zu können. Hinzu kommt, dass Mitarbeiter auch bei eigenen Fehlern nicht schweigen, sondern diese ansprechen sollen.

Außerdem ermöglichen DevOps und Scrum ein fokussiertes, störungsfreies Arbeiten:

> *„Scrum lets a team set up problems and provides them an environment in which it can focus.“* [50, S. 39]

Insbesondere DevOps beschreibt, dass Entwickler ausschließlich an einem Projekt arbeiten und keine unnötige Kommunikation führen sollen. Auf diese Weise können Entwickler sich ungestört auf eine Aufgabe konzentrieren und sie bestmöglich bearbeiten.

5.2 Ergebnisse der Interviewstudie

Im Folgenden werden die Ergebnisse der Interviewstudie präsentiert. In Abbildung 5.2 ist ein Überblick der erfassten Themen dargestellt. Die Themen und ihre zugehörigen Subthemen werden in den einzelnen Unterkapiteln näher beleuchtet. Dabei wurden zehn Themen mit insgesamt 31 Subthemen identifiziert.

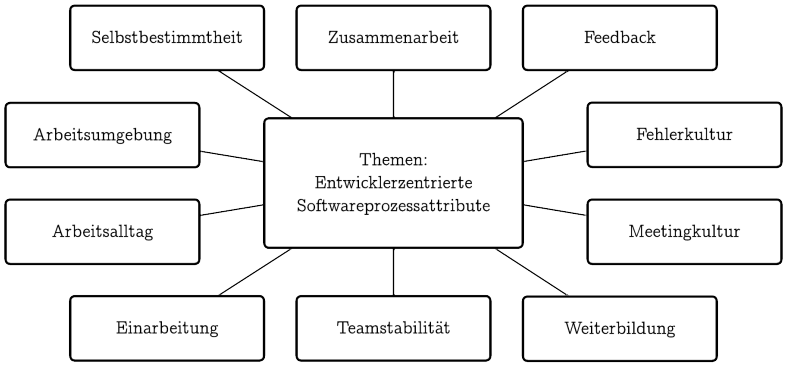

Abbildung 5.2: Die mittels thematischer Analyse erfassten Themen

In Abbildung 5.3 ist die Anzahl der den Themen zugeordneten Codes sowie ihre prozentuale Verteilung dargestellt.

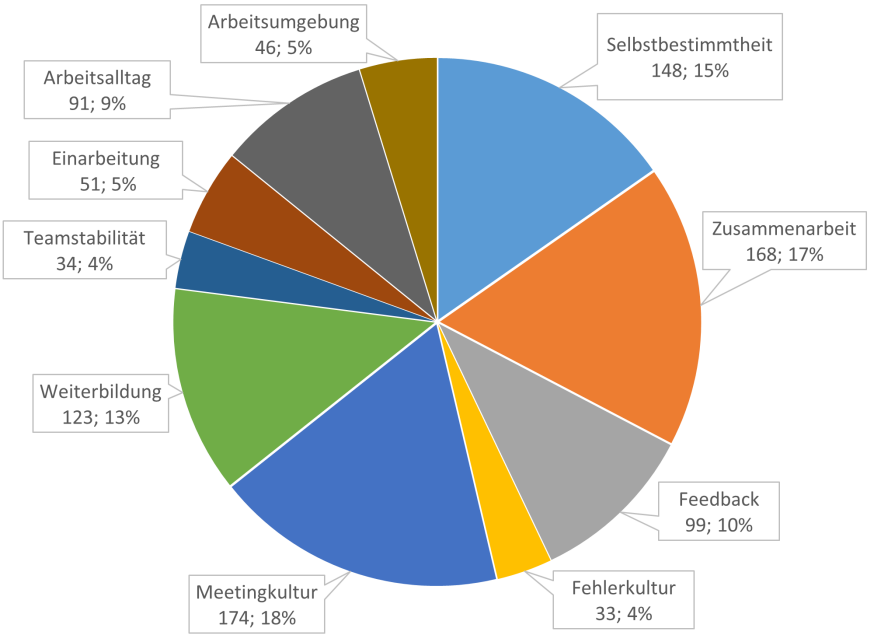

Abbildung 5.3: Die Aufteilung der gefundenen Codes auf die Themen

5.2.1 Selbstbestimmtheit

Viele Probanden gaben an, ihren Arbeitsalltag in einem gewissen Rahmen selbstbestimmt gestalten zu können. Diese Selbstbestimmtheit äußerte sich in verschiedenen Aspekten, welche in Abbildung 5.4 dargestellt sind.

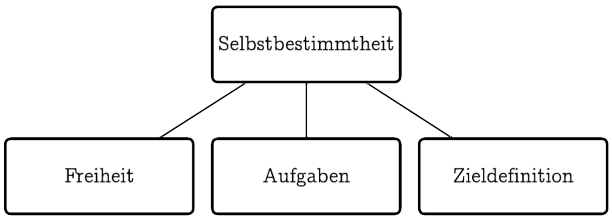

Abbildung 5.4: Die Subthemen des Themas Selbstbestimmtheit

Freiheit Eine wichtige Facette der Freiheit im Arbeitsalltag ist das selbstbestimmte Arbeiten. Elf Probanden sehen hier persönliche Vorteile durch den Prozess, welche sich wiederum in unterschiedlichen Faktoren ausdrücken.

Zehn Personen sehen Vorteile in der Flexibilität, die ihnen die Arbeit bietet. Sie ermöglicht es, die Aufgaben unabhängig und zeitlich flexibel zu gestalten und dennoch regelmäßig Feedback zu erhalten. Zudem ermöglicht die Flexibilität eine freie Aufgabenauswahl und die Wahrnehmung privater Termine während der Arbeitszeit. Homeoffice ist ein weiterer Aspekt der Flexibilität:

> *„Ja, also das Homeoffice arbeiten natürlich, ist in Verbindung halt sehr flexibel [...]. Wir haben auch Gleitzeit [...]. Das sind so die persönlichen Vorteile, dass ich halt meine Arbeit eigentlich so gestalten kann, wie ich möchte. Es gibt natürlich immer so ein paar restriktivere Sachen, wenn du halt eben zu bestimmten Zeiten mit den Teamkollegen kommunizieren musst, [...] ansonsten ist das relativ flexibel.“* (Interview 20, Absatz 70)

Fünf Probanden sprachen außerdem positiv von der Stressfreiheit ihrer Arbeit:

> *„Also bis jetzt habe ich damit so keinen Zeitdruck. Also die Arbeit soll erledigt [werden], aber das soll zumindest nicht an diesem Zeitpunkt erledigt werden. Es gibt keine Deadlines und es hängt immer [davon] ab, welche Fortschritte ich mache [...]“* (Interview 14, Absatz 60)

Es ist die Aufgabe des *Product Owners* das Entwicklungsteam vom Projektstress abzuschirmen und dafür zu sorgen, dass sie angenehm arbeiten können. Eine Person führte an, dass die Vorgesetzten ihm keinen Druck machen und er zwischenzeitlich Pausen einlegt, um neue Kraft und Ideen zu sammeln.

Aufgaben Die Hälfte der Probanden gab an, dass sie ihre Aufgaben im Rahmen des Sprints frei und in Absprache mit ihrem Team wählen. Unter Berücksichtigung des Könnens und des Interesses der Teammitglieder wird darüber entschieden, wer welche Aufgaben übernimmt:

> *„Also ich kann es mir quasi selbst aussuchen. Es ist jetzt nicht so als wenn irgendwer sagt ‚Okay, du musst das machen‘ sondern das ist eine Teamentscheidung. Das müssen wir bearbeiten. Wer hat da Bock drauf? Ja, okay, ich könnte es machen und dann, ja.“* (Interview 13, Absatz 44)

Die Reihenfolge, in welcher die Tickets bearbeitet werden, kann ebenfalls beeinflusst werden, sofern die Tickets nicht aufeinander aufbauen. Vier Probanden können zudem eigene Ideen und Anmerkungen zu Tickets einbringen, um die Gestaltung der Software zu beeinflussen.

Neun Interviewteilnehmer sprachen allerdings davon, dass ihnen ihre Aufgaben zugewiesen werden und sie wenig bis keinen Einfluss auf diesen Prozess haben. Tickets erhalten eine Priorisierung und müssen in dieser Reihenfolge abgearbeitet werden. Insbesondere Fehlertickets müssten schnell

gelöst werden. Außerdem bekommen sechs Personen ihre Aufgaben durch einen Vorgesetzten zugewiesen:

> *„Also es ist nicht so, dass wir einen großen Haufen haben und jeder nimmt sich von oben eine Karte runter, sondern die einzelnen Tickets werden direkt an die Leute verteilt.“* (Interview 8, Absatz 48)

Ein Interviewteilnehmer, welchem die Aufgaben zugewiesen werden, sprach zusätzlich davon, dass er und sein Team zu viele Aufgaben für einen Sprint hätten. Sie schaffen diese häufig nicht vollständig zu bearbeiten, was sich negativ auf die Motivation ausschlägt:

> *„[...] denn man hat oft das Problem, dass du nimmst ja, sag ich mal, fünf Sachen vor und schaffst dann aber nur drei. Und das ist allein von der Motivation her ein großes Manko, weil du dann jedes Mal am Ende des Sprints siehst, was du nicht geschafft hast, anstatt zu sehen, ah, ich habe drei Tickets geschafft.“* (Interview 8, Absatz 48)

Eine andere Person berichtete von einem ganz ähnlichen Problem. Sie bewältigt die Aufgaben des aktuellen Sprints häufig auch nicht, weil zu wenig Zeit eingeplant worden ist. Das führt dazu, dass die Anforderungen in den kommenden Sprint übernommen werden und es daher unmöglich ist, das angestrebte Pensum zu halten.

Zieldefinition Die Hälfte der interviewten Personen gab an, die Zieldefinition des Projekts und der Sprints beeinflussen zu können. Die grundlegende Richtung ist durch den Kunden definiert, allerdings kann häufig über verschiedene Details diskutiert werden, um ein passendes Ziel zu definieren:

> *„Das [...] ist eine Teamentscheidung. Wir haben ein Product Backlog, wo unsere Tickets [...] drinstehen, die eine gewisse Priorität haben und auch eine Schätzung. Ansonsten gibt es noch Vorschläge fürs Backlog, [...]. Und dann landet das bei unserem Teamleiter. Wir besprechen das mit dem Team, wie wichtig das Thema ist, priorisieren das Ganze.“* (Interview 13, Absatz 46)

Fünf Personen sahen einen Vorteil darin, Arbeitspakete selbst zu definieren und diese so kleinschrittig wie möglich zu halten. Auf diese Weise kann zum einen der Fortschritt im Projektverlauf besser gemessen werden. Zum anderen hat das Team auch die Teilerfolge im Blick und kann sich besser strukturieren:

> *„Ich glaube für die Zufriedenheit hat es viel gebracht, weil man auch Teilerfolge sieht und weil man eben nicht so viel Zeit damit verbringt, dass man denkt ‚Ach Mist, was soll ich denn überhaupt hier machen?‘ Man ärgert sich einfach nicht so viel.“* (Interview 10, Absatz 88)

Auf der anderen Seite sprachen neun Personen davon, dass sie wenig Einfluss auf die Zieldefinition haben. Vorgesetzte entscheiden, in welche Richtung das Projekt laufen soll. Dabei geht es häufig um Fristen und Bestimmungen durch den Kunden, welche eingehalten werden müssen:

> *„[...] wir haben eine Teamleiterin sozusagen, die entscheidet das. Hauptsächlich geht [es] halt oft um Fristen vom Kunden oder Dringlichkeiten und ja so wird das eigentlich festgelegt. Da hat das Team jetzt nicht so viel mitzureden.“* (Interview 20, Absatz 40)

5.2.2 Zusammenarbeit

Die Zusammenarbeit stellt einen wichtigen Faktor in Softwareprojekten dar [29]. In den Unternehmen der Interviewteilnehmer gestaltete diese sich vielfältig. In Abbildung 5.5 sind die erfassten Subthemen dargestellt.

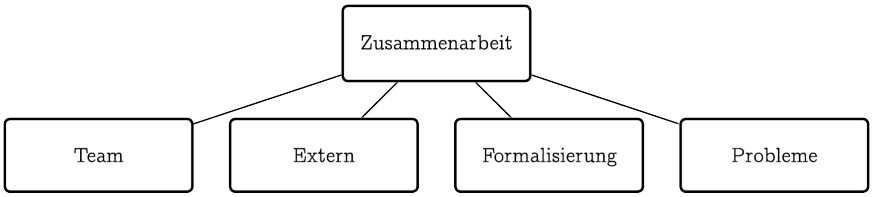

Abbildung 5.5: Die Subthemen des Themas Zusammenarbeit

Zusammenarbeit im Team Die Zusammenarbeit im Team wurde durch die Interviewteilnehmer facettenreich beschrieben.

Der Hauptfokus liegt hierbei auf direkter Zusammenarbeit, welche für fünf Probanden einen expliziten Vorteil des genutzten Prozesses darstellt. Im Büro können Mitarbeiter einfach zum Schreibtisch des anderen gehen, um dort zusammen zu arbeiten oder um Unklarheiten zu klären. Im Homeoffice kann sich unkompliziert angerufen und beispielsweise der Bildschirm geteilt werden, um miteinander zu arbeiten. Die Hälfte der Probanden führte an, dass sie regelmäßig auf diese Weise mit ihren Teammitgliedern zusammenarbeiten. Sie brainstormen oder lösen gemeinsam Probleme:

> *„[...] du hast halt diesen konstanten Austausch, was dir bei Problemlösungen auf jeden Fall hilft.“* (Interview 11, Absatz 68)

Zehn der Interviewteilnehmer berichteten außerdem von *Pair Programming* als gängige Praktik:

> *„Also Pair Programming machen wir. Also ein Kollege ruft mich an und sagt, ich komme hier nicht weiter. Dann wird [...] der Code geteilt. [...] Das bedeutet, also man macht dann halt Calls einfach auf, ruft sich gegenseitig an und dann arbeitet man zusammen an den Problemen.“* (Interview 2, Absatz 48)

Auch *Meetings* bieten eine gute Gelegenheit, um mit Kollegen zusammenzuarbeiten und Fragen zu klären. In diesem Rahmen kann offen über komplexere Lösungsansätze diskutiert und sich über Themen ausgetauscht werden. Müssen größere Angelegenheiten geklärt werden, können laut einem Probanden auch Workshops organisiert werden.
Für vier Personen war die Hilfsbereitschaft im Team sehr wichtig:

> *„Ja, und was ich auch sehr an meinem Team schätze, ist die Hilfsbereitschaft. Egal was und wann, ist jemand immer noch da.“* (Interview 19, Absatz 50)

Ein weiterer Proband stellte die Arbeitsteilung in den Mittelpunkt der Zusammenarbeit:

> *„Und wenn dann einer sagt, ich kriege die ganzen Sachen hier nicht fertig, ich bräuchte wen, der mir das abnimmt, dann kannst du natürlich gut da umverteilen und sagen, ja, bei mir ist gerade jetzt nicht so viel los. Komm, ich nehme dir noch die ein, zwei Fehler-Tickets gerade ab. Dann hast du irgendwie gerade den Rücken frei, um das andere Zeug zu machen.“* (Interview 5, Absatz 50)

Außerdem kann sich ein Entwickler durch den *Wissenstransfer* bei der Zusammenarbeit mit erfahreneren Teammitgliedern zusätzlich weiterbilden. Laut einem Probanden ist es auch hilfreich, Tickets thematisch zu rotieren, um eine Silobildung zu vermeiden. Auf diese Weise ist sichergestellt, dass Themen nicht nur von einzelnen Personen bearbeitet werden können und das Team neue Thematiken kennenlernt.

Zusammenarbeit mit Externen Insbesondere Probanden, welche in einem Unternehmen mit Softwareabteilung aber anderer, wirtschaftlicher Ausrichtung arbeiten, berichteten von fachlicher Kommunikation außerhalb des eigenen Teams. Diese Verbindung ist wichtig, um inhaltliche Fragen schnell und zielgerichtet klären zu können. Dabei sprachen alle fünf Teilnehmer, die in diesem Bereich Erfahrung hatten, von einer direkten und einfachen Zusammenarbeit. Sie können ihre Fragen unkompliziert stellen und bekommen direkt die Unterstützung, die benötigt wird:

> *„[...] dann nehmen die sich da immer die Zeit, erklären mir das, warum das so ist, geben mir so ein bisschen Kontext dazu und dann hilft mir das auf jeden Fall bei der Umsetzung.“* (Interview 4, Absatz 58)

Zusammenarbeit durch Formalisierung Drei Probanden berichteten, dass eine Formalisierung bestimmter Aspekte die Zusammenarbeit im Team vereinfacht. Ein Proband, welcher zu 100% im Homeoffice arbeitet, führte an, dass es für die Zusammenarbeit wichtig ist, bestimmte Meetings nicht ausfallen zu lassen. Insbesondere das *Daily* Meeting für den Austausch ist wichtig, um die Kommunikation im Team aufrecht zu erhalten. Des Weiteren vereinfacht die Nutzung von *Best Practices* die Zusammenarbeit deutlich:

> *„Wir haben versucht, Clean Code zu leben, dass man so programmiert, dass der Code nicht in sich einstürzt und dass die anderen auch damit klarkommen. Was natürlich eine unglaubliche Unterstützung ist. Aber jetzt nicht, das ist mehr so was, was man so laufend dann machen muss. Keine Einzelaktion.“* (Interview 10, Absatz 74)

Außerdem ist die Nutzung von zentralen Wissensplattformen, wie beispielsweise Wikisystemen, wichtig für die Zusammenarbeit. Auf diese Weise kann auf Informationen verlinkt werden, um effizienter zusammenarbeiten zu können.

Probleme bei der Zusammenarbeit In den Interviews nannten die Probanden eine Vielzahl an unterschiedlichen Problemen in Bezug auf die Zusammenarbeit. Drei Probanden berichteten, dass die Zusammenarbeit durch das Homeoffice erschwert wird. Sie können häufig nicht einfach zum nächsten Schreibtisch gehen, um Fragen zu klären, sondern müssen telefonieren oder Nachrichten schreiben. Ein Proband berichtete auch, dass hierdurch die Kommunikation abgenommen hat:

> *„Tendiert aber natürlich schon dazu, dass du dich vielleicht länger im eigenen Saft bewegst, als es vielleicht sinnvoll wäre. Das du vielleicht doch mal jemanden früher hättest fragen sollen.“* (Interview 6, Absatz 46)

Zwei weitere Probanden berichteten von erschwerter Kommunikation im Unternehmen, da Teams sich in unterschiedlichen Zeitzonen befinden und daher auch zu unterschiedlichen Zeiten arbeiten.

Ein weiteres Problem für die Zusammenarbeit im Team eines Probanden stellt die große Aufgabenlast dar:

> *„[...] wenn man dann eben sagt ‚Ich müsste jetzt eigentlich mal längere Konzept- oder Problembesprechung mit einem Kollegen machen‘, dann ist es immer schwierig einen zu erwischen, weil alle haben so ihre Aufgaben die durchkommen müssen.“* (Interview 20, Absatz 54)

Ein Proband kommuniziert sehr wenig mit seinen Teammitgliedern. Ein anderer hingegen sprach davon, dass die Kommunikation eine so große Rolle spielt, dass er sich manchmal wünschen würde, „wieder in Ruhe programmieren zu können“. Ein Interviewteilnehmer sprach zudem von Abhängigkeiten zwischen Teams. Sein Team kann mit manchen Aufgaben nicht starten, weil sie von den Ergebnissen eines anderen abhängig sind. Eine weitere Person führte aus, dass sein Team auf unterschiedliche Projekte verteilt ist, was die Zusammenarbeit ebenso erschwert.

5.2.3 Feedback

Abbildung 5.6 zeigt eine Übersicht der identifizierten Subthemen des Themas *Feedback*. Die Interviewteilnehmer gaben an, in erster Linie durch

verschiedene Personen und Gruppen Feedback zu erhalten. Dabei spielte das Feedback durch das eigene Team die größte Rolle. Zusätzlich bekommen die Probanden durch ihre Vorgesetzten und Kunden, aber auch durch technische Tools Feedback zu ihrer Arbeit.

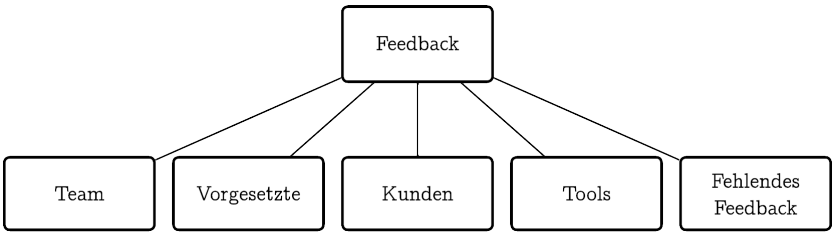

Abbildung 5.6: Die Subthemen des Themas Feedback

Feedback durch das Team Eine große Mehrheit der Teilnehmer gab an, Feedback durch ihr Team zu erhalten. Lediglich zwei Probanden teilten mit, dass sie wenig bis kein Feedback durch ihre direkten Kollegen bekommen.

Das Feedback gestaltet sich dabei sehr divers. Einen großen Aspekt stellen Code-Reviews bei *Pull Requests* und das dadurch erhaltene Feedback durch Kollegen dar. Diese wurden von insgesamt 15 der 22 Teilnehmer als Feedbackmechanismus in ihrem Unternehmen genannt. Da der selbstgeschriebene Code nicht eigenhändig gereviewt werden kann, muss dies durch Teammitglieder geschehen. Dabei schreiben die Entwickler ihre Anmerkungen an den Code, welcher verbessert oder anders gelöst werden kann, um auf diese Weise direktes Feedback und ihre eigenen Erfahrungen zu teilen.

Einen weiteren wichtigen Punkt stellen Review-Termine für die Vorstellung erarbeiteter Features dar:

> *„Dann Sprint Review, auch dieses Review Meeting, was wir haben. Das finde ich auch sehr gut, weil da dann nochmal so der kritische Blick von allen Leuten, die daran beteiligt sind, dann nochmal drüber schaut über die Features und auch nochmal Änderungswünsche genannt werden können.“* (Interview 21, Absatz 82)

Hier erhalten die Probanden hauptsächlich durch das Team und den *Product Owner* Feedback zum Projekt, um überprüfen zu können, ob ihre Entwicklung in die gewünschte Richtung geht oder sie Änderungen einpflegen müssen. Ein Proband gab zudem an, dass sein Team Reviews mit dem Teamleiter, dem Produktmanagement und den Geschäftsführern durchführt.

Ein Proband berichtete außerdem von sogenannten *„Smoketests“*. Andere Mitarbeiter begutachten das entwickelte Feature kritisch, bevor es zur weiteren Prüfung an die *Quality Assurance* gereicht wird. Denn je später ein

Fehler entdeckt wird, desto teurer wird er für das Unternehmen.

Allerdings ist das Feedback im Team nicht auf feste Termine oder Praktiken beschränkt. Die Probanden berichteten auch von Feedback im Arbeitsalltag:

> *„Genau, und auch im Team, was ich auch sehr an meinem Team schätze, ist dieses [sic!] Feedback-Loop, sagt man so. Dass wir auch untereinander Feedback geben, was man alles gebaut hat, entwickelt hat.“* (Interview 19, Absatz 44)

Zwei Personen berichteten von teaminternen Terminen, in denen sie zwischendurch ihre Lösung und den dazugehörigen Code präsentieren, um Feedback zu erhalten. Insbesondere das Feedback von erfahreneren Kollegen ist wertvoll für die persönliche Weiterentwicklung. Das *Daily Meeting* stellt hierfür einen guten Rahmen dar.

Feedback durch Vorgesetzte Neben dem Feedback durch das Team berichtete die Hälfte der Probanden auch von Rückmeldungen über ihre Arbeit durch Vorgesetzte. Das findet häufig in Mitarbeitergesprächen statt, die in regelmäßigen Abständen geplant sind. Dabei wird nicht nur über die eigentlichen Arbeitserzeugnisse gesprochen, sondern auch über die langfristige Entwicklung im Unternehmen. Es wurde außerdem von Gesprächen in der Einarbeitungsphase berichtet, um das Feedback anderer Mitarbeiter mitzuteilen:

> *„Ansonsten hatte ich jetzt in der Anfangszeit monatliche Feedbackgespräche mit dem Chef, der sich dann Feedback von den anderen aus dem Team eingeholt hat und mir das gespiegelt hat. Oder was aus seiner Sicht über das läuft. Ich konnte sagen, wie es für mich lief, was ich so gemacht habe.“* (Interview 4, Absatz 46)

Feedback durch den Kunden Fünf Interviewteilnehmer gaben an, dass sie Feedback durch den Kunden erhalten. Das geschieht zum einen indirekt über Benutzerstudien, in welchen die Nutzer das Produkt beurteilen. Zum anderen gibt es direktes Feedback durch den Kunden, insbesondere durch den *Product Owner*, aber auch durch andere Vertreter. Der Product Owner schaut sich das entwickelte Produkt im Rahmen eines Review-Termins an und stellt hier sein Feedback vor. Ein Teilnehmer berichtete außerdem von einem *Stakeholder-Review*, in welchem auch andere Stakeholder ihr Feeback mitteilen können:

> *„Genau, ein Stakeholder Review, da kriegt man halt irgendwie ein ausführliches Feedback, weil Stakeholder sind halt anders interessiert, was passiert, was hat man so Schönes entwickelt, im Frontend besonders, wenn die irgendwie ein bisschen durchklicken können oder neue Funktionen dazugekommen sind. Da freuen sie sich mega drauf.“* (Interview 19, Absatz 44)

Feedback durch technische Tools Für viele stellt Feedback mithilfe von Tools, wie beispielsweise *Pull Requests* über GitHub[1] oder Kommentare zu Tickets in Jira[2] einen wichtigen Aspekt dar. Automatisierte Tests stellen neben Kontrollen durch die *Quality Assurance* Abteilung zusätzlich wichtige Feedbackmöglichkeiten dar, um sicherzustellen, dass gute Arbeit geleistet worden ist.

Fehlendes Feedback Sechs Probanden berichteten, dass sie insgesamt wenig Feedback erhalten:

> *„Allgemein sehr wenig. Wir haben kein Review. Das heißt eigentlich nur, solange ich nicht ankomme und jemanden mal um eine Meinung frage oder irgendwer im Nachhinein zu mir kommt und in der Historie sieht, sag mal, was hast du denn da verzapft? Solange alles läuft, gibt es nicht weiter Feedback.“* (Interview 15, Absatz 42)

Wenn sie Feedback bekommen, geschieht das eher zwischendurch, nicht beispielsweise in einem *Review Meeting*. Ein anderer Proband berichtete, dass bei ihm auch in Reviews neu implementierte Features nur „abgenickt“ werden, ohne weiter darauf einzugehen.

Allerdings ist fehlendes Feedback nicht immer etwas Schlechtes. Zwei Probanden gaben zu verstehen, dass es für sie gutes Feedback ist, wenn ihr Code ohne Beanstandung durch Dritte funktioniert:

> *„Wenn es nichts zu meckern gibt, ist es gutes Feedback.“* (Interview 4, Absatz 46)

5.2.4 Fehlerkultur

Die Mehrheit der Probanden berichtete von einer positiven und lösungsorientierten Fehlerkultur in ihren Unternehmen. Es wird kein Schuldiger gesucht, sondern gemeinsam eine Lösung für das Problem entwickelt. Außerdem wird geschaut, wie sich solche Problematiken zukünftig vermeiden lassen:

> *„Also es gibt kein Fingerpointing oder so. Wenn es einen Fehler gibt, dann wird kein Schuldiger gesucht, sondern man guckt dann ja gemeinsam drauf und schaut, wie man das gut lösen kann.“* (Interview 18, Absatz 44)

Ein Proband berichtete zusätzlich vom *Sprint Retrospektive* Meeting als Kernstück der Fehlerkultur in ihrem Unternehmen. Hier können Mitarbeiter Fehler und Probleme ohne Umschweife ansprechen, da davon ausgegangen wird, dass jeder nach *bestem Wissen und Gewissen* gehandelt habe. Es wird nicht der Verantwortliche, sondern eine Lösung gesucht. Andere Teilnehmer berichteten, dass Fehler teamintern behandelt werden und ein kurzer Hinweis für die Beseitigung meist reicht. Eine Person schilderte

[1] Zu finden auf: https://github.com/
[2] Zu finden auf: https://www.atlassian.com/de/software/jira

außerdem, dass ihr *Mentor* hilfreiche Tipps gibt, wenn ihm Fehler in der Programmierung auffallen. Auch die Arbeitsatmosphäre wird durch entstandene Fehler wenig beeinflusst:

> *„Nee, also wir sind halt ein Team und wir sind mit Fehlern und Erfolgen und alles halt verantwortlich. Also wenn irgendwas halt kaputt geht, sind wir alle dafür verantwortlich und das schätze ich auch sehr an meinem Team.“* (Interview 19, Absatz 48)

Lediglich drei Teilnehmer berichteten von einer Fehlerkultur, welche negativ und nicht lösungsorientiert ist:

> *„Wenn dann der Projektleiter in jedem Daily dort sitzt und anderthalb Stunden irgendwie, ja, dann die anderen Kollegen da zur Sau macht. Das ist für mich dann nicht Fehlerkultur, beziehungsweise, wie man vernünftig damit umgehen sollte.“* (Interview 11, Absatz 46)

Auch wenn es für entstandene Fehler keinerlei Konsequenzen gibt, berichtete ein Proband von Schuldzuweisungen sowohl innerhalb des Teams, als auch durch Vorgesetzte. Ein weiterer Proband berichtete von unterschiedlichen Herangehensweisen an die Fehlerbehandlung, je nachdem wer den Fehler verursacht hat.

5.2.5 Meetingkultur

Meetings spielen im Arbeitsalltag eines jeden Probanden eine wichtige Rolle. Jeder berichtete von Terminen zu verschiedenen Themen, an welchen er teilnehmen muss. Eine Übersicht über die erfassten Subthemen ist in Abbildung 5.7 dargestellt.

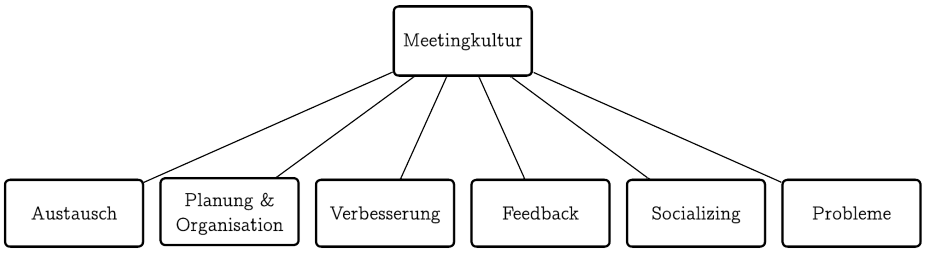

Abbildung 5.7: Die Subthemen des Themas Meetingkultur

Meetings für den Austausch 17 Personen berichteten von *Daily* Meetings, welche für den Austausch im Team genutzt werden. Der Termin dient als Synchronisationspunkt. Das Team tauscht sich über den aktuellen Stand aus, woran gerade gearbeitet wird und wo Probleme liegen. Zudem werden bilateral Informationen ausgetauscht oder Termine für weiterführende Gespräche gemacht. Zeitlich soll dieser Termin in den meisten Fällen ca. 15 Minuten nicht überschreiten. Lediglich ein Proband berichtete von

einer vollen Stunde täglich. Zudem findet in zwei Fällen das *Daily* nur zweimal in der Woche statt. Allerdings ist es ein guter Rahmen, um seine Teammitglieder zu unterstützen:

> *„[...] ist gerade irgendwas Wichtiges, [...], was gerade geklärt werden muss, [...], damit das alle wissen oder vielleicht auch irgendwie alle davon Bescheid wissen und merken können, könnte ich da vielleicht irgendwie beisteuern, dann geht das da so ein bisschen halt direkt an alle.“* (Interview 9, Absatz 60)

Außerdem sprachen vier Interviewteilnehmer von *Weekly* Meetings. Zum einen wird hier der Arbeitsablauf wochenweise geplant. Zum anderen wird dieses Meeting auch für den *Wissenstransfer* untereinander genutzt:

> *„In dem wir über aktuelle Themen sprechen, die, unter allen was angeht oder einfach, womit jemand was wissen möchte, aber auch gleichzeitig das Wissen teilen möchte.“* (Interview 8, Absatz 66)

Des Weiteren berichteten insgesamt fünf Personen von einem gesonderten Meeting für die Synchronisation im Team. Hier wird das Team, ähnlich wie im Daily, auf den neuesten Stand gebracht. Es wird beispielsweise über den Sprintverlauf gesprochen. Außerdem kann über aufgetretene Hindernisse und verbesserungswürdige Punkte gesprochen werden.

Hinzu kämen spontane Meetings mit anderen Teammitgliedern, welche für die *Zusammenarbeit* genutzt werden. Es werden beispielsweise Fragen geklärt oder um Feedback gebeten:

> *„Also [...] wenn ich von mehreren Entwicklern etwas möchte oder präsentieren möchte, dann erstelle ich ein Meeting.“* (Interview 21, Absatz 54)

Planung- und Organisationsmeetings Zusätzlich zu Meetings für den Austausch im Team wurden häufig Planungsmeetings genannt, an welchen die Probanden teilnehmen. 15 Interviewteilnehmer berichteten von solchen Meetings. Dabei wurde das *Sprint Planning* mit 14 Nennungen am häufigsten angeführt. In diesen Meetings wird der kommende Sprint geplant. Es wird festgelegt, welche Aufgaben durchgeführt werden und in welcher Reihenfolge das geschehen muss. Außerdem wird der Aufwand für die Aufgaben geschätzt:

> *„Sprintplanung gehen wir durch, was wurde gemacht. Wo gibt's Probleme, wie machen wir weiter? Bewertung der anstehenden Aufgaben, Priorisierung, ob das jetzt in diesen Sprint gehört oder nicht. Abschätzung der benötigten Zeit, um halt den Sprint vollzukriegen und alle vernünftig auszulasten.“* (Interview 3, Absatz 34)

Meetings für Weiterbildung und Verbesserung Die Teilnehmer berichteten von verschiedenen Meetings, in welchen ein Wissenstransfer

und damit eine Weiterbildung stattfindet. Insbesondere die *Retrospektive* wurde von der Hälfte der Probanden genannt. Hier wird besprochen, wie der letzte Sprint verlief und welche Probleme aufgetreten sind. Es kann erzählt werden, was aus persönlicher Sicht schief lief und was den Mitarbeitern nicht so gut gefallen hat. Ebenso können Verbesserungsvorschläge durch die Mitarbeiter eingebracht werden. Ein Proband beschrieb dieses Meeting sogar als Kernstück des Scrum-Prozesses:

> *„Also ich muss sagen, das ist aus meiner Sicht wirklich so ein Kern des Scrum-Prozesses, wo du halt eine Atmosphäre schaffst, wo sinnvoll kritisiert werden kann und wo aus der Kritik dann auch was folgt, was ja sonst immer so ein bisschen schwierig ist.“* (Interview 6, Absatz 32)

Zwei Personen berichteten außerdem von teamübergreifenden Meetings für den Wissentransfer. Hier kann mit anderen Teams über Vorgehensweisen gesprochen werden, um neue Ideen zu finden oder Prozesse zu optimieren.

Feedbackmeetings Wie bereits in Abschnitt 5.2.3 angeschnitten, wird das sogenannte *Review Meeting* nach Angaben der Probanden für Feedback genutzt. Sie stellen dem Product Owner ihre Erzeugnisse vor und dieser nimmt sie ab, wenn alles in Ordnung ist. In den Unternehmen von zwei Interviewteilnehmern sind, neben dem Product Owner, auch die eigentlichen Kunden regelmäßig während eines Reviews vor Ort:

> *„Und Review, wie gesagt, hatten wir auch tatsächlich dann mit den Stakeholdern, mit den Leuten, die das am Ende benutzt haben, größtenteils.“* (Interview 10, Absatz 56)

Ein Proband berichtete das im Review Mitarbeiter ebenfalls Ideen für zukünftige Entwicklungen äußern können:

> *„Und dann darf man noch Ideen reinschreien und dann kann der Entwickler sagen, ja, während des Entwickelns ist mir aufgefallen, man könnte ja noch eigentlich das hier machen.“* (Interview 6, Absatz 32)

Neben dem Review berichtete ein Teilnehmer auch von Workshops, welche in Abständen von einigen Monaten für die Kunden durchgeführt werden.

Socializing-Meetings Drei Personen berichteten zudem von *Socializing*-Meetings. Ein Proband nannte diese *Kaffee-Termin*, ein anderer *Kuschel-Call*. Dabei geht es in erster Linie um zwischenmenschliche Aspekte, nicht um projektbezogenen Austausch. Ein Interviewteilnehmer berichtete, dass diese Termine für ihn sehr wichtig sind, da er zu 100% im Homeoffice arbeitet. Auf diese Weise kann er menschlichen Kontakt beibehalten, welcher sonst verloren gehen würde. Allerdings wird auch in diesen Meetings unter Umständen über inhaltliche Themen gesprochen:

> *„Das war ursprünglich mehr so ein Zwischenmenschliches. Es ist aber auch [...] zum Gucken, ja, was steht an?“* (Interview 3, Absatz 56)

Probleme der Meetingkultur Fünf Probanden berichteten von Problemen, welche mit der Meetingkultur ihrer Unternehmen einhergehen. Zum einen gaben drei Personen an, dass die Retrospektive in ihrem Unternehmen nur sehr kurz und sporadisch durchgeführt wird oder sogar ganz ausfällt. Zum anderen berichteten vier weitere Personen von sehr vielen oder langen Meetings, an welchen sie teilnehmen müssen. Diese vereinnahmen einen beträchtlichen Teil ihres Arbeitstags und sie haben daher weniger Zeit für ihre anderen Aufgaben:

> *„Je nachdem gibt es natürlich immer noch andere Termine, die da so reinfallen. Also ich nenne das gerne [...] Reibungen. Du als Entwickler willst ja so wenig Reibungen wie möglich haben, sondern du willst ja entwickeln. Aber es kommt halt immer irgendwas anderes noch dazwischen.“* (Interview 11, Absatz 34)

Ein weiterer Proband sprach davon, dass seine Teammitglieder in vielen Meetings nicht zuhören würden.

5.2.6 Weiterbildung

Die Probanden berichteten von mehreren Aspekten der Weiterbildung in ihrem Unternehmen. Diese sind in Abbildung 5.8 dargestellt.

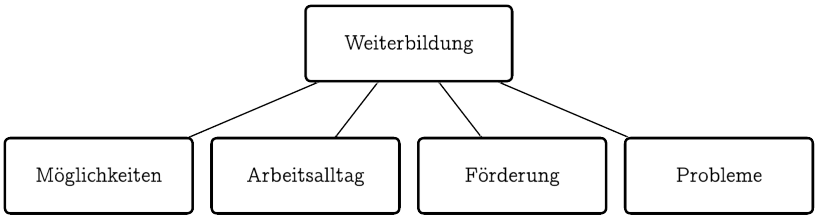

Abbildung 5.8: Die Subthemen des Themas Weiterbildung

Weiterbildungsmöglichkeiten Es wurden zahlreiche Möglichkeiten der Weiterbildung genannt. Zwei Probanden gaben an, dass sie insbesondere die fachlichen Weiterbildungsmöglichkeiten in ihrem Unternehmen als persönlichen Vorteil sehen.

Den größten Punkt stellen dabei *Präsenzschulungen* dar. Diese werden zu vielen verschiedenen Themen angeboten. Es gibt Fortbildungen, in welchen technische Themen erläutern werden. Beispielsweise wurde hier ein Kurs für *SQL für die Anwendungsentwicklung* oder Schulungen für neue, technische Frameworks erwähnt. Zwei Probanden berichteten zudem von Zertifizierungen, die sie für ihre Arbeit benötigen und durch eine Teilnahme an Schulungen erhalten würden. Hier wurden insbesondere Software-Architektur-Schulungen genannt. Außerdem gaben drei Interviewteilnehmer an, dass erfahrene Mitarbeiter regelmäßig interne Schulungen zu aktuell im

Unternehmen relevanten Themen anbieten. Diese werden häufig in Form von *Workshops* durchgeführt, in welchen die gelernten Dinge direkt ausprobiert werden können:

> *„Und dann gibt es noch Workshops. Wir haben zum Beispiel ein Workshop [...] gemacht, [...], um einfach mal zu schauen, wie das eigentlich geht, um da so ein Gefühl dafür zu bekommen. [...]. Das war dann ein Tag so ein bisschen mit [Tool] rumspielen, wenn man so möchte. Natürlich mit dem Ziel rauszukriegen, wie es geht, aber ja.“* (Interview 5, Absatz 62)

Drei Personen erläuterten, dass sie die Möglichkeit haben, thematisch passende Konferenzen zu besuchen. Fünf Probanden bilden sich auch mithilfe von interner Dokumentation, wie beispielsweise Wikiseiten, weiter:

> *„Und auch werden wir wahrscheinlich in unserer Wikiseite gucken, wenn wir etwas wissen wollen. Da ist [sic!] auch teilweise bestimmte Sachen gut beschrieben.“* (Interview 1, Absatz 32)

Neben technischen gibt es auch verschiedene, fachliche Schulungen. Diese decken die für den Arbeitsalltag benötigten Themen ab. Unter anderem wurde ein Kurs *Grundzüge Versicherungswesen* von einem Mitarbeiter in einem Versicherungsunternehmen genannt. Insgesamt wurde von einem sehr diversen Schulungsangebot berichtet:

> *„Das ist ein sehr breit gefächertes Spektrum. Da haben wir auch einige Kurse, die wir buchen können für Requirements Engineering und solche Späße.“* (Interview 12, Absatz 78)

Zwölf Probanden berichteten darüber hinaus von *Online-Weiterbildungen*. Bei diesem Aspekt wurden insbesondere *E-Learning-Plattformen* genannt, wie beispielsweise *Udemy*[3] oder *Coursera*[4]. Hier können Video-Kurse zu vielerlei Themen angeschaut und sich auf diese Weise weitergebildet werden. Darüber hinaus schilderten drei Personen, dass sie über die Mutterfirmen ihres Unternehmens Zugriff auf interne Fortbildungsplattformen haben:

> *„Wir haben intern ein Schulungsportal, das auch bedingt durch [Mutterfirma] recht groß ist, wo man sich über alles Mögliche, was technisch gesehen ist, weiterbilden kann.“* (Interview 8, Absatz 78)

Weiterbildung im Arbeitsalltag Neben expliziten Möglichkeiten der Weiterbildung wurde häufig auch die Weiterbildung im Rahmen der täglichen Arbeit genannt. Den wichtigsten Aspekt stellte hierbei das *Mentoring* dar, welches von insgesamt 14 der 22 Probanden genannt wurde. Diese Art der Weiterbildung ist besonders hilfreich während der *Einarbeitungsphase*, jedoch nicht ausschließlich darauf beschränkt. Die Probanden erwähnten

[3] Zu finden auf: https://www.udemy.com/de/
[4] Zu finden auf: https://www.coursera.org/

eine Ansprechperson, welche ihnen bei Fragen oder Unklarheiten mit Rat und Tat zur Seite steht:

> *„[...] da habe ich meinen Paten, der sich halt intensiv damit auskennt. Mit dem spreche ich mich dann halt auch ab, wenn ich irgendwelche Fragen habe, was das Verständnis von dem System angeht oder sowas.“* (Interview 12, Absatz 58)

Sie schauen gemeinsam über Ergebnisse, um diese zu kontrollieren und gegebenfalls zu verbessern. Zwei Interviewteilnehmer gingen zudem auf *Code Reviews* als Möglichkeit ein, um das eigene Wissen zu erweitern. Hier findet ein Wissenstransfer, beispielsweise von Programmierpraktiken, statt. Insbesondere auch, weil sie die Möglichkeit haben, über die beste Lösung gemeinsam zu diskutieren:

> *„Das hilft auch einfach, um dann von der Codequalität noch besser zu werden oder vom Programmierstil, wenn einfach andere Leute noch mal drauf schauen und vielleicht auch nur ihre Meinung so dazuschreiben. [...] Dann kann man auch drüber diskutieren und dann vielleicht was finden, was dann irgendwie noch besser ist [...]. Das ist eine sehr gute Sache.“* (Interview 5, Absatz 68)

Elf Probanden berichteten zudem von einem Wissenstransfer im Team, welcher an mehreren Stellen stattfindet. Es wurden verschiedene Meetings genannt, die sich hierfür eignen. Drei Probanden berichteten von Terminen, welche explizit für diesen Zweck eingerichtet wurden:

> *„[...] haben wir auch regelmäßige Wissenstransfers, also auch einen Austausch in der Woche über Software-Themen, aber auch, dass man sich intern im Prinzip weiterbildet.“* (Interview 22, Absatz 50)

Des Weiteren wurden das *Daily Meeting* und die *Sprint Retrospektive* genannt. Hier können Fragen gestellt und Wissen in einem unkomplizierten Rahmen geteilt werden. Insbesondere die Retrospektive (vgl. Abschnitt 5.2.5) ist eine gute Möglichkeit, um die eigene Vorgehensweise zu reflektieren und potentielle Verbesserungsmöglichkeit mit Teammitgliedern zu besprechen:

> *„Unterschätz die Retrospektiven in der Hinsicht nicht. Du nimmst viel daraus mit, wenn du wirklich mal reflektierst, was du eigentlich machst.“* (Interview 6, Absatz 60)

Förderung durch das Unternehmen Viele Probanden berichteten von einer Unterstützung für die Weiterbildung durch den Arbeitgeber. Zum einen erläuterten sechs Personen, dass sie ein festgelegtes Kontingent an Arbeitszeit für die Weiterbildung nutzen können. Ein Interviewteilnehmer führte an, dass er in einer normalen Woche zwei von 38 Stunden für Bildung nutzen könnte. Jemand Weiteres erzählte von *Open Work Days*, welche zur freien Verfügung für Bildungsmaßnahmen stehen:

> *„Wir haben 11 Open Work Days im Jahr [...]. Da arbeiten wir nicht im Projekt, sondern können uns im Endeffekt damit beschäftigen, was wir machen wollen, also uns selber zu Themen fortbilden.“* (Interview 16, Absatz 64)

Außerdem schilderten vier Probanden eine Unternehmenskultur, welche Fort- und Weiterbildungen direkt unterstützt und Mitarbeiter dazu auffordert, Schulungen zu besuchen:

> *„Ich wurde halt jetzt neulich von meinem Vorgesetzten angesprochen, ob ich nicht vielleicht in einem Jahr Softwarearchitekt vielleicht werden möchte. Wenn da Potenzial in den Leuten gesehen wird, dann wird man gefragt.“* (Interview 18, Absatz 64)

Probleme bei der Weiterbildung Allerdings sprachen Probanden auch von Problemen im Bezug auf die Weiterbildung im Unternehmen. Zwei Personen gaben an, dass keine Schulungen durchgeführt werden:

> *„Theoretisch heißt es immer, wir sollen immer machen, aber in der Praxis gibt es bei uns irgendwie nie Fortbildungen. [...] Vor allem kommt nichts proaktiv irgendwie mal, so willst du nicht das oder so, sondern wenn muss man da schon ziemlich, ziemlich nachhauen. Das ist nicht so, wie man es sich eigentlich wünschen würde.“* (Interview 9, Absatz 62)

Zwei Interviewteilnehmer berichteten von zu wenig Zeit für Fortbildungen. Wenden sie Arbeitszeit für die eigene Bildung auf, bleiben andere Aufgaben auf der Strecke. Diese müssen im Anschluss in Form von Überstunden oder Wochenendarbeit aufgeholt werden:

> *„Es ist halt nur immer die Frage, passt das mit der Zeit, die du ins Projekt planen musst oder nicht, weil sonst sind es hinterher wieder Überstunden oder du musst es am Wochenende machen und ehrlich gesagt hat man da oft nicht so Bock drauf.“* (Interview 11, Absatz 62)

5.2.7 Teamstabilität

Die Stabilität ihres Teams war für die meisten Interviewteilnehmer relevant. Für 20 Personen war ein stabiles Team ohne übermäßige Fluktuation wichtig. Es wurden unterschiedliche Vorteile eines konstanten Teams genannt. Zum einen sei das für den Projekterfolg von großer Relevanz, da Projekte in vielen Fällen sehr groß sind und die Einarbeitung viel Zeit kostet. In Projekten, die von neuen Teammitgliedern übernommen werden, ereignen sich große Probleme:

> *„Die Stabilität ist für mich eigentlich Prio [...]. Ich würde sogar fast sagen Top 1 Prio [...] es ist schon ein ziemlich großes Projekt [...]. Und das komplette Team wurde abgesetzt und sozusagen ein Neues draufgesetzt oder zumindest ein Kleineres sogar. Und ich war*

> *sozusagen der Typ, der noch so ein bisschen die Zeit mit den [Externen] erlebt hat und dann aber auch das neue Team einarbeiten musste. Dementsprechend, das war richtig scheiße. Also ich kann dir nur sagen, Stabilität ist echt, es ist extrem wichtig, [...] dass du nicht so viel Fluktuation hast in dem Team [...]. Du kannst nicht sagen von heute auf morgen, ich setze das komplette Team ab, setze ein neues Team auf ein Softwareprojekt drauf. Das geht halt komplett nach hinten raus.“* (Interview 11, Absatz 56)

Zum anderen spielt auch der soziale Faktor eine wichtige Rolle. In einem eingespielten Team weiß jeder, wie miteinander umgegangen werden muss. Ein Teilnehmer berichtete, dass ihm die Arbeit gerade aufgrund seines Teams viel Spaß macht. Ein harmonisches Team unterstützt den Austausch und eine direkte Kommunikation:

> *„Also ein festes Kernteam ist wichtig. Ein Team, das gut zusammenarbeitet, ist sehr wichtig und ein Team, das menschlich harmoniert, ist sehr wichtig. Gerade wenn viele Junge dabei sind, ist ein netter Austausch einfach nochmal deutlich angenehmer und man kann direkte Wege machen und man kann auch direkt über etwas sprechen. Das hängt aber nicht mal vom Alter ab. [...] Das ist eine menschliche Frage.“* (Interview 22, Absatz 58)

Drei Teilnehmer gaben allerdings an, dass eine gewisse Fluktuation im Team hilfreich ist. Auf diese Weise gelangen neue Ideen ins Team und es ist schön, neue Menschen kennenzulernen. Allerdings geht hierdurch auch Produktivität und Wissen verloren:

> *„Dann halt zwar schade, um jemanden, mit dem man sich gut versteht oder der halt entsprechendes Wissen halt hat. Wenn das Wissen weg ist, ist halt auch doof. Aber da kommt man dann im Zweifel nicht drum herum.“* (Interview 12, Absatz 70)

5.2.8 Einarbeitung

Wird ein neuer Mitarbeiter einem Team zugewiesen, stellt die Einarbeitung die erste Hürde im Arbeitsalltag dar. Hierfür äußerten die Probanden eine Vielzahl an unterschiedlichen Herangehensweisen, allerdings auch einige Probleme, die dabei entstehen.

Möglichkeiten der Einarbeitung Fünf Teilnehmer führten an, dass es Einarbeitungspläne gibt, an welche sich neue Mitarbeiter halten sollen. Zu Beginn der Einarbeitung stehen dabei verschiedene Pflichtschulungen an. Diese behandeln neben Sicherheitseinweisungen auch Einführungen in verschiedene Themen. Ein Proband berichtete von einer Schulung über die genutzte Entwicklungsumgebung und die Generierung von Artefakten in

der Programmierung. Hinzu kommen außerdem Workshops und Schulungen. Hier werden das Team, das Projekt und die Software vorgestellt.

Insgesamt zwölf Probanden führten an, dass neue Teammitglieder über *„einfache“* Aufgaben an den Arbeitsalltag herangeführt werden. Hierunter fallen beispielsweise *Refactoring-Aufgaben*, um sich mit dem Projekt vertraut zu machen. Außerdem bekommen neue Mitarbeiter kleinere, technische Aufgaben, welche wenig Wissen aus Fachbereichen benötigen:

> *„Wir suchen uns dann eine möglichst technische Aufgabe vorher raus, womit man, wenn man die bearbeitet, möglichst die Anwendung gut kennenlernen kann, sodass man mal möglichst viele Aspekte so sieht, ohne jetzt aber schon zu viel von der Business-Domain wissen zu müssen.“* (Interview 5, Absatz 56)

Die mit sechs Nennungen am zweithäufigsten erwähnte Art der Einführung in die tägliche Arbeit stellt die *Dokumentation* dar. Hierzu zählen Dokumentationen für das Aufsetzen der Hardware aber auch Software-Dokumentation. Mithilfe dieser Dokumentationen können neue Mitarbeiter viele Aufgaben bereits selbstständig erledigen.

Einen weiteren wichtigen Stützpfeiler stellen die Kollegen in der Einarbeitungsphase dar. Diese sollen jederzeit für Fragen offen sein, um bei der Einarbeitung zu unterstützen. Zudem stellen diese auch im Laufe der Zeit verschiedene Thematiken vor, welche für den Arbeitsalltag wichtig sind. Weiterhin gaben fünf Probanden an, dass für die Einarbeitung auch Praktiken wie *Pair Programming* genutzt werden. Auf diesem Weg können neue Mitarbeiter schnell ins Team integriert und alleine arbeitsfähig werden:

> *„Wir hatten zum Beispiel Leute, die sind da neu ins Team gekommen, aber wir haben die da ins Team integriert, eben indem wir Pair Programming mit den echten User Stories gemacht haben. Das war wirklich ein sehr, sehr hilfreiches Tool, um die Leute da auf Geschwindigkeit zu bringen.“* (Interview 10, Absatz 74)

Insgesamt agieren die erfahrenen Kollegen als eine Art *Mentor*, um neue Mitarbeiter zu unterstützen:

> *„Man wird sozusagen von einem Mitarbeiter permanent an die Hand genommen, sozusagen, der für einen, sag ich mal, den Mentor wirkt in der Anfangszeit, in der On-Boarding-Phase und der bringt dann durch den Alltag durch und versucht dann langsam in die einzelnen Tools einzuführen.“* (Interview 8, Absatz 74)

Probleme bei der Einarbeitung Die Einarbeitung mithilfe von Dokumentation wurde von den Probanden sehr häufig genannt. Allerdings wurden hierfür ähnliche Probleme wie bei der Weiterbildung (vgl. Abschnitt 5.2.6) angeführt. So ist eine Einarbeitung durch Dokumentation nur wenig geeignet, weil eine ordentliche Dokumentation nicht existiere:

> *„Ganz ehrlich, das ist Zeitverschwendung. Kannst du dir sparen. Doku gibt es eh nur in drei Zuständen. Sie ist entweder nicht vorhanden, sie ist veraltet oder nicht relevant. Keiner von drei Zuständen ist hilfreich.“* (Interview 6, Absatz 56)

Vier Probanden berichteten von sehr komplexen Themen, welche eine schnelle Einarbeitung unmöglich machen. Aus diesem Grund reicht die Einarbeitungsphase für ein tiefgehendes Verständnis nicht aus. Kollegen müssen weiterhin für Verständnisfragen offen sein und neue Mitarbeiter unterstützen. Die Einarbeitung eines neuen Mitarbeiters verursacht auch viel Mehrarbeit für das Team. Es wird viel Zeit dafür gebunden und andere Projekte bleiben dabei auf der Strecke:

> *„Die kennen die Prozesse nicht, das ist immer am Anfang relativ viel Arbeit, die irgendwie ins Team zu integrieren. Wir hatten das tatsächlich auch schon jetzt in den letzten Jahren, das immer wieder Kollegen gegangen sind, gekommen sind. Und ja, das ist immer erstmal anstrengend, sage ich mal.“* (Interview 20, Absatz 58)

5.2.9 Arbeitsalltag

Der Arbeitsalltag der Entwickler wurde von vielen Teilnehmern ähnlich beschrieben. Zu Beginn des Tages überprüfen 14 Probanden ihre Postfächer und andere Informationskanäle auf Neuigkeiten. Drei Personen berichteten beispielsweise, dass Kollegen in anderen Zeitzonen früher mit der Arbeit beginnen und dementsprechend bereits Nachrichten geschrieben haben könnten. Neben der Haupttätigkeit des Programmierens verbringen die Entwickler allerdings auch viel Zeit in verschiedenen Meetings. Acht Probanden berichteten allerdings, dass sie in erster Linie alleine an ihren Aufgaben arbeiten:

> *„So eine direkte Kollaboration haben wir nicht, in dem Sinne, dass wir alle unsere eigenen Tickets bearbeiten.“* (Interview 4, Absatz 56)

Probleme im Arbeitsalltag Die Teilnehmer sprachen im Verlauf der Interviews von verschiedenen Problemen, die in ihrem Arbeitsalltag auftreten. Sechs Probanden berichteten von Personen im Projekt, welche mehrere Rollen innehaben. Das ist schädlich für die Produktivität, da sie sich gleichzeitig um unterschiedliche Belange und Aufgaben kümmern müssen:

> *„Weil das bei uns [...] ein großes Thema ist, dieses Doppelrollen belegen, also Product Owner ist Scrum Master, Entwickler ist Scrum Master, und so ein Scheiß halt und der Entwickler muss sich überlegen, wo geht die Vision hin oder was soll halt gemacht werden. Das halte ich halt für Bullshit. Du musst halt eine vernünftige Trennung haben, damit du überhaupt die Möglichkeit hast, vernünftig und in Ruhe zu entwickeln. Und du brauchst halt dieses ganze Projektmanagement*

> *Zeug, dass das halt wirklich abgekapselt ist von der Entwicklung.“* (Interview 11, Absatz 96)

Des Weiteren arbeiten vier Personen gleichzeitig in unterschiedlichen Projekten. Mitarbeiter werden in Projekten je nach Bedarf umverteilt und müssen aufgrund dessen verschiedene Aufgaben erfüllen.

Drei Personen sprachen außerdem von *Crunch-* bzw. Arbeitsphasen, in welchen viel Stress und Zeitdruck herrscht. Vor allem gegen Ende eines Sprints oder Meilensteins nimmt der Stress zu, um noch so viele Features wie möglich fertigzustellen. Daher berichtete ein Proband auch von Überstunden in dieser Zeit. Eine andere Person beschrieb einen sehr starken Druck durch die hohen Erwartungen der Vorgesetzten in solchen Phasen, welcher unter Umständen abschreckend auf Mitarbeiter wirken könnte:

> *„Die Erwartungen sind viel zu hoch. Die Ergebnisse kommen dann oft nicht. [...] Und dann geht das halt los mit diesem Micromanagement Zeug, was am Ende aber auch nichts bringt, weil es ja den Mitarbeiter auch nicht schneller macht, sondern er hat halt nur mehr Druck. Und am Ende des Tages wird er sich dann zweimal überlegen, ob er das noch will oder nicht.“* (Interview 11, Absatz 96)

5.2.10 Arbeitsumgebung

Alle Teilnehmer gaben an, im Homeoffice arbeiten zu können. Daher sind, nach Angaben einiger Probanden, die Büros vor Ort auch regelmäßig wenig besucht:

> *„Und ja, es ist halt sehr ruhig, weil hier in [Stadt] sind die meisten Leute halt, weil die aus der Umgebung kommen, viele, ist die Auslastung halt nicht so groß [...]“* (Interview 2, Absatz 60)

Der Umfang, in welchem das Arbeiten von Zuhause aus erlaubt ist, schwankte. Drei Probanden nutzen ihr Kontingent an Homeoffice-Tagen nicht vollständig aus, sondern fahren gerne in ihr Büro, um dort zu arbeiten (vgl. Tabelle 4.2). Daher spielt die Arbeitsumgebung eine wichtige Rolle im Arbeitsalltag.

Keiner der Probanden arbeitet in Einzelbüros, auch wenn Meetingräume als Rückzugsort für Konzentrationsphasen genannt wurden. Viele arbeiten in Büros für mehrere Personen, häufig sind diese für zwei bis sechs Personen ausgelegt. Zudem berichteten zehn Personen von *Open Work Spaces*, also offenen Büroflächen mit Schreibtischen, welche genutzt werden können. Häufig wird nach dem *Shared-Desk-Prinzip* gearbeitet, sodass jeder Mitarbeiter sich an jeden freien Arbeitsplatz setzen kann. Dieser Büroaufbau stärkt die Kommunikation und sorgt für häufigere Zusammenarbeit:

> *„[...] wenn wir [im] Büro sind, setzen wir uns quasi nebeneinander, wir haben Open Workspace, [...], das heißt da ist man ein paar Sekunden quasi entfernt von einer Person, da läuft man auch gerne mal zu einem*

anderen Schreibtisch und macht auch irgendwas gemeinsam an einem PC, das kommt auch sehr häufig vor, dass wir da zu zweit vor einem PC setzen und da irgendwas entwickeln [...]“ (Interview 9, Absatz 52)

Drei Probanden berichteten außerdem von *Kreativbereichen* in ihren Büros. Diese sind für eine entspannte Art der Arbeit ausgelegt, um die Kreativität zu unterstützen. Sie sind in einem abgetrennten Bereich und mit Sofas ausgestattet und bieten beispielsweise Flip-Charts für kreative Phasen.

5.2.11 Vorstellungen der Entwickler

In diesem Abschnitt werden die Vorstellungen der Entwickler zu einzelnen Themen erfasst, die sie im Laufe der Interviews geäußert haben. Diese sind in Abbildung 5.9 dargestellt. Des Weiteren zeigt Abbildung 5.10 die Gesamtanzahl der Codes, die den Themen zugeordnet sind, sowie ihre prozentuale Verteilung.

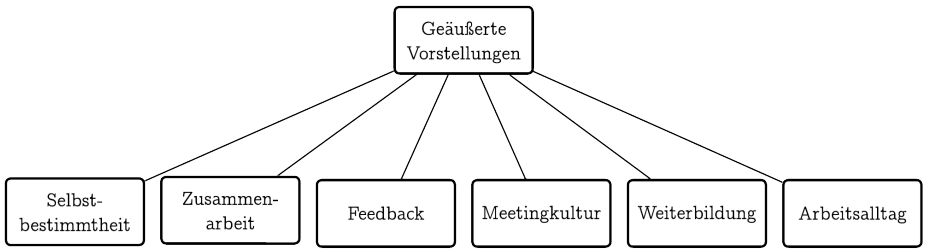

Abbildung 5.9: Die Vorstellungen der Entwickler zu ausgewählten Themen

Selbstbestimmtheit Im Hinblick auf ihre Selbstbestimmtheit äußerten die Entwickler verschiedene Vorstellungen. Zwei Probanden würden sich ihre Aufgaben gerne freier aus dem Backlog auswählen können, um an den Themen zu arbeiten, die sie interessieren. Vier Weitere würden ihre Aufgaben für den Tag gerne nach eigenem Ermessen erledigen. Auf diese Weise müssten sie seltener zwischen verschiedenen Aufgaben wechseln und könnten sich auf eine konzentrieren. Drei Interviewteilnehmer haben sich mehr Freiraum gewünscht, um kreativ über Thematiken nachzudenken. Sie müssten von einer zur nächsten Aufgabe arbeiten und könnten nicht über bessere oder skalierbare Lösungen nachdenken. Sieben Probanden äußerten außerdem den Wunsch, flexibler und häufiger aus dem Homeoffice heraus arbeiten zu können.

Eine Person wünschte sich bezüglich ihrer Arbeit weniger Freiheiten. Sie würde sich so viel Urlaub nehmen können, wie sie wolle. Allerdings würde im Umkehrschluss erwartet werden, dass sie ihren Rechner für Notfälle überall mit hinnehmen würde. Daher könne sie schlecht ihren Kopf von der Arbeit frei bekommen und würde sich striktere Regeln wünschen:

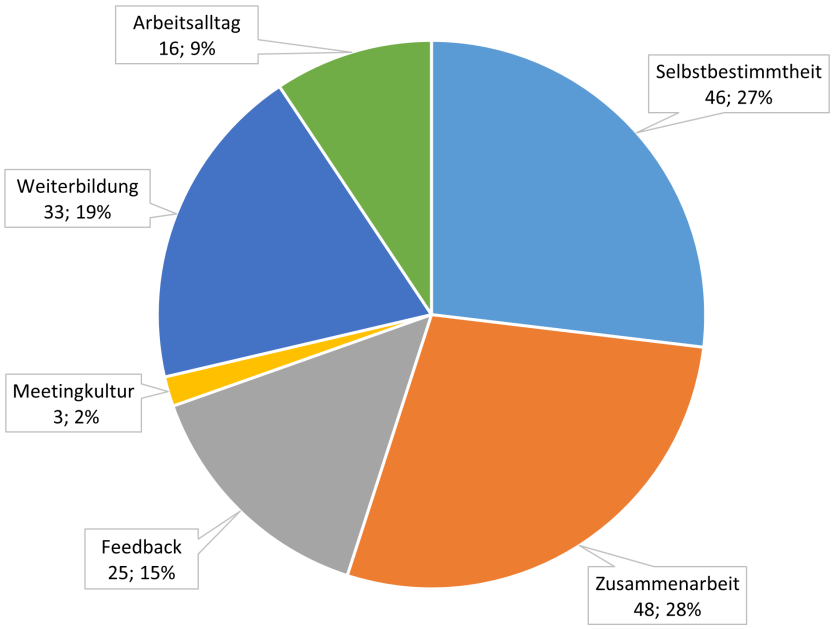

Abbildung 5.10: Die Aufteilung der gefundenen Codes auf die Themen

> *„Also eher andersrum. [...] Wir haben halt zum Beispiel eben das, dass wir eigentlich so viel Urlaub nehmen dürfen, wie wir wollen. [...] es wird dann natürlich trotzdem von dir erwartet, dass du den Laptop mitnimmst [...] Also ich bin eigentlich ganz gerne derjenige, der einfach abschalten will. Und wenn ich halt den Laptop mitnehme, ist das ein bisschen, fühlt sich sauer an irgendwie. Aber es ist halt die Firmenpolitik, sag ich mal.“* (Interview 3, Absatz 78)

Zusammenarbeit Nach den Vorstellungen der Entwickler ist eine direkte und klare Kommunikation und Zusammenarbeit sehr wichtig. Es soll ein Rahmen definiert werden, der zwanglose Zusammenarbeit unterstützt. Dennoch sollte eine gewisse Atmosphäre geschaffen werden, in welcher ein Austausch stattfinden kann. Dafür sei viel Transparenz wichtig, damit alle Beteiligten jederzeit wissen, wo sie gerade stehen und wie sie unterstützt werden können. Wichtig sei außerdem ein gemeinsames Verständnis über die Belange der Mitarbeiter und des Projekts. Hierfür würde sich laut vier Probanden die regelmäßige Dokumentation des aktuellen Standes anbieten. Des Weiteren sei die Arbeitsatmosphäre bestimmend über den Projektverlauf. Jedes Teammitglied soll harmonisch miteinander arbeiten und keinen schlechten Umgangston pflegen. Die Projektleitung soll Ratschläge der Entwickler beachten und zusammen mit ihnen Entscheidungen treffen. Ein

Proband empfand es als wichtig, dass ein Team selber entscheiden sollte, wie es am besten arbeite. Die Teamleitung soll dabei unterstützen und den Teammitgliedern Rückhalt bieten, ohne Einflüsse von außen zuzulassen.

Dadurch, dass viele Interviewteilnehmer in erster Linie aus dem Homeoffice arbeiten, berichteten fünf Personen, dass sie sich regelmäßige Präsenztermine wünschen würden:

> *„Ja, das ist jetzt auch wieder so ein Nachteil von Homeoffice. Ich finde es ehrlich gesagt immer ganz schön, wenn man seine Kollegen auch mal persönlich sieht, einfach nur um den Austausch zu vereinfachen [...]. Klar, so ein Teams-Meeting, da hat man immer so eine gewisse Hürde, um mal anzurufen [...]. Da ist das persönlich meistens deutlich einfacher.“* (Interview 13, Absatz 86)

Feedback Fast drei Viertel aller Teilnehmer äußerten zudem ihre Vorstellungen darüber, auf welche Weise sie gerne Feedback erhalten würden. Viele Probanden äußerten den Wunsch nach häufigerem, direktem Feedback. In erster Linie soll dies durch das Team zur eigenen Arbeit geschehen, allerdings wäre auch Feedback durch den Kunden über das Produkt erfreulich. Im Team kann dies durch einen Termin zur Präsentation der Arbeit geschehen, in welchem Rückmeldungen gesammelt werden können. Zudem wurde *Pair Programming* auch als eine Möglichkeit für direktes Feedback im Arbeitsalltag genannt. Ein Proband gab zudem an, dass häufigeres, ehrliches Feedback hilfreich sei:

> *„Das ist kein richtiger Mechanismus aber das man sich auch mal mehr traut, Feedback zu geben. Weil ganz oft nimmt man Sachen einfach nur hin, frisst Probleme in sich hinein und da ist so der offene und ehrliche Weg meistens der Beste. Das man sagt ‚Ey, das und das ist nicht so gut gelaufen, das müssen wir nächstes Mal besser machen‘.“* (Interview 13, Absatz 90)

Sieben Teilnehmer äußerten außerdem, dass sie gerne häufiger Feedback von der Unternehmens- und Projektleitung erhalten würden. Ein Proband würde gerne Entscheidungen der Geschäftsführung besser verstehen, da diese häufig realitätsfern seien. Ein Anderer wünschte sich mehr Informationen über den Projektfortschritt, um zu erfahren, was aktuell gut funktioniere und auf welche Weise das Team sich noch verbessern könnte. Mit dem Ziel, eine Einschätzung über die eigene Arbeitsweise und -geschwindigkeit zu erhalten gaben einige Probanden an, sich zusätzlich häufigere Feedbackgespräche mit Vorgesetzten zu wünschen.

Meetingkultur In Bezug auf die Meetingkultur äußerten die Entwickler zwei Vorstellungen. Zum einen sollte es so wenig wie möglich Pflichtmeetings geben, um seiner Arbeit nachkommen zu können. Jeder Mitarbeiter soll frei entscheiden können, ob eine Meetingteilnahme sinnvoll ist oder nicht. Zum

anderen wünschten sich zwei Probanden mehr oder intensivere *Socializing-Meetings*. Es soll möglich sein, sich Zeit zu lassen und zusätzlich über andere Themen als die Arbeit zu sprechen, um auf diese Weise das Zusammengehörigkeitsgefühl des Teams zu stärken.

Weiterbildung Im Bezug auf die eigene Weiterbildung in ihrem Unternehmen hatten die Interviewteilnehmer verschiedene Vorstellungen. Auch wenn mehrere Probanden zu verstehen gaben, dass Dokumentation häufig wenig nützlich für die Weiterbildung sei, wurde sie von zehn Personen als eine gute Option für Wissenstransfer genannt:

> *„Und das ist, finde ich, aber ein wichtiger Punkt für Wissensvermittlung, dass man es irgendwo festhält, gerade weil ja Wissen dann auch doch eher flüchtig ist.“* (Interview 5, Absatz 98)

Es soll eine Kultur etabliert werden, in der herausgefundene Informationen dokumentiert werden und existierende Dokumentation angepasst wird. Viele Mitarbeiter würden sich allerdings nicht trauen, Wissen zu dokumentieren:

> *„Das habe ich beobachtet, dass sich Kollegen häufig nicht trauen, Dokumentationen anzupassen, weil das hat ja irgendwer geschrieben und der muss ja schon irgendwo Recht gehabt haben.“* (Interview 5, Absatz 98)

Weitere fünf Personen sprachen davon, neues Wissen direkt und mündlich, beispielsweise in einem Meeting, mit Teammitgliedern zu teilen:

> *„Ansonsten ist der mündliche Weg natürlich etwas direkter. Man kann besser noch mal nachfragen.“* (Interview 5, Absatz 98)

Drei Personen sprachen sich außerdem für *Pair Programming* als Option für den Wissenstransfer aus. Auf diese Weise könnte das Team gemeinsam Probleme lösen und dabei Wissen teilen und kombinieren. Insgesamt soll ein *Austausch auf Augenhöhe* stattfinden und keine *Kopfmonopole* entstehen. Es soll für jeden Themenbereich verschiedene Ansprechpartner geben, welche auch in einem Austausch untereinander stehen sollten:

> *„Also der Prozess muss vorsehen, dass Wissen von allen Seiten kommt, egal wie jung oder alt man ist, egal wie neu oder wie lange man schon im Unternehmen ist. Ein Austausch, nicht von oben, ich bin Lehrer und stelle etwas vor, sondern wir tauschen uns wirklich aus.“* (Interview 22, Absatz 102)

Zuletzt wäre Mentoring eine sinnvolle Möglichkeit, um Mitarbeiter anzuleiten und ihnen neue Dinge beizubringen.

Arbeitsalltag Für ihren Arbeitsalltag sprachen die Probanden verschiedene Vorstellungen aus. Zwei Personen wünschten sich weniger Stress bzw. *Crunch-Phasen*, beispielsweise kurz vor einem Release. Zwei Weitere würden

sich wünschen, außerhalb der Arbeitszeit keine Anrufe entgegennehmen zu müssen, um beispielsweise auf Notfälle zu reagieren.

Ein Interviewteilnehmer artikulierte zudem die Vorstellung von mehr Sicherheitsnetzen in ihrem Prozess. Aktuell sei es so, dass viel Zeitdruck herrsche und Fehler nicht auffallen würden. Mehr Kontrollinstanzen wären sinnvoll, um Druck von den Entwicklern zu nehmen. Andere Teilnehmer äußerten Wünsche nach einer geringeren Stundenanzahl, Gleitzeit oder einem Überstundenkonto.

5.3 Vergleich zwischen Theorie und Praxis

Im Folgenden findet der Vergleich zwischen Theorie und Praxis der entwicklerzentrierten Softwareprozessattribute statt. Keine der interviewten Personen gab an, dass das in ihrem Unternehmen vorherrschende Vorgehensmodell nach Lehrbuch durchgeführt wird. Daher ist ein direkter Vergleich zwischen theoretischer Beschreibung einzelner Modelle in der Literatur und der praktischen Darstellung in den Interviews nicht zielführend, sodass darauf verzichtet wird. Stattdessen soll an dieser Stelle ein modellübergreifender Vergleich geschaffen werden.

In Tabelle 5.2 ist dieser Vergleich dargestellt. Dabei fällt auf, dass es eine große Überschneidung zwischen Theorie und Praxis gibt, da 20 von 25 EZSPA sowohl von der Theorie erwähnt als auch in der Praxis beobachtet wurden. In der Praxis berichten die Entwickler von fünf zusätzlichen Attributen, die ihren Arbeitsalltag erleichtern. Lediglich fünf der laut Lehrbuch zu findenden Attribute werden in der Praxis nicht oder nur eingeschränkt beobachtet. Dazu gehören Idealbilder wie „wenig Stress“ oder „selbstorganisierte Teams“, die in der Praxis oft schwer umzusetzen sind [34].

Die weiterführenden Vorstellungen bezüglich der EZSPA der Entwickler sind an dieser Stelle allerdings nicht aufgeführt, sondern in Unterabschnitt 5.2.11 zu finden. Die Vorstellungen entsprechen den Idealen der Probanden und sind dementsprechend nicht unbedingt in ihrem realen Softwareprozess aufzufinden.

Theorie	Theorie und Praxis	Praxis
Entscheidungsfreiheit*	Weiterentwicklung	Flexibilität
Eigene Zieldefinition*	Weiterbildung	Homeoffice
Selbstorganisiertes Team*	Feedback	Meetingkultur
Keine geteilten Aufgaben*	Wissenstransfer	Einarbeitung
Wenig Stress*	Mentoring	Arbeitsumgeb.
	Fortbildung	
	Unternehmensweites Lernen	
	Kontinuierliches Lernen	
	Zusammenarbeit	
	Transparenz	
	Unterstützung	
	Optimaler Informationsfluss	
	Klare Kommunikation	
	Interesse an den Anderen	
	Respekt	
	Wertschätzung	
	Umgang	
	Fehlerkultur	
	Mitarbeiterkontinuität	
	Gute Arbeitsbedingungen*	

* Diese entwicklerzentrierten Softwareprozessattribute werden nur teilweise erfüllt.

Tabelle 5.2: Auftreten der EZSPA in Theorie und Praxis

In der Literatur, besonders im Rahmen von Scrum, standen die prozessbezogenen EZSPA im Mittelpunkt. Ein Großteil der interviewten Personen berichtete von diesen Attributen in ihrem Prozess. Allerdings gaben auch jeweils neun Probanden an, dass sie wenig Entscheidungsfreiheit über ihre Aufgaben und ebenso wenig Einfluss auf die Zieldefinition ihres Sprints oder des Projekts haben. Viele Probanden können sich innerhalb ihres Teams selbst organisieren. Allerdings berichteten auch an dieser Stelle einige Teilnehmer von Weisungen durch Vorgesetzte. Außerdem gaben vier Personen an, gleichzeitig an mehreren Projekten zu arbeiten und drei Probanden sprachen von Stressphasen, beispielsweise kurz vor einem Release.

Bei den teambezogenen entwicklerzentrierten Softwareprozessattributen und den Attributen, welche sich auf die Weiterbildung beziehen, zeichnete sich ein einheitlicheres Bild ab. Sämtliche in der Literatur gefundenen Attribute ließen sich direkt oder indirekt auch in den Interviews wiederfinden. Beispielsweise beschrieben die Interviewteilnehmer sowohl die Zusammenarbeit als auch die Unterstützung im Team sehr facettenreich. Auch die Fehlerkultur und ein kontinuierlicher Lernprozess im Arbeitsalltag

wurde detailliert von den Probanden angeführt. Des Weiteren wurden einige Aspekte guter Arbeitsbedingungen aus der Literatur in den realen Softwareprozessen entdeckt. Dazu gehören die vielfach erwähnte Unterstützung durch Vorgesetzte oder eine stabile Teamkonstellation.

Neben den Übereinstimmungen zwischen Literatur und Praxis zeichneten sich durch die Interviews vielfältige entwicklerzentrierte Softwareprozessattribute ab, welche nicht in der Literatur zu finden waren. Die Probanden sprachen an dieser Stelle unter anderem von der Flexibilität im Arbeitsalltag. Dazu gehört, dass sie sich, bis auf einige Pflichttermine, ihren Arbeitstag und ihre Arbeitszeiten frei einteilen können. Die Möglichkeit, aus dem Homeoffice arbeiten zu können, ist ebenfalls ein oft erwähnter Aspekt der Flexibilität. Dabei ist für die Arbeit aus dem Homeoffice eine Formalisierung bestimmter Aspekte des Arbeitsalltag wichtig. Bestimmte Meetings dürften nicht mehr ausfallen, um alle Teammitglieder auf dem selben Informationsstand zu halten und um produktiv arbeiten zu können. Hinzu kommt die Meetingkultur, welche in der Literatur kaum direkt benannt worden ist. Diese spielte allerdings in den Interviews für viele entwicklerzentrierte Softwareprozessattribute eine zentrale Rolle. Meetings würden für Feedback, Weiterbildung aber auch Zusammenarbeit und die Einarbeitung von neuen Teammitgliedern genutzt werden. Des Weiteren spielte die Arbeitsumgebung eine Rolle für die Probanden. Neben neusten Arbeitsmitteln sei auch ein Open Work Space wichtig für die Zusammenarbeit und direkte Kommunikation im Team.

Im begrenzten Rahmen der Interviews konnten allerdings nicht alle aus der Literatur extrahierten Attribute überprüft werden. Für einen Gesamtüberblick sollten diese Aspekte in einer nachfolgenden Studie näher beleuchtet werden. Dabei handelt es sich um die folgenden EZSPA:

- Kontinuierliche Verbesserung
- Entscheidungsbefugnis
- Persönliche Erfüllung
- Crossfunktionale Teams
- Gute Arbeitsbedingungen (Gesundheit, Wohlbefinden, Nachhaltigkeit, Vertrauen)

Kapitel 6

Diskussion

In diesem Abschnitt werden die Forschungsfragen aus Abschnitt 4.2 beantwortet. Hierfür erfolgt zunächst die Definition der entwicklerzentrierten Softwareprozessattribute. Außerdem werden die Ergebnisse der Studie weiterführend interpretiert. Abschließend werden Limitierungen der durchgeführten Studie beleuchtet.

6.1 Beantwortung der Forschungsfragen

RQ1: Entwicklerzentrierte Prozessattribute nach Lehrbuch

RQ1

Welche entwicklerzentrierten Prozessattribute werden von weit verbreiteten Entwicklungsmethoden (Scrum, XP, DevOps, Spiralmodell und Wasserfall) erfüllt, wenn diese nach Lehrbuch ausgeführt werden?

Bevor die erste Forschungsfrage beantwortet werden kann, müssen die entwicklerzentrierten Softwareprozessattribute definiert werden. Dafür wurde auf Basis der in Abschnitt 4.3 definierten Inklusions- und Exklusionskriterien und den Ergebnissen der Literaturstudie folgende Definition für diese Arbeit entwickelt:

Definition: Entwicklerzentrierte Softwareprozessattribute

Entwicklerzentrierte Prozessattribute sind Eigenschaften eines Softwareprozesses, die den Entwickler in seiner täglichen Arbeit unterstützen oder dessen (Arbeits-)Leben bereichern sollen.

Die durchgeführte Literaturstudie zeigt, dass die entwicklerzentrierten Softwareprozessattribute viele verschiedene Facetten besitzen. Zahlreiche Aspekte der EZSPA hängen thematisch miteinander zusammen und unterstützen den Entwickler in verschiedenen Bereichen. Die extrahierten Attribute

lassen sich dabei vier verschiedenen Oberkategorien zuordnen. Eine genaue Auflistung der gefundenen entwicklerzentrierten Softwareprozessattribute ist in Abschnitt 5.1 zu finden.

Die erste Dimension der EZSPA stellen die persönlichen Aspekte dar. Hierunter fallen Attribute, welche den Entwickler in seiner täglichen Arbeit schützen und unterstützen sollen, beispielsweise die Work-Life-Balance. Die nächste Kategorie stellen die prozessbezogenen EZSPA dar. Diese schreiben dem Entwickler und seinem Team bestimmte Rechte im Prozess zu. Dazu gehört speziell eine eigene Zieldefinition und die Entscheidungsfreiheit, wie dieses Ziel erreicht werden soll. Darüber hinaus wurden teambezogene Attribute in der Literatur identifiziert. Sie beschreiben Aspekte wie die Zusammenarbeit oder eine klare Kommunikation im Team. Die letzte Kategorie der entwicklerzentrierten Softwareprozessattribute bezieht sich auf die Weiterentwicklung der Mitarbeiter. An dieser Stelle findet sich beispielsweise die Weiterbildung der Entwickler im Arbeitsalltag, nicht zuletzt durch Mentoring oder einen Wissenstransfer unter Teammitgliedern.

RQ2: Entwicklerzentrierte Prozesssattribute in realen Prozessen

RQ2

Welche entwicklerzentrierten Prozessattribute finden sich in realen Softwareprozessen?

Den Interviews nach zu urteilen gestalten sich die entwicklerzentrierten Softwareprozessattribute in realen Softwareprozessen mindestens ebenso vielschichtig wie in der Literatur. Die Interviews zeigten auch, dass viele der EZSPA miteinander zusammenhängen und sich gegenseitig beeinflussen und unterstützen. Die im Zuge dieser Arbeit erfassten entwicklerzentrierten Softwareprozessattribute lassen sich in zehn Themen mit insgesamt 31 Subthemen zusammenfassen. Eine Darstellung aller in der Praxis gefundenen EZSPA ist in Abschnitt 5.2 zu finden.

Die Ergebnisse der Interviewstudie zeigen, dass es in der Praxis unter anderem entwicklerzentrierte Softwareprozessattribute in den Bereichen der Selbstbestimmtheit, der Fehlerkultur und der Arbeitsumgebung gibt. Außerdem spielt die Meetingkultur für die EZSPA eine zentrale Rolle. Sie unterstützt bei der Zusammenarbeit, der Weiterbildung von Mitarbeitern sowie der Vergabe von Feedback. Dabei wurde insbesondere die Zusammenarbeit als EZSPA sehr facettenreich von den Probanden der Interviewstudie beschrieben und als wichtig erachtet. Darüber hinaus wurden entwicklerzentrierte Softwareprozessattribute gefunden, welche sich auf die Einarbeitung und die Teamstabilität bezogen.

RQ3: Vergleich zwischen Theorie und Praxis

RQ3

Inwiefern decken sich die aus der Literatur extrahierten Prozessattribute mit den Vorstellungen der Entwickler?

Der in Abschnitt 5.3 beschriebene Vergleich zwischen Theorie und Praxis zeigte große Übereinstimmungen. Viele der in der Literatur präsentierten entwicklerzentrierten Softwareprozessattribute ließen sich auch in der Praxis wiederfinden. Insbesondere die Attribute der Weiterbildung und die teambezogenen Attribute zeigten sich auch in der Praxis als erfüllt. Allerdings beschrieben die Probanden die prozessbezogenen EZSPA sehr unterschiedlich. Die Mehrheit der 22 Interviewteilnehmer gab an, im Arbeitsalltag freie Entscheidungen bezüglich ihrer Aufgaben treffen zu können und auch bei der Zieldefinition mitwirken zu können. Allerdings führten auch jeweils neun Personen an, dass das in ihrem Fall nicht möglich wäre und Vorgesetzte über diese Aspekte entscheiden würden.

Abgesehen von den Übereinstimmungen zwischen Literatur und realen Softwareprozessen beschrieben die Entwickler auch EZSPA, welche nicht in der Literatur zu finden waren. Beispielsweise beschrieben sie eine flexible Arbeitsweise, in welcher sie sich ihre Arbeitszeit und die Aufgaben eigenständig einteilen könnten. Außerdem führten sie die Meetingkultur als wichtigen Punkt ihrer entwicklerzentrierten Softwareprozessattribute an. Sie stellt den Mittelpunkt für viele weitere EZSPA dar, welche für die Entwickler eine wichtige Rolle spielen. Wie bereits in bei der Beantwortung von RQ2 beschrieben, wirken Meetings für viele Aspekte der entwicklerzentrierten Softwareprozessattribute verstärkend und bieten den Entwicklern so selber auch Unterstützung in ihrer täglichen Arbeit.

Darüber hinaus gaben die Praktiker auch Vorstellungen über die EZSPA preis, welche über die Literatur und die Beschreibungen ihres eigenen Prozesses hinausgehen. Eine detaillierte Auflistung dieser Attribute ist in Unterabschnitt 5.2.11 zu finden. Die Probanden beschrieben an dieser Stelle eine kreative Arbeitsweise als sehr wichtig, in welcher Aufgaben frei und mit genug Zeit bearbeiten werden können, um auch über tiefergehende Lösungen nachdenken zu können. Für eine verbesserte Zusammenarbeit wünschten die Teilnehmer sich zusätzlich, dass sie frei über eine Meetingteilnahme entscheiden könnten und dass es mehr Socializing-Meetings geben sollte. Demnach kann das Team auf diese Weise enger zusammenwachsen und -arbeiten. Transparenz und eine angenehme Arbeitsatmosphäre sind an dieser Stelle von zentraler Bedeutung, um die Teammitglieder optimal unterstützen zu können.

6.2 Interpretation der Ergebnisse

Im folgenden Abschnitt sollen einige Beobachtungen in Verbindung mit den Ergebnissen der durchgeführten Studien beschrieben werden.

Hohe Entwicklerzentriertheit

Insgesamt wurden sowohl in der Literatur, als auch in der Interviewstudie viele unterschiedliche Aspekte der entwicklerzentrierten Softwareprozessattribute identifiziert. Im direkten Vergleich zwischen Theorie und Praxis ließen sich dabei zahlreiche Überschneidungen finden. Auch wenn noch weiterführende Vorstellungen der Entwickler erfasst worden sind, spricht das in dieser Arbeit erzielte Ergebnis bereits für eine hohe Entwicklerzentriertheit moderner, praxisrelevanter Softwareprozesse.

Scrum: Wenig Entwicklerzentriertheit in der Literatur

Über die Hälfte der Interviewteilnehmer gab an, einen auf Scrum basierenden Prozess in ihrem Projekt zu nutzen (vgl. Tabelle 4.4). Dabei weist Scrum mit 28 gefundenen entwicklerzentrierten Softwareprozessattributen die wenigsten Attribute der untersuchten Prozesse auf (vgl. Abschnitt 5.1). Das zeigt, dass Scrum den Entwickler verglichen mit DevOps und XP in weniger Bereichen des Arbeitsalltag unterstützt. Zudem fanden sich in der Literatur viele prozessbezogene Attribute, welche den Entwicklern Freiheiten und Befugnisse im Prozess zusprechen und weniger, die sich auf das Team oder Weiterbildungsmöglichkeiten beziehen. Das könnte darin begründet sein, dass es sich bei Scrum eher um ein Vorgehensmodell des Projektmanagements handelt. Dieses wird im Anschluss mit Methoden und Praktiken ergänzt, um es für ein Softwareprojekt anzupassen.

Entwicklerzentrierte Softwareprozessattribute in älterer Literatur

Wie bereits in Abschnitt 5.1 beschrieben, ließen sich weder in *„A Spiral Model of Software Development and Enhancement“* [5] noch in *„Managing the Development of large Software Systems: Concepts and Techniques“* [47] entwicklerzentrierte Softwareprozessattribute finden. Dieser Umstand ist wahrscheinlich auf die frühe Zeit der Softwareentwicklung, in welcher diese Arbeiten veröffentlicht worden sind, zurückzuführen. Vor dem Jahrtausendwechsel und dem agilen Manifest wurden die Entwickler nur wenig bei der Softwareprozesskonstruktion berücksichtigt. Erst mit dem Aufkommen der agilen Methoden rückten diese in den Fokus der Softwareentwicklung und wurden dementsprechend in den Prozess miteinbezogen [19].

Schwierigkeiten durch unzureichende Beachtung der entwicklerzentrierten Prozessattribute

Im Rahmen der Interviewstudie kamen verschiedene Probleme im Bezug auf die analysierten EZSPA auf. Ein Teil der in Abschnitt 5.2 beschriebenen Probleme soll an dieser Stelle kurz erwähnt und diskutiert werden.

Ein Problem stellt die durch das Homeoffice erschwerte Zusammenarbeit dar. Dadurch, dass viele Teammitglieder von Zuhause aus arbeiten, ist es umso wichtiger, dass regelmäßig Austausche stattfinden. Auf diese Weise kann auch aus dem Homeoffice eine gute Zusammenarbeit gewährleistet werden.

Insgesamt sechs Interviewteilnehmer sprachen außerdem an, dass sie wenig Feedback erhalten würden. Allerdings stellt Feedback eine einfache Art und Weise dar, Informationen zu teilen und so zur Weiterentwicklung der Entwickler beizutragen. Auch die Retrospektive wurde häufig im Rahmen der Weiterentwicklung genannt. Drei der Probanden berichteten, dass diese bei ihnen häufig sehr kurz wäre oder ganz entfalle. Zwei Personen berichteten sogar, dass in ihrem Unternehmen keine Schulungen zu für sie relevante Themen durchgeführt würden.

Andere Interviewteilnehmer sprachen davon, dass sie viele Aufgaben innerhalb eines Sprints zugewiesen bekämen. Diese könnten sie in den seltensten Fällen alle erledigen und müssen sie, zusätzlich zu ihren neuen Aufgaben, im nächsten Sprint bearbeiten. Das führt dazu, dass sie unmotiviert in ihren Arbeitstag starten und nicht so produktiv sind, wie sie es sein könnten.

Des Weiteren sprachen einige Probanden davon, dass sie entweder in mehreren Projekten mitarbeiten oder mehrere Rollen in einem Projekt bekleiden würden. Dieser Aspekt sorgt unweigerlich dafür, dass die Entwickler nicht ihre vollständige Energie in ein Projekt stecken und daher nicht vollständig produktiv sein können.

6.3 Limitierung der Ergebnisse

An dieser Stelle wird die Validität der durchgeführten Studien diskutiert. Für diesen Zweck unterscheiden Wohlin et al. [61] zwischen der *Reliability*, der *Construct Validity* sowie der *Internal* und *External Validity*, um die Verlässlichkeit der Ergebnisse zu überprüfen. Es werden sowohl die Literaturstudie als auch die Interviewstudie auf diese Aspekte der Validität untersucht.

6.3.1 Construct Validity

Die *Construct Validity* beschreibt, ob die erfassten Daten tatsächlich das repräsentieren, was gemäß den Forschungsfragen untersucht werden soll [61].

Vor Beginn der Literaturstudie wurden die in Abschnitt 4.3 beschriebenen Inklusions- und Exklusionskriterien definiert. Das geschah mit dem Ziel, sicherzustellen, dass nur Textpassagen extrahiert werden, welche die Kriterien der entwicklerzentrierten Softwareprozessattribute erfüllen und mit ihnen die erste Forschungsfrage beantwortet werden kann.

Außerdem wurden die extrahierten Prozessattribute als Basis für den Entwurf des Interviewleitfadens genutzt. Auf diese Weise sollte ermöglicht werden, die zweite Forschungsfrage mithilfe der Interviews beantworten zu können.

Mit dem Ziel, ein einfaches Verständnis der Interviewfragen zu gewährleisten und sicherzustellen, dass die Erklärungen der Probanden zur Beantwortung der zweiten Forschungsfrage beitragen können, wurde der Leitfaden mit zwei Bachelorstudenten der Informatik und einem wissenschaftlichen Mitarbeiter des Fachgebiets für Software Engineering getestet. Außerdem konnte sowohl der Interviewer, als auch die interviewte Person, jederzeit Rückfragen stellen, um Unklarheiten zu lösen.

Zu Beginn wurden die Probanden gebeten, ihren Arbeitsalltag und damit ihren Entwicklungsprozess zu beschreiben. Das geschah, um einen Einblick ohne Bias in die entwicklerzentrierten Softwareprozessattribute der Teilnehmer zu erhalten. Da die EZSPA allerdings ein breites Forschungsfeld darstellen, musste sich im Anschluss auf die wichtigsten Aspekte fokussiert werden. Diese Fokussierung kann zu einem Bias in den Ergebnissen geführt haben. Wenngleich auf diese Weise die Aussagekraft für alle EZSPA nicht gewährleistet ist, ermöglichte die Einschränkung detaillierte Einblicke in ausgewählte EZSPA. Die hier nicht näher beleuchteten Attribute sollten in künftiger Forschung im Detail analysiert werden.

6.3.2 Internal Validity

Die *Internal Validity* beschreibt, ob sich die beobachteten Ergebnisse und Zusammenhänge tatsächlich auf die Befragung beziehen oder diese durch Störfaktoren von außen beeinflusst worden sind [61]. Da die Forschung in dieser Arbeit explorativer und qualitativer Natur ist, liegt eine Beeinträchtigung der *Internal Validity* vor.

Die im Rahmen der Interviews erfassten Daten beschreiben die subjektive Wahrnehmung der Probanden. Es kann nicht überprüft werden, ob und inwiefern die Teilnehmer mögliche Einflussfaktoren bei ihrer Antwort berücksichtigt haben. Außerdem wurde mit semistrukturierten Interviews gearbeitet, sodass sich die Antworten in den Interviews aufgrund ihrer teilweise offenen Natur nur schwer in einen direkten Zusammenhang setzen lassen. Diesem Aspekt wurde versucht entgegenzuwirken, indem für die Auswertung eine thematische Analyse genutzt wurde. Auf diese Weise konnten die Aussagen der Teilnehmer abstrahiert und relevante Themen herausgearbeitet werden. Es ist zudem möglich, dass ein Bias durch das

Verhalten und die Fragestellungen des Interviewers entstanden ist. Mit dem Ziel, diese Gefahr zu minimieren, wurde bereits im Vorfeld ein Leitfaden für semistrukturierte Interviews entworfen, um, trotz möglicher Rückfragen, für einen möglichst unvoreingenommenen und standardisierten Interviewprozess zu sorgen. Die Interviewfragen wurden offen und neutral gestellt, um Suggestivfragen zu vermeiden und die Teilnehmer nicht zu einer bestimmten Antwort zu drängen [43]. Des Weiteren kann ein sogenannter *self-reporting Bias* vorliegen [41]. Möglicherweise sprachen die Probanden eher von einer idealisierten Vorstellung ihres Entwicklungsprozesses und nicht von der Realität in ihrem Projekt.

6.3.3 External Validity

Die *External Validity* beschreibt, inwiefern die Ergebnisse einer Studie verallgemeinert werden können [61]. Im Rahmen der Literaturstudie wurden lediglich fünf Primärliteraturwerke zu Vorgehensmodellen untersucht. Die HELENA-Studie [28] zeigte allerdings, dass weitere Vorgehensmodelle in der Praxis eingesetzt werden. Diese konnten aufgrund des begrenzten zeitlichen Rahmens dieser Masterarbeit nicht untersucht werden. Des Weiteren erheben die in der Literatur gefundenen entwicklerzentrierten Softwareprozessattribute keinen Anspruch auf Vollständigkeit. Daher kann nicht davon ausgegangen werden, dass alle relevanten Aspekte der EZSPA identifiziert worden sind. Die Ergebnisse der Literaturstudie lassen sich daher nicht ohne Einschränkungen auf andere Vorgehensmodelle übertragen. Sie sollen einen ersten Überblick über mögliche Attribute liefern, sowie die ersten Anhaltspunkte für die Interviewstudie geben.

Auf Basis der kleinen Stichprobe mit 22 Interviewteilnehmern und der qualitativen Natur von Interviews ist es nicht möglich, die erhaltenen Ergebnisse zu generalisieren. Außerdem könnte ein *Selection Bias* vorhanden sein. Die Teilnehmerakquise fußte in erster Linie auf der Ansprache persönlicher Kontakte. Hinzu kommt die Verlosung eines 20€-Gutscheins unter den Teilnehmern. Probanden könnten dies als Grund zur Teilnahme an der Studie gesehen haben, ohne tiefe Einblicke in das Thema geben zu können. Allerdings wurde der Betrag mit Absicht gering gewählt, sodass dieser nur als Aufwandsentschädigung dient. Es wurden zudem ausschließlich Personen aus dem Bereich der Softwareentwicklung interviewt. Dabei hatten die meisten Teilnehmer weniger als fünf Jahre Berufserfahrung. Lediglich eine Person wies mehr als zehn Jahre Erfahrung in der Industrie auf. Hingegen konnten Entwickler aus Unternehmen unterschiedlicher Sektoren gewonnen werden, was die externe Validität aufgrund verschiedener Perspektiven stärkt.

6.3.4 Reliability

Die *Reliability* beschreibt, inwieweit die Daten und die Analyse von den involvierten Forschern abhängt [61]. Die im Rahmen der Literaturstudie extrahierten Codes wurden, wie in Abschnitt 4.3 beschrieben, von einer Forscherin mit mehrjähriger Erfahrung im Bereich der Softwareprozesse und dem Autor dieser Arbeit zu zweit codiert. Auf diese Weise wurde die Reliability der Ergebnisse gestärkt.

Die Codierung und die thematische Analyse der Interviews wurden alleine durch den Verfasser dieser Arbeit durchgeführt. Dieser Umstand erschwert die Reproduzierbarkeit der Interviewstudie. Die Auswertung von Interviews ist auch mit einer thematischen Analyse ein sehr subjektiver Prozess und es ist nicht garantiert, dass andere Forscher auf die gleichen Ergebnisse gekommen wären. Mit dem Ziel, belastbarere Schlussfolgerungen zu erhalten, wäre auch an dieser Stelle ein Codierungsprozess durch mehr als eine Person wertvoll gewesen.

Allerdings unterstützt die im Rahmen dieser Arbeit genau beschriebene Planung und Durchführung der Studie die Reproduzierbarkeit. Insbesondere der Interviewleitfaden und die Definition der Zielgruppe kann für diesen Zweck herangezogen werden.

Kapitel 7

Zusammenfassung und Ausblick

Dieses letzte Kapitel schließt die Arbeit ab. Die in den vorangegangenen Kapiteln gewonnenen Ergebnisse und Erkenntnisse werden noch einmal zusammengefasst dargestellt. Außerdem werden mögliche Richtungen künftiger Forschung vorgeschlagen und diskutiert.

7.1 Zusammenfassung

Das Ziel dieser Arbeit war es, einen Vergleich der entwicklerzentrierten Softwareprozessattribute in Theorie und Praxis zu schaffen. Diese Attribute beschreiben Eigenschaften eines Prozesses, welche dem Entwickler in seinem Arbeitsalltag dienlich sind und dessen Arbeitsleben bereichern sollen.

Dafür wurde zu Beginn eine Literaturstudie auf Basis von Primärliteratur zu häufig in der Praxis genutzten Vorgehensmodellen durchgeführt. Die Absicht dieser Studie war es, eine Datenbasis für die anschließende Definition der entwicklerzentrierten Softwareprozessattribute zu schaffen. Außerdem dienten die Ergebnisse als Grundlage für die Planung der nachfolgenden Interviewstudie und den Vergleich zwischen Theorie und Praxis. Es wurde eine Vielzahl von entwicklerzentrierten Softwareprozessattributen identifiziert, welche die untersuchten Prozesse nach ihrem jeweiligen Lehrbuch bieten. Diese reichten von Entscheidungsbefugnissen der Entwickler in ihrem Prozess, über Maßnahmen zur Weiterbildung, bis hin zu Aspekten der Zusammenarbeit im Team.

Die darauffolgende qualitative Interviewstudie mit 22 Praktikern diente zum einen der Erfassung entwicklerzentrierter Softwareprozessattribute in realen Softwareprozessen. Zum anderen wurde sie genutzt, um weiterführende Vorstellungen der Praktiker über die EZSPA festzustellen. Mit dem Ziel, die relevanten Aspekte der Interviews zu extrahieren und zu aggregieren, wurden die Daten mithilfe einer thematischen Analyse ausgewertet. Dabei

ergab sich eine facettenreiche Zusammenstellung an Themen, welche für die Probanden entwicklerzentrierte Softwareprozessattribute darstellen. Unter anderem gaben die Teilnehmer an, dass für sie neben einer selbstbestimmten Arbeitsweise auch die Zusammenarbeit im Team wichtig sei. Weitere EZSPA ließen sich in den Bereichen der Weiterbildung, Meetingkultur oder auch der Arbeitsumgebung identifizieren.

Der abschließende Vergleich zwischen Theorie und Praxis der entwicklerzentrierten Softwareprozessattribute zeigte, dass bereits viele der in der Literatur nachgewiesenen Attribute auch in realen Softwareprozessen zu finden sind. Demnach berichteten beispielsweise alle Probanden von einer lösungsorientierten Zusammenarbeit im Team, welche auch in der Literatur eine Rolle spielt. Allerdings äußerten die Interviewteilnehmer auch Perspektiven über die EZSPA, welche über die Beschreibungen der Literatur hinausgehen. Hier kann exemplarisch der Wunsch nach mehr Raum für kreative Arbeitsphasen, oder die Möglichkeit selbstständig aus dem Homeoffice zu arbeiten, angeführt werden.

7.2 Ausblick

Die entwicklerzentrierten Softwareprozessattribute stellen ein neues Forschungsfeld mit viel Potential für weitere Arbeiten dar. Während in dieser Arbeit bereits tiefgehende Einblicke in die entwicklerzentrierten Softwareprozessattribute in Theorie und Praxis erarbeitet worden sind, kann die Forschung in diesem Bereich an vielen Stellen fortgeführt werden. Nachfolgend werden einige potenzielle Ansätze für zukünftige Forschung im Bereich der entwicklerzentrierten Softwareprozessattribute vorgestellt.

Untersuchung weiterer Vorgehensmodelle

Im Kontext dieser Arbeit wurden fünf Primärliteraturwerke über Vorgehensmodellen untersucht. Die HELENA-Studie [28] zeigte allerdings auf, dass weitere Prozesse in der Praxis angewendet werden, welche im begrenzten Rahmen dieser Arbeit nicht untersucht werden konnten. Die Lehrbücher, in welchen diese Prozesse beschrieben werden, könnten weitere, noch nicht erfasste, entwicklerzentrierte Softwareprozessattribute aufweisen. Daher kann ein Ansatz weiterer Forschung die Untersuchung verbliebener Literatur sein, um einen umfassenderen Überblick der EZSPA zu erhalten.

Außerdem werden Prozesse selten nach Lehrbuch ausgeführt [56]. In vielen Unternehmen existieren hybride Ansätze, in welchen unterschiedliche Modelle kombiniert zum Einsatz kommen [56]. Die Analyse von Sekundärliteratur mit weiterführenden Interpretationen der ursprünglichen Ideen kann hilfreich sein, um einen praxisorientierten, tiefgründigen Einblick in die entwicklerzentrierten Softwareprozessattribute zu erhalten.

Des Weiteren gab über die Hälfte der Interviewteilnehmer an, nach Scrum zu arbeiten. Zusätzlich zu Literatur über praxisrelevante Vorgehensmodelle könnten auch Interviews mit Entwicklern, die nach diesen Modellen arbeiten, wertvolle praktische Einblicke liefern. Als Basis könnte die, in dieser Arbeit entwickelte und ausführlich dokumentierte, Interviewstudie und der dazugehörige Leitfaden genutzt werden.

Weiterführende Datenerhebung

Für die durchgeführte Literaturstudie wurden Inklusions- und Exklusionskriterien definiert, welche auf die Textpassagen angewendet wurden, um diese als potentielle EZSPA zu klassifizieren. Allerdings kann ein subjektiver Bias bei der Auswahl der Passagen nicht vollständig ausgeschlossen werden. Eine erneute Bearbeitung der Literaturpassagen mit mehr als einem Forschenden kann dabei helfen, diesen zu vermindern.

Des Weiteren konnten im Rahmen der Interviewstudie nicht alle aus der Literatur extrahierten entwicklerzentrierten Softwareprozessattribute überprüft werden (vgl. Abschnitt 5.3). Für einen Gesamtüberblick sollten diese in einer Folgestudie noch einmal umfassender beleuchtet werden.

Mithilfe von (Online-)Umfragen könnte eine deutlich größere Zielgruppe angesprochen werden, als es im Rahmen von einer Interviewstudie möglich ist. Im Zuge dessen könnten tiefergehende Details über die Auffassungen der Entwickler zu den EZSPA ihres Prozesses und ihren weiterführenden Vorstellungen darüber aufgenommen werden.

Entwicklerzentrierte Prozesskonstruktion

Ein langfristiges Ziel kann die Konstruktion und Evaluation eines Softwareprozesses mit expliziter Berücksichtigung der entwicklerzentrierten Softwareprozessattribute darstellen. Da diese Aspekte allerdings bei der aktuellen Prozesskonstruktion noch nicht berücksichtigt werden, bedarf dieser Schritt einer tiefgreifenden Planung und Anpassung momentaner Prozesse. Es müssen Metriken bzw. bestimmte Key Performance Indicators (KPIs) definiert werden, auf Basis derer die aktuelle Form eines Prozesses sowie die, mithilfe der entwicklerzentrierten Softwareprozessattribute, angepasste Version evaluiert werden kann. Die Bewertung von entwicklerzentrierten Metriken stellt allerdings eine Herausforderung dar, weil sich diese Metriken lediglich subjektiv, unter Einbeziehung der Menschen, überprüfen lassen.

Außerdem müssen die Prozessmetamodelle aktualisiert werden, um gegebenenfalls neue Prozesselemente und Elementtypen einzupflegen. Es muss überprüft werden, auf welche Weise das Prozessmodell adaptiert werden kann und welche Teile überhaupt verändert werden müssen, um das gewünschte Ziel zu erreichen. Des Weiteren müssen auch die Grenzen der möglichen Anpassung beachtet werden.

Menschenzentrierte Softwareprozessanforderungen

Eine weitere interessante Forschungsperspektive besteht darin, die entwicklerzentrierten Softwareprozessattribute auf andere involvierte Personengruppen auszudehnen. Denn neben den Entwicklern sind auch andere Menschen in den Prozess involviert, wie beispielsweise die Projektleitung oder der Kunde. Diese haben, ebenso wie das Entwicklungsteam, Rechte, welche im Rahmen des Prozesses beachtet werden müssen. Hier kann etwa die Einbindung des Kunden in den Entwicklungsprozess, zum Beispiel in Form des Product Owners, angeführt werden. Mit dem Ziel alle involvierten Gruppen zu berücksichtigen, müssen zuerst die Bedürfnisse und Vorstellungen dieser Personen erfasst werden. Das ermöglicht im nächsten Schritt die Berücksichtigung im Prozess.

Außerdem kann untersucht werden, auf welche Weise die Softwareprozessattribute zu Anforderungen erweitert werden können. Für diese *menschenzentrierten Softwareprozessanforderungen* muss ein systematischer Ansatz entwickelt werden, wie aus Prozessattributen Anforderungen gestaltet werden können. Hierdurch wird eine Einbindung der menschenzentrierten Anforderungen in den Prozess ermöglicht.

Anhang A

Interviewleitfaden

Auf den folgenden Seiten ist der Leitfaden abgebildet, welcher für die Durchführung der Interviews im Rahmen dieser Arbeit verwendet worden ist. In Unterabschnitt 4.4.2 ist er detailliert beschrieben worden. Dabei ist das Dokument in die Abschnitte *Einleitung, Demografie, RQ3, RQ4 und Schluss* unterteilt. Aufgrund einer Anpassung der Forschungsfragen im Laufe der Arbeit ist die hier mit **RQ3** betitelte Forschungsfrage in der Arbeit **RQ2**. Die Forschungsfrage **RQ4** im Leitfaden wird in der Arbeit mit **RQ3** betitelt.

Leitfaden für die Interviewstudie zu entwicklerzentrierten Prozessqualitätsattributen

Einleitung

□ Begrüßung

□ Für die Teilnahme bedanken

□ Verlosung am Ende (Teilnahme auch bei Abbruch möglich)

□ Persönliche Vorstellung

- Lukas Köhler
- Informatik-Masterstudent an der Leibniz Universität Hannover
- Studie für meine Masterarbeit

□ Vorstellung des Themas

- Titel der Arbeit: „Vergleich von entwickler-zentrierten Prozessqualitätsanforderungen in Theorie und Praxis“
- Ziel: Herausfinden, welche entwickler-zentrierten Qualitätsattribute Prozesse in realen Softwareprojekten haben. Außerdem soll untersucht werden, welche Vorstellungen von diesen (entwickler-zentrierten Prozessqualitätsattribute) Entwickler in Bezug auf ihren Entwicklungsprozess haben
  - Beispiele: Gute Arbeitsbedingungen, Work-Life-Balance
  - Langfristiges Ziel: Entwicklern helfen

□ Gibt es bis hierhin Fragen?

□ Vorstellung des Ablaufs

- Zu Beginn: Warm-Up und demografische Fragen
- Anschließend Hauptteil und dann Schluss

□ Um ehrliche Antworten bitten

- Kein Richtig/Falsch
- Interessant sind persönliche Erfahrungen und Ansichten
- Zugrundeliegende Strukturen sind interessant

□ Interview wird aufgezeichnet

- Transkribiert und pseudonymisiert
- Ergebnisse nur pseudonymisiert veröffentlicht
- Kein Rückschluss von Dritten auf die Person möglich
- Die Teilnahme ist freiwillig
  - Abbruch ist jederzeit möglich
  - Sollten spezielle Fragen nicht beantwortet werden wollen/können/dürfen, ist das in Ordnung

□ Angepeilte Dauer: 45 Minuten

- Auch abhängig von der Ausführlichkeit der Antworten

□ Noch Fragen, bevor es losgeht?

□ Einverständniserklärung (Teilnahme + Aufnahme)

□ Aufnahme wird gestartet

□ Aufnahme starten

□ Erneut Einverständniserklärung

Demografie

□ 1. Welche Rolle haben Sie inne und welche Aufgaben übernehmen Sie?

□ 2. Wie lange arbeiten Sie schon in der Softwareentwicklung?

- □ 2.1. Und wie lange in der aktuellen Rolle?

□ 3. Wie groß ist das Unternehmen, in welchem Sie arbeiten?

□ 4. In welchem Sektor ist es angesiedelt?

□ 5. Wie groß ist das Projekt, in dem Sie aktuell arbeiten?

- □ 5.1. Wie lange laufen die Projekte durchschnittlich, in denen Sie arbeiten? Wie lange das aktuelle?
- □ 5.2. Wie kritisch ist das Projekt, in dem Sie arbeiten? (ISO-Normen) => Was passiert, wenn die Software ausfällt?
  - Finanzieller Verlust fürs Unternehmen?
  - Sind Menschenleben in Gefahr?

□ 6. Wie viele Personen sind in Ihrem Team?

□ 7. Arbeiten Sie im Büro oder können Sie Homeoffice machen?

- □ 7.1. Wie groß ist der Homeoffice-Anteil?

Notizen

_____

_____

_____

_____

_____

Hauptteil; RQ3: Welche entwickler-zentrierten Prozessqualitätsattribute haben reale Softwareprozesse?

□ 8. Bitte beschreiben Sie mir so ausführlich wie möglich Ihren Alltag im Entwicklungsprozess. (Organisatorischer Rahmen für den Ablauf eines Projekts (bspw. das Wasserfallmodell oder Scrum))

- □ 9. Was ist das vorherrschende Entwicklungs- bzw. Prozessframework (wie bspw. Wasserfallmodell/Scrum)?
- □ 10. Wie sieht Ihr typischer Arbeitsalltag aus?
- □ 11. Wie läuft ein typischer Sprint ab? (Oder Ablauf bis QG etc.)
- □ 12. Wie erhalten Sie Ihre Aufgaben (Selbst/Vorgesetzte)?
- □ 13. Wie wird darüber entschieden, woran als Nächstes gearbeitet wird?
- □ 14. Wie erhalten Sie Feedback zu Ihrer Arbeit/der Aufgabe?
  - □ 14.1. Von wem?
  - □ 14.2. Auf welchem Weg (mündl./schriftl./Mail)?
- □ 15. Wie wird Ihrer Erfahrung nach in Ihrem Unternehmen mit Fehlern umgegangen?

Notizen

_____

_____

_____

_____

_____

□ 16. Wie arbeiten Sie mit Teammitgliedern zusammen?

- □ 16.1. Wie mit anderen Kolleginnen und Kollegen?
- □ 16.2. Auf welchen Wegen kommunizieren Sie?
- □ 16.3. Wie unterstützen Sie sich gegenseitig?
- □ 16.4. Wie wichtig ist Ihnen die Stabilität Ihres Teams?
- □ 16.5. Wie verläuft die Einarbeitung eines neues Teammitglieds?

□ 17. Welche (regelmäßigen) Meetings haben Sie?

- □ 17.1 Über welche Themen werden in diesen Meetings gesprochen?

□ 18. Wie können Sie sich fort- und weiterbilden?

- □ 18.1. Welche Angebote stehen Ihnen zur Verfügung?

□ 19. Wie sieht Ihr Büro/Ihre Arbeitsumgebung aus (Großraumbüro/Einzelbüros/etc.)?

□ 20. Welche Vorteile bietet der momentane Prozess für Sie als Entwickler (Arbeitsbeding., Freie Zeiteinteilung, Work-Life-Balance)?

- □ 20.1. Welche Vorteile bietet der momentane Prozess für Sie persönlich?

Notizen

_____

_____

_____

_____

_____

RQ4: Inwiefern decken sich die aus der Literatur extrahierten Prozessqualitätsattribute mit den Vorstellungen der Entwickler?

□ 21. Was würden Sie sich zusätzlich wünschen, dass der Prozess für Sie als Entwickler tun sollte?

- □ 22. Was für Arbeitsbedingungen wünschen Sie sich? (Work-Life-Balance, Home-Office, Wenig Stress, …)
- □ 23. Was sollte der Prozess in Bezug auf gute Zusammenarbeit für Sie ermöglichen?
- □ 24. Wie sollte der Prozess eine klare Kommunikation ermöglichen?
- □ 25. Welche Feedbackmechanismen würden Sie sich wünschen?
- □ 26. Wie sollte der Prozess die Weitergabe von Wissen im Team unterstützen?
- □ 27. Welche Freiheiten wünschen Sie sich in Ihrem Arbeitsalltag?
  - □ Auf Nachfrage: Können Sie z.B. selbst entscheiden, an welchen Aufgaben Sie arbeiten?

Notizen

_____

_____

_____

_____

_____

Schluss

□ Möchten Sie noch etwas auf der Aufnahme ergänzen oder haben Sie mit Fragen gerechnet, die nicht gestellt wurden, die aber interessant gewesen wären?

□ Für Teilnahme bedanken

□ Audioaufnahme beenden

□ Haben Sie noch interessante Punkte oder Fragen, welche Sie nicht auf der Aufnahme haben wollten?

□ (ggf.) Können Sie noch jemanden empfehlen, der Interesse an der Studie hätte?

□ Verlosung

□ Für Zeit bedanken

□ Wenn Interesse besteht, kann die Masterarbeit nach Abschluss zugesandt werden

□ Verabschiedung

Notizen

_____

_____

_____

_____

_____

Notizen

Anhang B

Informationsblatt zur Studie

Das Informationsblatt, das den Probanden im Vorfeld der Studie zugesandt worden ist, wird auf der nächsten Seite dargestellt. Dabei wird der Hintergrund, die Ziele und einige organisatorische Informationen zur Studie beschrieben. Lediglich die persönlichen Kontaktinformationen des Autors dieser Arbeit wurden im Hinblick auf die Abgabe und anschließende Veröffentlichung entfernt.

Interviewstudie zu entwicklerzentrierten Prozessqualitätsattributen

Hintergrund: Im Rahmen meiner Masterarbeit im Studiengang Informatik an der Leibniz Universität Hannover führe ich, Lukas Köhler, eine Interviewstudie zu entwickler-zentrierten Prozessqualitätsattributen durch. Dabei handelt es sich um Eigenschaften des Prozesses, welche spezifisch den Entwicklern dienen sollen, wie zum Beispiel die gezielte Unterstützung der Work-Life-Balance oder die Minimierung von stressbehafteten Phasen in der Entwicklung.

Ziel: Das Ziel dieser Studie ist es, herauszufinden, welche entwickler-zentrierten Prozessqualitätsattribute reale Softwareprojekte haben. Zusätzlich soll untersucht werden, welche Vorstellungen von entwickler-zentrierten Prozessqualitätsattributen Entwickler in Bezug auf ihren Entwicklungsprozess haben.

Langfristig ist es das Ziel, die entwickler-zentrierten Softwareprozessattribute bereits bei der Prozesskonstruktion zu berücksichtigen. Auf diese Weise soll ein für die Entwickler angenehmes Arbeitsumfeld geschaffen und qualitativ hochwertigere Software entwickelt werden.

Teilnehmer: Eine Teilnahme an der Studie ist möglich, wenn Sie an Softwareprojekten mitwirken und Ihre Hauptaufgabe in der Entwicklung liegt.

Dauer: ca. 45 Minuten.

Termin: Nach individueller Absprache.

Durchführung: Im Rahmen eines persönlichen Gesprächs vor Ort oder online über einen im Vorfeld bereitgestellten BigBlueButton[1]-Raum**.**

Kontakt: Lukas Köhler ([E-Mail-Adresse]; [Telefonnummer])

Ich würde mich sehr über eine Teilnahme von Ihnen freuen. Kontaktieren Sie mich hierzu gerne über meine Mailadresse oder einen kurzen Anruf. Sollten weiterführende Frage aufkommen, beantworte ich diese gerne auch auf diesem Weg.

Die Interviews werden aufgezeichnet und die Daten elektronisch zeitlich unbegrenzt gespeichert und zur Auswertung des Experiments herangezogen. Die aufgezeichneten Daten werden für die wissenschaftliche Forschung genutzt und ausschließlich pseudonymisiert ausgewertet. Die Daten können in wissenschaftlichen Publikationen in pseudonymisierter Form veröffentlicht werden.

[1] https://bigbluebutton.org/

Anhang C

Einverständniserklärung

Die Einverständniserklärung, die die Teilnehmer im Voraus des Interviews unterschreiben mussten, ist auf der folgenden Seite dargestellt. Ohne die ausgefüllte Erklärung konnten die Probanden nicht am Interview teilnehmen. Es wird zu Beginn noch einmal kurz der Hintergrund der Studie dargestellt. Außerdem wird beschrieben, wie die Daten gespeichert, verarbeitet und veröffentlicht werden können. Dabei wird ausdrücklich auf die Pseudonymisierung der erfassten Informationen hingewiesen und erläutert, dass jederzeit einer Nutzung der Daten widersprochen werden kann.

Einverständniserklärung zur Teilnahme an der Interviewstudie zu entwicklerzentrierten Softwareprozessattributen in Theorie und Praxis

Diese Interviewstudie wird von mir, Lukas Köhler, im Rahmen meiner Masterarbeit mit dem Titel „Vergleich von entwickler-zentrierten Softwareprozessanforderungen in Theorie und Praxis“ im Studiengang Informatik an der Leibniz Universität Hannover durchgeführt. Betreut wird die Arbeit von Dr. Jil Klünder, Fachgebiet Software Engineering, Leibniz Universität Hannover.

Im Rahmen dieser Interviewstudie untersuche ich, welche entwickler-zentrierten Prozessqualitätseigenschaften in realen Softwareprojekten vorkommen. Das heißt, ich untersuche, was Softwareprozesse für die Entwicklerinnen und Entwickler leisten und wo es Verbesserungspotenzial für die Entwicklerinnen und Entwickler gibt.

Mit Ihrer Erlaubnis wird das Interview aufgezeichnet und im Anschluss, um eine Analyse zu ermöglichen, transkribiert. Die im Rahmen des Gesprächs erhobenen Daten werden pseudonymisiert. Es wird kein Rückschluss auf Ihre Person möglich sein. Die Ergebnisse werden im Rahmen dieser Masterarbeit ausgewertet und veröffentlicht. Eine Veröffentlichung der Transkripte und der Audioaufnahmen findet dabei nicht statt. Darüber hinaus besteht die Möglichkeit, dass die Befunde in anderen wissenschaftlichen Arbeiten pseudonymisiert veröffentlicht werden. Dabei können Aussagen wörtlich wiedergegeben werden, die keine Rückschlüsse auf Ihre Person oder Ihr Unternehmen ermöglichen.

Die erfassten Daten werden nach Abschluss der Masterarbeit in pseudonymisierter Form elektronisch zeitlich unbegrenzt auf institutseigenen Servern gespeichert. Ein Zugriff durch externe Personen ist zu keinem Zeitpunkt möglich.

Gemäß dem Datenschutz gegenüber dem Informationsträger haben Sie das Recht auf Auskunft sowie Löschung Ihrer personenbezogenen Daten. Diese Einverständniserklärung kann jederzeit widerrufen werden. Nach einem erfolgten Widerruf werden Ihre personenbezogenen Daten gelöscht und für keine weiteren Publikationen mehr verwendet.

Mit den oben aufgeführten Punkten bin ich

□ einverstanden. □ nicht einverstanden.

_____

Nachname, Vorname

_____

Ort, Datum, Unterschrift

Anhang D

Entwicklerzentrierte Prozessattribute

In Abbildung D.1 ist die vollständige Tabelle der aus der Literatur erfassten EZSPA dargestellt.

Weiterbildung und Lernen	Teambezogen	Prozessaspekte	Persönliches
Weiterentwicklung	Zusammenarbeit (Sozial-fähiges Team, Harmonie, Ausgeglichenheit)	Entscheidungsfreiheit	Persönliche Erfüllung
Feedback	Transparenz (über die Erwartungen anderer)	Eigene Zieldefinition	Gute Arbeitsbedingungen (Arbeitsschutz, Arbeitsschutz (40-Stunden-Woche, Gesundheit, Wohlbefinden, Work-Life-Balance, Nachhaltigkeit, Verständnis, Wenig Stress), Sicherheit, Fokus, Störungsfreies Arbeiten)
Weiterbildung (Erfahrungstransfer)	Gegenseitige Unterstützung (zur Weiterentwicklung)	Selbstorganisiertes Team	
Kontinuierliche Verbesserung	Klare Kommunikation	wenig Abhängigkeiten zwischen Teams	
Wissenstransfer	Interesse an den Anderen	Entscheidungsbefugnis (andere Entscheidungen als bei Scrum)	
Mentoring	Respekt (Individualität, Wertesystem)	Optimaler Informationsfluss	
Fortbildung	Wertschätzung	Gute Arbeitsbedingungen (Keine geteilten Aufgaben, Risiken befürworten, Unterstützung durch Vorgesetzte)	
Unternehmensweites Lernen	Crossfunktionale Teams		
Kontinuierliches Lernen (Wissensaustausch)	Umgang		
	Fehlerkultur		
	Unterstützung (durch Scrum Master)		
	Kontinuität in den Mitarbeitern (Aufstiegsmöglichkeiten)		
	Gute Arbeitsbedingungen (Teamkonstellation bleibt stabil, Vertrauen)		

Abbildung D.1: Die aus der Literatur erfassten EZSPA

In der Tabelle D.1 sind die in der Literatur gefundenen Codes zusammen mit der Quelle und der Anzahl der Befunde dargestellt.

Code	XP [3]	Scrum [50]	DevOps [24]
Weiterentwicklung	2	0	0
Zusammenarbeit	13	2	4
Transparenz	1	0	0
Feedback	2	1	2
Entscheidungsfreiheit	1	4	0
Eigene Zieldefinition	1	1	0
Weiterbildung	2	1	0
Kontinuierliche Verbesserung	3	1	0
Persönliche Erfüllung	1	0	0
Gegenseitige Unterstützung	3	1	2
Klare Kommunikation	3	0	0
Wissenstransfer	3	0	0
Interesse an den Anderen	1	0	0
Respekt	4	2	0
Wertschätzung	1	0	0
Gute Arbeitsbedingungen	12	8	18
Crossfunktionale Teams	1	0	0
Umgang	1	0	0
Fehlerkultur	1	0	3
Selbstorganisiertes Team	0	3	0
Unterstützung	0	2	0
Mentoring	0	1	0
Fortbildung	0	1	0
Wenig Abhängigkeiten zw. Teams	0	0	4
Unternehmensweites Lernen	0	0	4
Kontinuierliches Lernen	0	0	11
Entscheidungsbefugnis	0	0	1
Optimaler Informationsfluss	0	0	1
Kontinuität in den Mitarbeitern	0	0	1

Tabelle D.1: Die Anzahl der EZSPA-Codes in der jeweiligen Literatur

Anhang E

Inhalt der beiliegenden CD

Auf der beiliegenden CD findet sich folgender Inhalt:

- Dieses Dokument („MA-Koehler2024.pdf“).
- Die LaTeX-Quelldateien dieses Dokuments („MA-Koehler2024.zip“).
- Ein Ordner namens „Studie“. Darin lässt sich folgender Inhalt finden:
  - Ein Ordner namens „Ergebnisse“. Dieser beinhaltet die *MaxQDA*-Projektdatei „Masterarbeit.mx24“, in welcher die Auswertung der Interviews stattgefunden hat, sowie die *Excel*-Datei „EZSPA_Literatur.xlsx“, die die Ergebnisse der Literaturstudie beinhaltet.
  - Ein Ordner namens „Transkripte“. Dieser beinhaltet die Transkripte der durchgeführten Interviews.
  - Ein Ordner namens „Vorlagen“. Dieser beinhaltet das für die Interviewstudie genutzte Informationsblatt, sowie die Einverständniserklärung und den Interviewleitfaden. Alle Dateien finden sich hier sowohl als *.docx*-Datei, als auch als *.pdf*-Datei.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Bezeichnung	**Beschreibung**
BBB	BigBlueButton 22, 30
EZSPA	entwicklerzentrierte Softwareprozessattribute 2, 17–20, 22, 24, 26, 31, 33, 67–69, 71–73, 75–77, 79–81, 97, 98, 101, 103
KPI	Key Performance Indicator 81
RQ	Research Question 18, 21, 24, 73
XP	Extreme Programming 7, 8, 101

Literaturverzeichnis

[1] O. A. Adeoye-Olatunde and N. L. Olenik. Research and scholarly methods: Semi-structured interviews. *JACCP: Journal of the American College of Clinical Pharmacy*, 4(10):1358–1367, 2021.

[2] A. Aitken and V. Ilango. A comparative analysis of traditional software engineering and agile software development. In *46th Hawaii International Conference on System Sciences*, pages 4751–4760, 2013.

[3] K. Beck and C. Andres. *Extreme Programming Explained: Embrace Change*. XP Series. Addison-Wesley, 2004.

[4] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for agile software development. `https://agilemanifesto.org/`, 2001. Zugriff: 10.03.2024.

[5] B. Boehm. A spiral model of software development and enhancement. *ACM SIGSOFT Software engineering notes*, 11(4):14–24, 1986.

[6] A. Boland, R. Dickson, and G. Cherry. *Doing a Systematic Review: A student's guide*. SAGE Publications Ltd, 2017.

[7] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006.

[8] S. Brinkkemper. Method engineering: Engineering of information systems development methods and tools. *Information & Software Technology*, 38:275–280, 1996. Method Engineering and Meta-Modelling.

[9] S. Brinkmann. Unstructured and Semi-Structured Interviewing. In *The Oxford Handbook of Qualitative Research*. Oxford University Press, 2014.

[10] M. Broy and M. Kuhrmann. *Projektorganisation und Management im Software Engineering*. Springer Vieweg Berlin, Heidelberg, 2013.

[11] C. M. Castellan. Quantitative and qualitative research: A view for clarity. *International Journal of Education*, 2, 2010.

[12] V. Clarke and V. Braun. Thematic analysis. *The Journal of Positive Psychology*, 12(3):297–298, 2017.

[13] D. Cohen, M. Lindvall, and P. Costa. An introduction to agile methods. *Advances in Computers*, 62(03):1–66, 2004.

[14] J. M. Corbin and A. Strauss. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 13(1):3–21, 1990.

[15] E. Diener, D. Wirtz, W. Tov, C. Kim-Prieto, D.-w. Choi, S. Oishi, and R. Biswas-Diener. New well-being measures: Short scales to assess flourishing and positive and negative feelings. *Social Indicators Research*, 97(2):143–156, 2010.

[16] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. Devops. *IEEE Software*, 33(3):94–100, 2016.

[17] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson. On the unhappiness of software developers. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, EASE '17, page 324–333, New York, NY, USA, 2017. Association for Computing Machinery.

[18] D. Graziotin, X. Wang, and P. Abrahamsson. Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ*, 2:e289, 2014.

[19] P. Hohl, J. Klünder, A. van Bennekum, R. Lockard, J. Gifford, J. Münch, M. Stupperich, and K. Schneider. Back to the future: origins and directions of the "agile manifesto" – views of the originators. *Journal of Software Engineering Research and Development*, 6(1):15, 2018.

[20] M. E. Hussein, S. Hirst, V. Salyers, and J. Osuji. Using grounded theory as a method of inquiry: Advantages and disadvantages. *The Qualitative Report*, 19(27):1–15, 2014.

[21] C. Iriarte and S. Bayona. It projects success factors: A literature review. *International Journal of Information Systems and Project Management*, 8(2):49–78, 2020.

[22] B. Johnson, T. Zimmermann, and C. Bird. The effect of work environments on productivity and satisfaction of software engineers. *IEEE Transactions on Software Engineering*, 47(4):736–757, 2021.

[23] S. H. Khandkar. Open coding. *University of Calgary*, 23(2009), 2009.

[24] G. Kim, J. Humble, P. Debois, J. Willis, and N. Forsgren. *Das DevOps-Handbuch: Teams, Tools und Infrastrukturen erfolgreich umgestalten.* O'Reilly, 2022.

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

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

[27] 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. Kuepper, G. Ruhe, E. Knauss, O. Oezcan-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.

[28] M. Kuhrmann, P. Tell, J. Klünder, R. Hebig, S. Licorish, and S. MacDonell. Helena stage 2 results. `https://helenastudy.wordpress.com/helena-results/publications/`, 2018. Zugriff: 10.03.2024.

[29] Y. Lindsjørn, D. I. Sjøberg, T. Dingsøyr, G. R. Bergersen, and T. Dybå. Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software*, 122:274–286, 2016.

[30] J. Manning. *In Vivo Coding.* John Wiley & Sons Inc., 2017.

[31] E. McLellan, K. M. MacQueen, and J. L. Neidig. Beyond the qualitative interview: Data preparation and transcription. *Field Methods*, 15(1):63–84, 2003.

[32] D. Méndez Fernández, W. Böhm, A. Vogelsang, J. Mund, M. Broy, M. Kuhrmann, and T. Weyer. Artefacts in software engineering: a fundamental positioning. *Software & Systems Modeling*, 18(5):2777–2786, 2019.

[33] S. Misoch. *Qualitative Interviews*. De Gruyter Oldenbourg, Berlin, München, Boston, 2015.

[34] N. B. Moe, T. Dingsøyr, and T. Dybå. Understanding self-organizing teams in agile software development. In *19th Australian Conference on Software Engineering*, pages 76–85, 2008.

[35] J. Münch, O. Armbrust, M. Kowalczyk, and M. Sotó. *Software Process Definition and Management*. Springer Berlin, Heidelberg, 2012.

[36] E. Murphy-Hill, C. Jaspan, C. Sadowski, D. Shepherd, M. Phillips, C. Winter, A. Knight, E. Smith, and M. Jorde. What predicts software developers' productivity? *IEEE Transactions on Software Engineering*, 47(3):582–594, 2021.

[37] National Academies of Sciences, Engineering, and Medicine. *Reproducibility and Replicability in Science*. The National Academies Press, Washington, DC, 2019.

[38] J. Noll, M. A. Razzak, J. M. Bass, and S. Beecham. A study of the scrum master's role. In M. Felderer, D. Méndez Fernández, B. Turhan, M. Kalinowski, F. Sarro, and D. Winkler, editors, *Product-Focused Software Process Improvement*, pages 307–323, Cham, 2017. Springer International Publishing.

[39] J. S. Oktay. *Grounded theory*. Pocket Guide to Social Work Research Methods. Oxford University Press Inc., 2012.

[40] Oxford English Dictionary. Stakeholder. `https://www.oed.com/dictionary/stakeholder_n?tab=meaning_and_use#12282312`, o. J. Zugriff: 10.03.2024.

[41] D. L. Paulhus, S. Vazire, et al. The self-report method. *Handbook of research methods in personality psychology*, 1:224–239, 2007.

[42] R. Pichler. *Scrum: Agiles Projektmanagement erfolgreich einsetzen*. dpunkt.verlag, Heidelberg, 2007.

[43] M. B. Powell, C. H. Hughes-Scholes, and S. J. Sharman. Skill in interviewing reduces confirmation bias. *Journal of Investigative Psychology and Offender Profiling*, 9(2):126–134, 2012.

[44] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever. Robust speech recognition via large-scale weak supervision. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 2023.

[45] K.-H. Renner and N.-C. Jacob. *Das Interview: Grundlagen und Anwendung in Psychologie und Sozialwissenschaften.* Basiswissen Psychologie. Springer Berlin, Heidelberg, 2020.

[46] J. Rowley. Conducting research interviews. *Management Research Review*, 35(3/4):260–271, 2012.

[47] W. W. Royce. Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, page 328–338, Washington, DC, USA, 1987. IEEE Computer Society Press.

[48] M. G. Salido O., G. Borrego, R. R. Palacio Cinco, and L.-F. Rodríguez. Agile software engineers' affective states, their performance and software quality: A systematic mapping review. *Journal of Systems and Software*, 204:111800, 2023.

[49] R. Sandsto and C. Reme-Ness. Agile practices and impacts on project success. *Journal of Engineering, Project, and Production Management*, 11(3):255–262, 2021.

[50] K. Schwaber and M. Beedle. *Agile Software Development with Scrum.* Agile Software Development. Prentice Hall, 2002.

[51] K. Schwaber and J. Sutherland. The Scrum Guide. `https://scrumguides.org/index.html`, 2011. Zugriff: 10.03.2024.

[52] A. Shrivastava, I. Jaggi, N. Katoch, D. Gupta, and S. Gupta. A systematic review on extreme programming. *Journal of Physics: Conference Series*, 2021.

[53] M.-A. Storey, T. Zimmermann, C. Bird, J. Czerwonka, B. Murphy, and E. Kalliamvakou. Towards a theory of software developer job satisfaction and perceived productivity. *IEEE Transactions on Software Engineering*, 47(10):2125–2142, 2021.

[54] J. Sutherland. Agile development: Lessons learned from the first scrum. *Cutter Agile Project Management Advisory Service: Executive Update*, 5(20):1–4, 2004.

[55] C. Tam, E. J. da Costa Moura, T. Oliveira, and J. Varajão. The factors influencing the success of on-going agile software development projects. *International Journal of Project Management*, 38(3):165–176, 2020.

[56] P. Tell, J. Klünder, S. Küpper, D. Raffo, S. MacDonell, J. Münch, D. Pfahl, O. Linssen, and M. Kuhrmann. Towards the statistical construction of hybrid development methods. *Journal of Software: Evolution and Process*, 33(1):e2315, 2021. e2315 smr.2315.

[57] 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, 2019.

[58] T. R. Tulili, A. Capiluppi, and A. Rastogi. Burnout in software engineering: A systematic mapping study. *Information and Software Technology*, 155:107116, 2023.

[59] L. R. Vijayasarathy and C. W. Butler. Choice of software development methodologies: Do organizational, project, and team characteristics matter? *IEEE Software*, 33(5):86–94, 2016.

[60] S. Wagner and M. Ruhe. A systematic review of productivity factors in software development. `https://arxiv.org/abs/1801.06475`, 2018.

[61] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Springer Berlin, Heidelberg, 2012.