Fachhochschule Graubünden
University of Applied Sciences

Churer Schriften
zur Informationswissenschaft

Herausgegeben von
Wolfgang Semar

Arbeitsbereich
Informationswissenschaft

Schrift 138

Ursachen für die geringe Verbreitung von Extreme Programming

Weshalb sich lediglich Praktiken der agilen Methode durchgesetzt haben

Mara Funaro

Chur 2021

Churer Schriften zur Informationswissenschaft

Herausgegeben von Wolfgang Semar

Schrift 138

Ursachen für die geringe Verbreitung von Extreme Programming

Weshalb sich lediglich Praktiken der agilen Methode durchgesetzt haben

Mara Funaro

Diese Publikation entstand im Rahmen einer Thesis zum Bachelor of Science FHGR in Digital Business Management.

Referent: Prof. Dr. phil. Alexandra Weissgerber

Korreferent: Anthea Moravánszky

Verlag: Fachhochschule Graubünden

ISSN: 1660-945X

Ort, Datum: Chur, November 2021

Abstract

Vielerorts wird heute im Unternehmenskontext Agilität angestrebt. Eine Agilisierung auf Softwareentwicklungsebene kann durch den Einsatz von Extreme Programming (XP) erfolgen. Dennoch wird XP als agile Methode heute kaum noch angewandt. In dieser Arbeit wird nach Erklärungen gesucht, weshalb die Methode nur noch gering verbreitet ist, obwohl die Praktiken daraus noch zur Anwendung kommen. Dafür werden Ursachen aus der Literatur ermittelt und durch Erkenntnisse aus fünf Interviews mit erfahrenen Agile Coaches ergänzt. Die erhobenen Daten werden mit einer qualitativen Inhaltsanalyse ausgewertet. Die Ursachen, welche daraus ermittelt werden konnten, weisen eine erhöhte Heterogenität auf und deuten auf ein komplexes Phänomen unterschiedlichster Aspekte hin. In Summe scheitert XP in der Praxis, weil das Fundament einer agilen Arbeitsweise nicht gegeben ist. Demnach resultieren am häufigsten Aspekte, die das Mindset, die Kultur und die Werte betreffen als unzureichend.

Inhaltsverzeichnis

1 Einleitung 1
1.1 Ein- und Abgrenzung 1
1.2 Relevanz des Themas 1
1.3 Forschungsfrage und Teilfragen 2
1.4 Ziel der Arbeit 2
1.5 Inhaltlicher Aufbau der Arbeit 3
2 Theoretischer Hintergrund 5
2.1 Methodische Vorgehensweise Teilfrage 1 5
2.2 Bestandteile einer agilen Vorgehensweise 5
2.2.1 Agiles Mindset 6
2.2.2 Agile Werte 6
2.2.3 Agile Prinzipien 7
2.2.4 Agile Praktiken 8
2.3 Extreme Programming als agile Softwareentwicklungsmethode 8
2.4 Aktueller Stand der Verbreitung und Anwendung von Extreme Programming 13
2.5 Häufigste XP-Praktiken im Einsatz 14
2.6 Zusammenfassung der Erkenntnisse und Beantwortung der Teilfrage 1 15
3 Methodisches Vorgehen 17
3.1 Literaturrecherche für Teilfrage 2 17
3.2 Teilstrukturierte Interviews für Teilfrage 3 18
3.3 Qualitative Inhaltsanalyse zur Auswertung für Teilfrage 2 und 3 18
4 Gründe aus der Literatur für die geringe Verbreitung von XP 21
4.1 Ergebnisse der Literaturrecherche 21
4.1.1 Ergebnisse einzelne Praktiken 21
4.1.2 Ergebnisse hybride Vorgehensmodelle 27
4.1.3 Ergebnisse XP-Methode 29
4.2 Zusammenfassung der Erkenntnisse und Beantwortung der Teilfrage 2 31
5 Gründe aus der Praxis für die geringe Verbreitung von XP 37
5.1 Konzeption des Interviewleitfadens 37
5.2 Auswahl der interviewten Personen 37
5.3 Durchführung der Interviews 39
5.4 Transkription und Auswertung 39
5.5 Ergebnisse der Leitfadeninterviews 40
5.5.1 Ergebnisse einzelne Praktiken 40

5.5.2 Ergebnisse hybride Vorgehensmodelle 45
5.5.3 Ergebnisse XP-Methode 49
5.5.4 Ergebnisse spezifisch für die Schweiz 53
5.6 Zusammenfassung der Erkenntnisse und Beantwortung der Teilfrage 3 54
6 Diskussion der Ergebnisse 59
7 Fazit 69
8 Kritische Reflexion 71
9 Literaturverzeichnis 73
10 Anhang 83
10.1 Anhang A: Datengrundlage zur Visualisierung der Methoden in Anwendung . 83
10.2 Anhang B: Berechnungsgrundlage Praktiken im Einsatz 84
10.3 Anhang C: Kodierleitfaden 87
10.4 Anhang D: Auswertung Literatur (Kategoriensystem) 91
10.5 Anhang E: Interviewleitfaden 111
10.6 Anhang F: Auswertung Interviews (Kategoriensystem) 113
10.7 Anhang G: Zusammenfassung aller Gründe und Zuweisung für Diskussion 138

Abbildungsverzeichnis

Abbildung 1: Agile Onion (Rowell, 2020) 6
Abbildung 2: Praktiken, Grundprinzipien und Werte von XP (Eigene Darstellung in Anlehnung an Beck, 2004) 9
Abbildung 3: Prozentuale Anwendung der Methoden Scrum, XP und Hybrid (Eigene Darstellung in Anlehnung an Meier & Kropp, 2013a, 2015, 2017; VersionOne Inc., 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019) 14
Abbildung 4: Durchschnittliche Anwendung der XP-Praktiken auf globaler Ebene (Eigene Darstellung in Anlehnung an VersionOne Inc., 2006; 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019) 15
Abbildung 5: Darstellung der Moore's Curve von Pelrine, 2020 (in Anlehnung an Moore, 1995; Pietri, 2011) 47
Abbildung 6: Hindernde Faktoren für die einzelnen Praktiken (in Summe) 59
Abbildung 7: Hindernde Faktoren für die Praktik Kunde vor Ort 60
Abbildung 8: Hindernde Faktoren für die Praktik 40-Stunden Woche 61
Abbildung 9: Hindernde Faktoren für die Praktik gemeinsame Verantwortlichkeit 61

Abbildung 10: Hindernde Faktoren für die Praktik Paarprogrammierung 61
Abbildung 11: Hindernde Faktoren für die Praktik Refactoring.....62
Abbildung 12: Hindernde Faktoren für die Praktik Testen 62
Abbildung 13: Hindernde Faktoren für die Praktik einfaches Design 63
Abbildung 14: Hindernde Faktoren für die Praktik Metapher.....63
Abbildung 15: Hindernde Faktoren für die XP-Methode 64
Abbildung 16: Hindernde Faktoren für hybride Vorgehensmodelle 65
Abbildung 17: Hindernde Faktoren für eine Anwendung von XP und XP-Bestandteilen in der Schweiz.....66
Abbildung 18: Spannungsfeld zwischen Management und Entwicklung.....70

Tabellenverzeichnis

Tabelle 1: Gründe aus der Literatur für die geringe Verbreitung einzelner XP-Praktiken 33
Tabelle 2: Gründe aus der Literatur für die geringe Verbreitung von XP 34
Tabelle 3: Auswahlkriterien und Angaben der einzelnen Interviewpersonen 39
Tabelle 4: Datengrundlage für die Abbildung 3 zuzüglich der Visualisierung der Daten für die Schweiz, welche im Text erwähnt wurden. (Eigene Darstellung in Anlehnung an Meier & Kropp, 2013a; 2015, 2017; VersionOne Inc., 2006; 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019) 83
Tabelle 5: Berechnungsgrundlage für die Abbildung 4 (Eigene Darstellung in Anlehnung an VersionOne Inc., 2006; 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019) 84
Tabelle 6: Praktiken im Einsatz in der Schweiz (Eigene Darstellung in Anlehnung an Meier & Kropp, 2013a; 2015; 2017) 85
Tabelle 7: Berechnungsgrundlage für die XP-Praktik Planungsspiel in der Schweiz (Eigene Darstellung in Anlehnung an Meier & Kropp, 2013a; 2015; 2017).....86

Abkürzungsverzeichnis

FDA	Food and Drug Administration
FF	Forschungsfrage
FH	Fachhochschule
IT	InformationTechnology
N/A	Not Available (dt. nicht verfügbar)
TDD	Test Driven Development
TF	Teilfrage
TÜV	Technischer Überwachungsverein
XP	Extreme Programming

1 Einleitung

Die Anforderungen an Software sind volatil und es bedarf einer agilen Arbeitsweise, um auf Bedürfnisse der Kundschaft zeitnah zu reagieren. Agile Vorgehensmethoden stehen daher, wie auch der Trends Report der SwissQ Consulting AG (2021, S. 9) aufzeigt, ‹hoch im Kurs›. Rund 75.1% der im Report Befragten wenden im Unternehmen Scrum an. Praktiken des Extreme Programmings (XP), einer agilen Softwareentwicklungsmethode, die von Kent Beck vorgestellt wurde, haben sich in der Praxis ebenfalls etabliert. Die grundlegende agile Methodologie von XP und Scrum ist dieselbe. Dennoch gilt Scrum als erfolgreich etabliert und XP als Methode, gemäss mehreren Reports (Meier & Kropp, 2017; SwissQ Consulting AG, 2020; VersionOne Inc., 2019), kaum angewandt. Wie der Trends Report der SwissQ Consulting AG (2020) zeigt, ist die Methode in der Schweiz mit 1.8% wenig im Einsatz. Der neuste Report der SwissQ Consulting AG (2021) führt XP gar nicht mehr auf.

Die Gründe für den vermeintlich geringen Einsatz in der Praxis können unterschiedlichen Ursprungs sein. Im Rahmen der Thesis werden diese aus der Literatur systematisch erarbeitet und mit Erkenntnissen aus der Praxis ergänzt. Durch die Kombination von Literatur und Praxis soll eine Übersicht entstehen, welche Gründe für eine geringe Verbreitung der Methodik aufzeigt.

1.1 Ein- und Abgrenzung

Die Thesis beschäftigt sich mit den von Kent Beck (2005) eingeführten Praktiken, welche ein wesentlicher Bestandteil von XP sind. Auf die Werte und Prinzipien von Extreme Programming wird in der Arbeit nicht der Fokus gelegt. Was aber nicht bedeutet, dass sie weniger relevant sind. Werte und Prinzipien sind in der zur Verfügung stehenden Zeit ungenügend messbar und deren Datenerhebung würde den Rahmen der vorliegenden Arbeit sprengen. Neben Extreme Programming werden keine weiteren agilen Softwareentwicklungsmethoden näher erläutert. Des Weiteren grenzt sich die Arbeit bei der qualitativen Datenerhebung mittels Interviews geografisch auf die Schweiz ein.

1.2 Relevanz des Themas

Die Relevanz des Themas begründet sich durch die Aktualität und Verbreitung von Agilität. Wie die dritte Swiss Agile Studie zeigt, wenden 85% der befragten IT-Unternehmen in der Schweiz agile Methoden in unterschiedlicher Ausprägung an (Meier & Kropp, 2017). Die Studien von Meier und Kropp (2013a, 2015, 2017) zeigen, dass

Scrum als häufigste Methode im Einsatz ist. Gemäss dem Scrum Guide ist Scrum ein Rahmenwerk, in welchem andere agile Methodiken zur Produktentwicklung eingesetzt werden können (Schwaber & Sutherland, 2020). Scrum selbst gibt keine Praktiken vor, wie die Produktentwicklung erfolgen kann. Diese Tatsache verstärkt den Klärungsbedarf, da dieselben Studien zeigen, dass Praktiken von XP deutlich in der Praxis angewendet werden, was im Widerspruch steht mit der geringen Anwendung der Methodik.

1.3 Forschungsfrage und Teilfragen

Im Rahmen der Thesis soll die folgende Forschungsfrage beantwortet werden:

FF: Weshalb findet XP als agile Entwicklungsmethode in der Schweiz nur noch geringe Anwendung, obwohl die Praktiken in der Praxis etabliert sind?

Um diese Frage vollumfänglich beantworten zu können, wurden die nachfolgenden Teilfragen ausgearbeitet:

TF1: Was ist Extreme Programming, welche Praktiken gibt es und wie häufig werden sie in der Praxis angewandt?

TF2: Welche Gründe finden sich in der Literatur für die geringe Anwendung der Methodik?

TF3: Welche Gründe werden durch Personen aus der Praxis für die geringe Anwendung der Methodik genannt?

Die Teilfrage 1 dient dem gemeinsamen Verständnis der Lesenden und der Ausarbeitung der Praktiken, welche am häufigsten in der Praxis zur Anwendung kommen.

Die Teilfrage 2 wird mittels einer Literaturrecherche und -analyse beantwortet. Es werden aus relevanter Literatur Gründe eruiert, die eine geringe Anwendung der Methodik sowie einzelner Praktiken erklären können. Die Literaturrecherche wird geografisch nicht eingeschränkt, da der Sachverhalt global ist.

Die Teilfrage 3 dient zum Abgleich der Befunde aus der Literatur und zur Erweiterung der Erkenntnisse durch eine qualitative Datenerhebung und -auswertung. Die Grundlage zur Beantwortung dieser Teilfrage bilden die Befragungen von Personen mit Praxiserfahrung vorwiegend in der Schweiz.

1.4 Ziel der Arbeit

Im Rahmen der Thesis wird nach Erklärungen gesucht, weshalb die Methode nur noch gering verbreitet ist, obwohl die Praktiken daraus noch zur Anwendung kommen. Durch

die Kombination der Literatur mit Einblicken in die Praxis soll der Forschungsgegenstand in seiner Breite analysiert werden. Die Arbeit hat einen explorativen Charakter und verfolgt das Ziel, den genannten Sachverhalt zu erkunden und zu beschreiben. Aufgrund der Erkenntnisse aus der offenen Forschungsfrage können im Anschluss Hypothesen und Theorien entwickelt werden (Döring & Bortz, 2016, S. 192), die Aussagen darüber machen, wie die Softwareentwicklung in einem agilen Umfeld optimiert werden kann. Dabei liegt der Schwerpunkt auf XP als Methode und auf den einzelnen Praktiken. Die Offenlegung der ermittelten Aspekte soll Optimierungspotenziale aufdecken und zur Sensibilisierung einer agilen Arbeitsweise im Umfeld der Softwareentwicklung führen. Die Arbeit richtet sich an Personen, die sich mit der Planung und Entscheidung zu einer agilen Softwareentwicklung auseinandersetzen. In der Arbeit werden Aspekte herausgearbeitet, die den Einsatz von XP erschweren. Unter Berücksichtigung dieser Aspekte kann die Agilität im Kontext der Softwareentwicklung verbessert werden. Den Personen, die bereits agil Software entwickeln und XP als Ganzes oder Praktiken daraus einsetzen, bietet die Arbeit einen Einblick in Herausforderungen, die bei der Anwendung von XP auftreten können.

In dieser Arbeit werden weder Handlungsempfehlungen abgegeben, noch wird nach Möglichkeiten gesucht, die den Bekanntheitsgrad der Methode steigern könnten.

1.5 Inhaltlicher Aufbau der Arbeit

Dieses Kapitel gibt den Lesenden eine Übersicht über den inhaltlichen Aufbau der Arbeit. Der Einleitung folgend wird in Kapitel 2 auf den theoretischen Hintergrund der Thematik eingegangen. Darin werden die Bestandteile einer agilen Arbeitsweise erläutert. Dem folgend werden Extreme Programming sowie die zwölf Praktiken beschrieben. Im Anschluss wird eine Grafik erstellt, welche die häufigsten Praktiken im Einsatz visualisiert. Das Kapitel 2 schliesst mit einer Zusammenfassung und der Beantwortung der Teilfrage 1 im Unterkapitel 2.6 ab.

Im Kapitel 3 wird das methodische Vorgehen für die Teilfragen 2 und 3 im Detail erläutert. Dabei werden in den Unterkapiteln die jeweilige Datenerhebung je Teilfrage sowie die Auswertungsmethode präsentiert.

Zur Beantwortung der Teilfrage 2 dient das Kapitel 4. Darin werden die Gründe für eine geringe Verbreitung der Methode und einzelner Praktiken, welche aus der Literatur ermittelt wurden, systematisch aufgearbeitet und inhaltlich verdichtet. Das Kapitel schliesst mit der Beantwortung der Teilfrage im Unterkapitel 4.2 ab.

Im Kapitel 5 werden die Konzeption des Leitfadens sowie die Auswahl der interviewten Personen thematisiert. Darauf folgt eine Erläuterung zur Durchführung der Interviews und der anschliessenden Auswertung. Die Ergebnisse der Untersuchung werden im Unterkapitel 5.5 detailliert präsentiert. Die Teilfrage 3 wird im Unterkapitel 5.6 beantwortet.

Die Ergebnisse aus der Literatur und den Interviews werden im Kapitel 6 diskutiert, indem eine Synthese der Ergebnisse stattfindet und eine Interpretation als Diskussionsgrundlage gemacht wird. Im Fazit in Kapitel 7 wird die Forschungsfrage beantwortet. Zu guter Letzt wird im Kapitel 8 die Arbeit kritisch reflektiert und Limitationen thematisiert.

Alle ausführlichen, relevanten Tabellen sowie die Transkripte können dem Anhang entnommen werden.

2 Theoretischer Hintergrund

In diesem Kapitel werden die Bestandteile einer agilen Kultur und Vorgehensweise hergeleitet und Extreme Programming als agile Softwareentwicklungsmethode näher erläutert. Anschliessend wird der aktuelle Stand der Verbreitung und Anwendung von Extreme Programming sowohl in der Schweiz als auch global betrachtet und visualisiert. Es werden die zwölf XP-Praktiken genannt und beschrieben. Das Kapitel schliesst mit einer Zusammenfassung und der Beantwortung der Teilfrage 1 ab.

2.1 Methodische Vorgehensweise Teilfrage 1

Zur Beantwortung der Teilfrage 1 wird primär die Literatur von Beck und Andres (2005) verwendet, da Kent Beck als Erfinder der XP-Methode gilt. Zur Aufarbeitung des aktuellen Standes auf nationaler Ebene werden die Studien von Andreas Meier und Martin Kropp berücksichtigt. Die beiden sind Dozierende an Schweizer Fachhochschulen und haben sich im Rahmen des Swiss Agile Research Network verstärkt mit der Agilität in der Schweiz auseinandergesetzt (Swiss Agile Research Network [SARN], o.J.). Des Weiteren werden Trend Reports zu agilen Vorgehensweisen in Unternehmen berücksichtigt, wie die Publikationen von SwissQ Consulting AG und die State of Agile Reports, die jährlich von Digital.ai herausgegeben werden. Die Graue Literatur bietet sowohl global als auch national einen Einblick in die Praxis und ermöglicht eine Aussage über den aktuellen Stand in Unternehmen.

2.2 Bestandteile einer agilen Vorgehensweise

Durch das agile Manifest, welches 2001 aus einem Treffen von führenden Softwareentwicklern entstanden ist (Andersson & Sandahl, 2021; Highsmith, 2001), wurden Faktoren für ein erfolgreiches Softwareentwicklungsprojekt definiert (Andersson & Sandahl, 2021, S. 3). Das Manifest hält die wichtigsten Aspekte einer agilen Entwicklung fest. Es besteht aus vier Werten und zwölf Prinzipien, welche in Kapitel 2.2.2 und 2.2.3 kurz erläutert werden.

Die Verkapselung agiler Aspekte wird in Abbildung 1 ersichtlich. Während Prinzipien, Praktiken, Tools und Prozesse einer agilen Vorgehensweise dienen, bilden das Mindset, die Kultur und die Werte deren Fundament. Ein vollständiges Ausschöpfen der jeweiligen agilen Methode ist erst möglich, wenn der Kern verinnerlicht wurde (Sauter et al., 2018, S. 23).

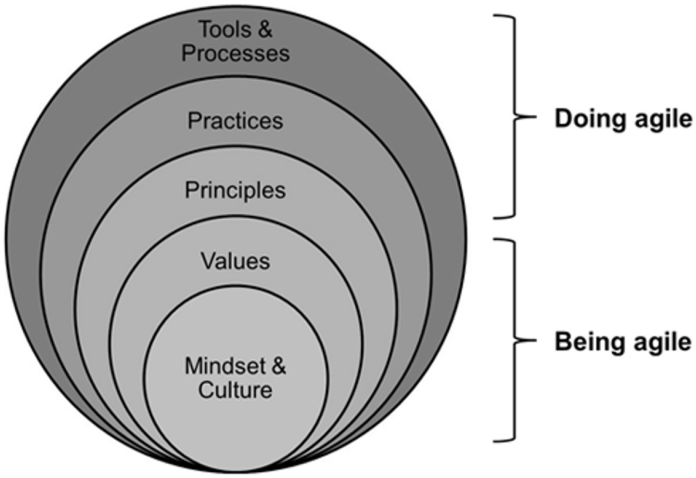

Abbildung 1: Agile Onion (Rowell, 2020)

2.2.1 Agiles Mindset

Das Mindset bildet den Kern einer agilen Arbeitsweise und beginnt, wie Hanschke (2017) beschreibt, im Kopf; «Agilität geht einher mit *‹Agile Thinking›.* Es ist eine Grundhaltung, die verinnerlicht werden muss» (S. 2). Das Mindset ist demnach eine Denkweise und «wird durch Werte und daraus abgeleitete Prinzipien beschrieben, die das Verhalten eines Menschen beeinflussen» (Freyth, 2019, S. 175). Praktische agile Tätigkeiten sind zwar stärker sichtbar, jedoch entsprechend reduziert wirksam, wenn tieferliegende Faktoren wie das Mindset unzureichend gefestigt oder inexistent sind. Das agile Mindset geht davon aus, dass sich Individuen mit den ständigen Veränderungen auseinandersetzen müssen und durch eine positive Haltung diese Veränderungen leichter erreicht werden können als mit Kontrolle und unter Druck (Dams, 2019a, S. 5). Die Fähigkeit, sich kontinuierlich einer unsicheren Zukunft anpassen zu können, ist somit Bestandteil des agilen Mindsets (Sauter et al., 2018, S. 244).

2.2.2 Agile Werte

Die agilen Werte basieren auf dem Mindset und bilden zugleich das Fundament für die Prinzipien (Hofert, 2018, S. 10). Im agilen Manifest wurden die folgenden Wertpaare in vier Leitsätzen mit Fokus auf die Softwareentwicklung (Dams, 2019b, S. 9) festgehalten:

1. «**Individuen und Interaktionen** mehr als Prozesse und Werkzeuge
2. **Funktionierende Software** mehr als umfassende Dokumentation
3. **Zusammenarbeit mit dem Kunden** mehr als Vertragsverhandlung
4. **Reagieren auf Veränderung** mehr als Befolgen eines Plans» (Beck, Beedle et al., 2001a)

Bei den Leitsätzen handelt es sich gemäss Hanser (2010) um eine Art Randbedingungen für eine agile Softwareentwicklung (S. 10). Wie Beck, Beedle et al. (2001a) erwähnen, sind die Werte auf der rechten Seite zwar wichtig, die Werte auf der linken Seite, hier in fetter Schrift hervorgehoben, werden jedoch als relevanter eingeschätzt. Die vier Leitsätze gelten als zentrale Werte, diese können nach Bedarf entsprechend konkretisiert und erweitert werden (Böhm, 2019, S. 17).

2.2.3 Agile Prinzipien

Nebst den vier Grundwerten beinhaltet das agile Manifest auch zwölf Prinzipien (Beck, Beedle et al., 2001b). Agile Werte werden durch diese Prinzipien handlungsleitend (Böhm, 2019, S. 19) und somit in der täglichen Praxis anwendbar (Hanschke, 2017, S. 8).

Die zwölf Prinzipien sind ausführlich im agilen Manifest[1] festgehalten. In diesem Kapitel wird lediglich der Grundgedanke für ein allgemeines Verständnis zusammengefasst:

Die Software soll in regelmässigen und kurzen Abständen dem Auftraggebenden präsentiert und Rückmeldung eingeholt werden. Das Feedback während der Entwicklungsphase ermöglicht es, Missverständnisse zu beseitigen, neue Erkenntnisse zu berücksichtigen und Fehlentwicklungen frühzeitig offenzulegen. Insofern sind Änderungen der Anforderungen willkommen und werden als wertvollen Input wahrgenommen. Agile Teams arbeiten interdisziplinär. Das bedeutet, dass alle notwendigen Funktionen, die für die Produkterstellung erforderlich sind, Teil des Projektteams sind. Damit effizient gearbeitet werden kann, spricht sich das Team täglich ab. Hindernisse, Doppelspurigkeiten oder Unterstützungsbedarf werden so offengelegt und es kann entsprechend reagiert werden. Eine Produktvision hilft dem Team, das grosse Ganze nicht aus den Augen zu verlieren. Es muss ein Umfeld geschaffen werden, in dem sich interdisziplinäre Teams autonom entfalten können. Eine permanente Kontrolle durch das Management wird dabei als kontraproduktiv wahrgenommen. Für den Erfolg massgebend ist der Fortschritt der Projektarbeit. Bei der Erreichung der Projektziele soll jedoch darauf geachtet werden, dass sich das Team nicht verausgabt, vielmehr soll eine Kontinuität aufgebaut werden, die produktives Arbeiten fördert. Des Weiteren wird die direkte Kommunikation von Angesicht zu Angesicht bei komplexeren Sachverhalten als effizienteste Austauschform betrachtet. Eine erfolgreiche Umsetzung der Prinzipien erfordert hohe fachliche, soziale und organisatorische Kompetenzen der einzelnen Teammitglieder. Durch die Arbeit der Teammitglieder soll primär Nutzen gestiftet werden,

[1] https://agilemanifesto.org/iso/de/principles.html

dies bedingt, dass alles, was überflüssig und nicht zwingend notwendig ist, auch nicht berücksichtigt werden sollte. Die Dinge sollten so einfach wie möglich gehandhabt werden. Auf fachlicher, sozialer und organisatorischer Ebene soll eine kontinuierliche Verbesserung stets zu optimalen Lösungen führen (Beck, Beedle et al., 2001b; Michl, 2018, S. 6–12).

2.2.4 Agile Praktiken

In der Literatur werden Begriffe wie agile Praktiken und agile Techniken teilweise unterschiedlich verwendet oder analog genutzt (Preußig, 2020, S. 59). Der Begriff agile Techniken wird in dieser Arbeit nicht verwendet. Unter agilen Praktiken werden «anwendbare Werkzeuge, Methoden und Handlungsweisen (…), welche die praktische Art des Arbeitens erleichtern» (Böhm, 2019, S. 22) verstanden. Wie Hanschke (2017) schreibt, haben viele dieser agilen Praktiken ihren Ursprung beim Extreme Programming (S. 30). Da für diese Arbeit die Praktiken von Extreme Programming im Fokus stehen, werden diese im folgenden Kapitel näher erläutert.

2.3 Extreme Programming als agile Softwareentwicklungsmethode

Extreme Programming ist eine agile Softwareentwicklungsmethode, die in den 1990er Jahren von Kent Beck eingeführt wurde (Brandt-Pook & Kollmeier, 2020, S. 27). XP verfolgt das Ziel, die Softwarequalität zu verbessern und rasch auf ändernde Anforderungen zu reagieren (Alam & Gühl, 2020, S. 138; Beck, 2004). Die notwendige Flexibilität erlangt XP dadurch, dass sich die Methode von unnötigem Ballast, welcher für die Programmierung hindernd ist, distanziert und sich ausschliesslich auf die Entwicklung konzentriert. Die Befreiung solchen Ballasts macht XP zu einer sogenannten leichtgewichtigen Methode (Beck, 2004, S. 17; Beck & Andres, 2005, S. 3; Brandt-Pook & Kollmeier, 2020, S. 27–28; Highsmith, 2001). Wie Freedman (2016) schreibt, eignet sich XP für die Softwareentwicklung und ist nicht als Projektmethode anwendbar (S. 214). Durch XP wird Agilität ins Extreme getrieben (Halstenberg et al., 2020, S. 17). Programmierpraktiken werden in ihrer Extreme angewendet (Brandt-Pook & Kollmeier, 2020, S. 29), was auch den Namen der Methodik erklärt.

Beck und Andres (2005) nennen die Werte, Prinzipien und Praktiken als feste Bestandteile der Methode. Als Ausgangslage agiler Entwicklung gelten die Praktiken, welche den Anwendenden einen Anhaltspunkt geben und die Programmierung effektiver gestalten. Als Praktiken gelten die Dinge, welche auf routinierter Basis täglich ausgeübt werden, sie sind klar und objektiv definiert (S. 13–34).

Die Werte verleihen den Praktiken einen Sinn. Sie sorgen dafür, dass die ausgeführten Praktiken eine klare Richtung und einen Zweck haben. Die Abbildung 2 visualisiert die Bestandteile von XP. Die Werte werden durch Prinzipien mit den Praktiken verbunden. Prinzipien werden als domänenspezifische Richtlinien definiert und sollen den Anwendenden eine bessere Vorstellung darüber vermitteln, was die Praktiken erreichen sollen. Während die Werte allgemeingültig sind, gelten die Praktiken für das Tagesgeschäft als opportun (Beck & Andres, 2005).

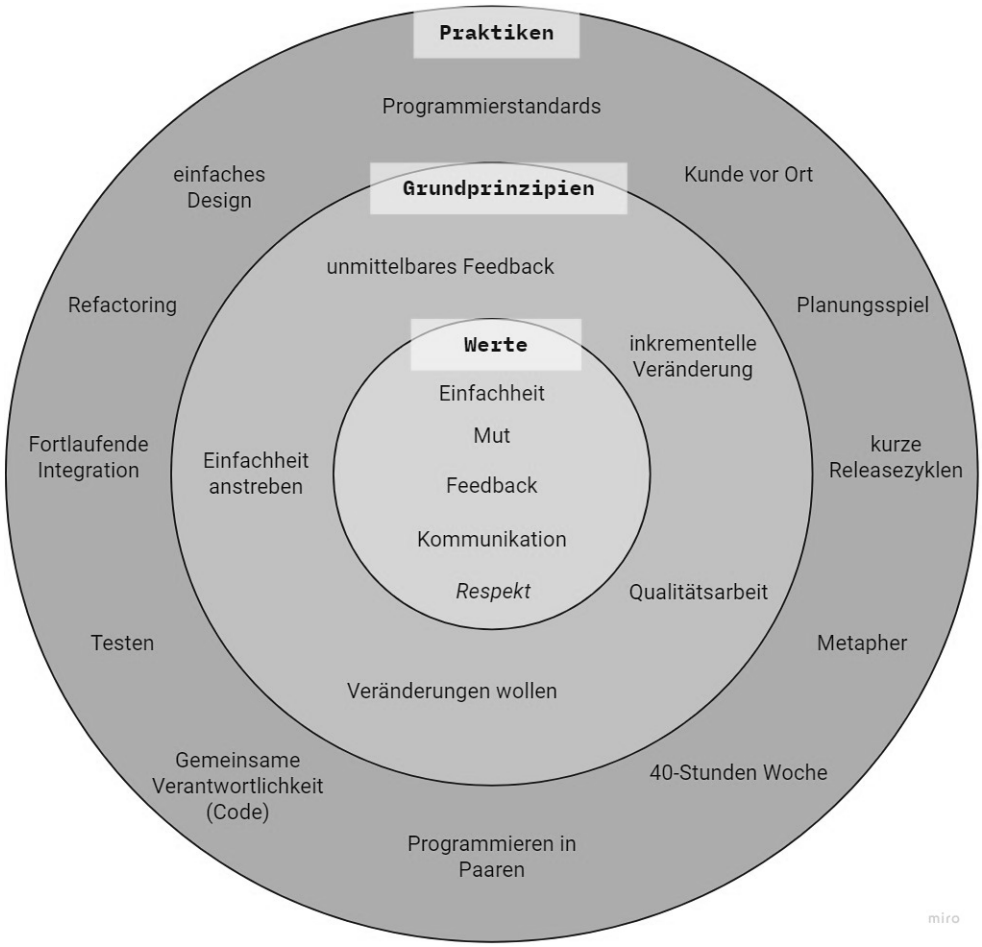

Abbildung 2: Praktiken, Grundprinzipien und Werte von XP (Eigene Darstellung in Anlehnung an Beck, 2004)

Folgend werden die zwölf Praktiken von Extreme Programming näher erläutert:

1 – Planungsspiel (*Planning Game*)

Am Anfang jedes Entwicklungszyklus findet eine Planungssitzung statt, in welcher das Entwicklungsteam mit der Kundin oder dem Kunden die gewünschten Anforderungen bespricht. Diese Anforderungen werden in Form von Stories definiert, verfeinert und geschätzt. Bei Stories handelt es sich um kurze Geschichten, welche die umzusetzende Anforderung beschreiben. Am Ende des Planungsspiels steht fest, welche Stories im Rahmen der nächsten Iteration realisiert werden (Alam & Gühl, 2020, S. 139; Freedman, 2016, S. 214; Hanschke, 2017, S. 30). Die Stories werden aus der Perspektive der Kundin oder des Kunden verfasst (Beck & Andres, 2005, S. 44–45), weshalb sie oft auch als User Stories bekannt sind.

2 – Kurze Releasezyklen (*Small Releases*)

Es wird das Ziel verfolgt, in kurzen Releasezyklen der Kundin oder dem Kunden möglichst früh funktionierende Software bereitzustellen. Dadurch können Aspekte wie unerwünschtes Verhalten oder Änderungswünsche schneller eruiert und in einer künftigen Auslieferung berücksichtigt werden. Die Software wird Komponente für Komponente zu einem System entwickelt. Dabei ist es wichtig, dass die ausgelieferte Version immer als Ganzes sinnvoll ist und getestet wird (Alam & Gühl, 2020, S. 139; Beck, 2004, S. 56; Brandt-Pook & Kollmeier, 2020, S. 29).

3 – Metapher (*Metaphor*)

Anstelle einer komplexen Systemarchitektur wird in XP eine Metapher verwendet. Diese soll dazu dienen, den Beteiligten die Gesamtidee und Funktionsweise des Systems mittels minimalem Einsatz von Fachjargon zu vermitteln und so eine Vision zu schaffen, die sowohl für die Entwickelnden als auch für die Kundin oder den Kunden verständlich ist (Alam & Gühl, 2020, S. 139; Beck, 2004, S. 56–57; Freedman, 2016, S. 215).

4 – Einfaches Design (*Simplicity*)

Einfachheit ist ein zentraler Aspekt für XP (Freedman, 2016, S. 215). Wie Alam und Gühl (2020) schreiben, «soll nur implementiert werden, was tatsächlich für die Umsetzung der Anforderungen benötigt wird. Unnötige Komplexität ebenso wie Duplikationen sind zu

vermeiden. Nicht mehr verwendeter Code soll gelöscht werden» (S. 139). Das, was gebraucht wird, soll erst dann implementiert werden, wenn es notwendig ist (Beck, 2004, S. 57).

5 – Testen (*Testing*)

In XP wird eine testgetriebene Softwareentwicklung (engl. *Test-Driven Development*) verfolgt. Dabei schreiben die Entwickelnden Tests, bevor sie den Code implementieren. Es wird eine Automatisierung dieser Tests angestrebt. Diese sorgen bei der Weiterentwicklung des Systems für Stabilität und stellen sicher, dass keine Seiteneffekte durch neue Implementierungen entstanden sind. Codeänderungen können durch automatisierte Tests schnell überprüft werden (Alam & Gühl, 2020, S. 139; Beck, 2004, S. 57–58; Brandt-Pook & Kollmeier, 2020, S. 29; Freedman, 2016, S. 215; Hanschke, 2017, S. 30).

6 – Refactoring

Unter Refactoring wird die kontinuierliche Optimierung des Codes verstanden. Mittels Refactoring wird Einfachheit, Eleganz und Kompatibilität des Systemcodes gewährleistet und die Software dadurch laufend verbessert (Alam & Gühl, 2020, S. 139; Beck, 2004, S. 58; Freedman, 2016, S. 215; Hanschke, 2017, S. 31) Diese Praktik gilt gemäss Freedman (2016) als Schlüsselelement von XP (S. 215).

7 – Programmieren in Paaren (*Pair Programming*)

Bei der Programmierung in Paaren sitzen zwei Teammitglieder zusammen vor einem Computer. Während das eine Teammitglied programmiert, beobachtet das andere simultan und führt dabei die Qualitätssicherung durch. Die Rollen sollten regelmässig getauscht werden, sodass das Vier-Augen-Prinzip optimal zur Anwendung kommen kann. Diese Praktik ist mit einem höheren Aufwand verbunden, die Codequalität kann jedoch dadurch gesteigert werden (Alam & Gühl, 2020, S. 139–140; Beck, 2004, S. 58–59; Brandt-Pook & Kollmeier, 2020, S. 29; Freedman, 2016, S. 215; Hanschke, 2017, S. 31).

8 – Gemeinsame Verantwortlichkeit (*Collective Ownership*)

Das Team ist als Ganzes für den Code verantwortlich. Jedes Teammitglied kann den Code ändern und weiterentwickeln und übernimmt somit individuell die Verantwortung für das System. Dies bedingt, dass der Code so programmiert ist, dass ihn jede Person

im Team verstehen kann. Diese Bedingung erhöht zugleich die Codequalität (Alam & Gühl, 2020, S. 139; Beck, 2004, S. 59–60; Brandt-Pook & Kollmeier, 2020, S. 29; Hanschke, 2017, S. 30).

9 – Fortlaufende Integration (*Continuous Integration*)

Alle Codeänderungen werden fortlaufend in das System integriert und getestet. Änderungen und Erweiterungen können so überprüft werden und es wird sichergestellt, dass alle Entwickelnden mit dem aktuellen Stand arbeiten. Die Integration erfolgt bestenfalls nach wenigen Stunden, spätestens aber nach einem Tag Entwicklungsarbeit (Alam & Gühl, 2020, S. 140; Beck, 2004, S. 60; Beck & Andres, 2005, S. 49).

10 – 40-Stunden Woche (*Sustainable Development*)

Überstunden sollen nur in Ausnahmefällen geleistet werden. Das Wohlergehen der Entwickelnden ist wichtig und XP lebt nach der agilen Philosophie, dass die Arbeit am besten geleistet werden kann, wenn das Team engagiert, energiegeladen, klar und konzentriert ist (Alam & Gühl, 2020, S. 140; Freedman, 2016, S. 215). Wichtig ist, dass die individuelle Belastungsgrenze nicht überschritten wird und nur so lange gearbeitet wird, wie Konzentration verfügbar ist, dabei gilt die 40-Stunden Woche als Richtwert (Beck, 2004, S. 60–61).

11 – Kunde[2] vor Ort (*Customer-on-Site*)

Die Kundin oder der Kunde ist in einem XP-Projekt integriert und steht bei Fragen und Unklarheiten dem Team zur Verfügung. Das regelmässige Feedback der Kundin oder des Kunden gilt als entscheidender Erfolgsfaktor, um das Produkt während der Entwicklung fachlich prüfen, bewerten und optimieren zu können (Alam & Gühl, 2020, S. 140; Brandt-Pook & Kollmeier, 2020, S. 29; Freedman, 2016, S. 216). Unter einer echten Kundin oder einem echten Kunden wird gemäss Beck (2004) eine Person verstanden, die das System letztendlich nutzen wird (S. 61).

[2] Der Name der Praktik bleibt unverändert. Daher wird hier auf eine gendergerechte Sprache verzichtet.

12 – Programmierstandards (*Coding standards*)

Damit die Praktik der gemeinsamen Verantwortlichkeit gut funktioniert, bedarf es laut Brandt-Pook und Kollmeier (2020) sogenannten Programmierstandards (S. 29). Diese Standards sollen die Kommunikation fördern (Beck, 2004, S. 62). Vom gesamten Team werden dieselben Standards verwendet. Diese sollen die Zusammenarbeit und das Zurechtfinden im Code erleichtern (Alam & Gühl, 2020, S. 139).

Die zwölf Praktiken können als Anleitung für die Ausführung agiler Produktentwicklung in einem XP-Projekt betrachtet werden.

2.4 Aktueller Stand der Verbreitung und Anwendung von Extreme Programming

Für den aktuellen Stand wird als Ausgangslage die Verbreitung von Extreme Programming in der Schweiz betrachtet. Wie in den Swiss Agile Studien von Meier und Kropp (2013a, 2015, 2017) zu erkennen ist, nimmt der Einsatz von Extreme Programming als Methode in Firmen ab. In der ersten durchgeführten Studie wird ersichtlich, dass 51% aller agilen Unternehmen Scrum anwenden. XP wird im Vergleich lediglich zu 5% angewandt (2013a, S. 12). In der darauffolgenden Studie erhöhte sich die Verbreitung von Scrum auf 59% und der Einsatz von XP reduzierte sich auf 3%. Scrum ist gemäss der Studie die mit Abstand am häufigsten angewendete Methode in den befragten Schweizer Unternehmen (2015, S. 10). Die neuste Studie zeigt nach wie vor Scrum als Spitzenreiter. Zu 52% kommt diese Methodik in agilen Unternehmen zum Einsatz. Extreme Programming hingegen ist mit 0% unter den Befragten (n = 90) gar nicht mehr vertreten (2017, S. 8).

Derselbe Sachverhalt lässt sich auch auf internationaler Ebene beobachten. Der jährlich publizierte State of Agile Report gibt Einblicke zum Thema Agilität in Unternehmen und dem Einsatz agiler Methoden. Aus den Reports geht hervor, dass die Befragten am häufigsten aus Nordamerika stammten, gefolgt von Europa und Asien (VersionOne Inc., 2012; 2013, 2014, 2015, 2016, 2017, 2018, 2019). In Abbildung 3 werden die Resultate der Studien dargestellt. Was in den State of Agile Reports auffällt, ist die hybride Methode Scrum/XP. Deren Tendenz ist zwar sinkend, dennoch bereits früh vertreten im Vergleich zu den Schweizer Studien, in welchen sie erstmals 2016 thematisiert wurde und damals mit 17% vertreten war.

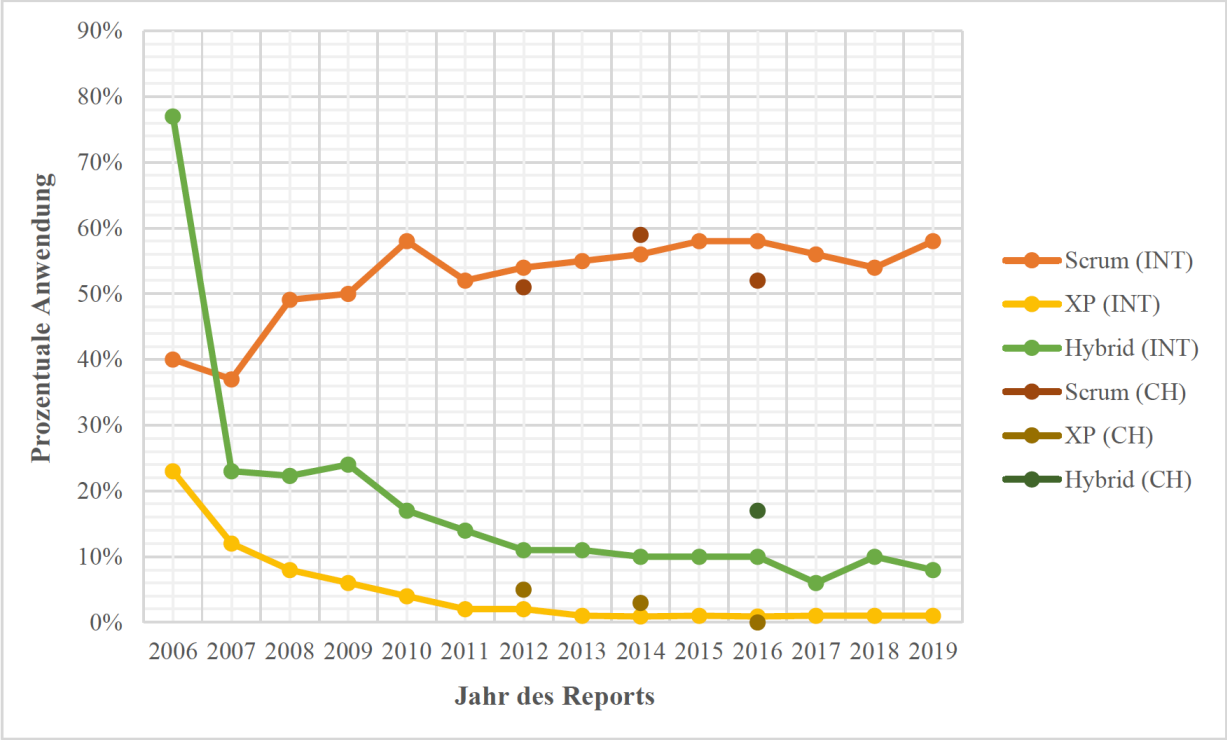

Abbildung 3: Prozentuale Anwendung der Methoden Scrum, XP und Hybrid (Eigene Darstellung in Anlehnung an Meier & Kropp, 2013a, 2015, 2017; VersionOne Inc., 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019)

Die Legende der Abbildung beinhaltet sowohl die Angaben auf internationaler Ebene (INT), als auch jene für die Schweiz (CH).

2.5 Häufigste XP-Praktiken im Einsatz

Wie in Abbildung 3 ersichtlich, kann basierend auf den vorhandenen Daten die Annahme getroffen werden, dass die Verbreitung und Anwendung von Extreme Programming sowohl national wie international die gleichen Tendenzen aufweisen. Aufgrund dieser Erkenntnis werden nachfolgend die häufigsten Praktiken im Einsatz auf internationaler Ebene visualisiert. Es wurde bewusst auf eine Visualisierung auf nationaler Ebene verzichtet, da die drei verfügbaren Studien wenig Datengrundlagen bieten, um eine aussagekräftige Erkenntnis gewinnen zu können. Zudem ist die Aktualität der Swiss Agile Studien nicht gewährleistet.

Zur Erstellung der Grafik wurden in den State of Agile Reports lediglich die Praktiken berücksichtigt, welche eindeutig XP zugeordnet werden konnten. Dabei resultierte eine durchschnittliche Anwendung der Praktiken über 14 Jahre hinweg (2006–2019). Die Berechnungsgrundlage kann dem Anhang entnommen werden.

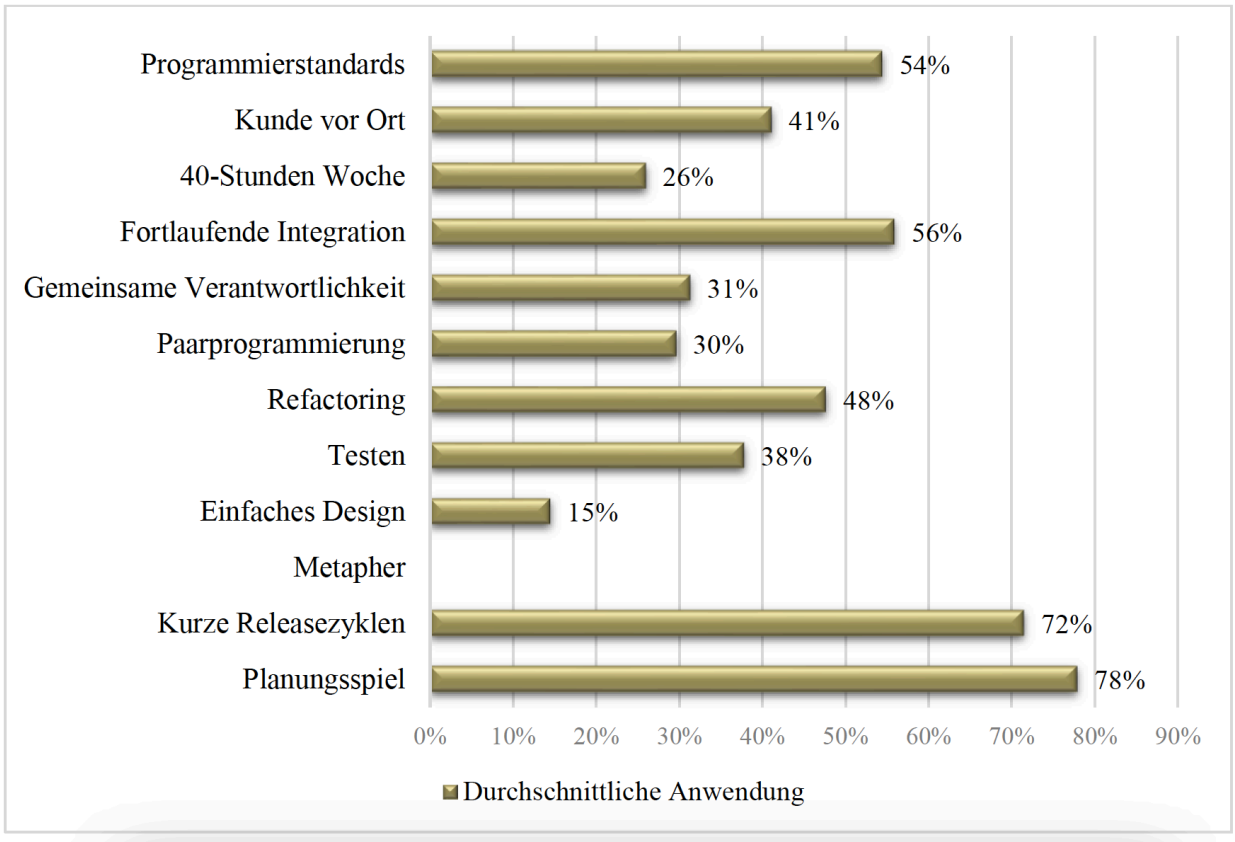

Abbildung 4: Durchschnittliche Anwendung der XP-Praktiken auf globaler Ebene (Eigene Darstellung in Anlehnung an VersionOne Inc., 2006; 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019)

Wie aus der Grafik hervorgeht, ist die Metapher die einzige Praktik, welche nie im Report erschienen ist. Praktiken wie das Planungsspiel, kurze Releasezyklen, fortlaufende Integration und Programmierstandards sind mit über 50% häufig vertreten.

2.6 Zusammenfassung der Erkenntnisse und Beantwortung der Teilfrage 1

Das Kapitel 2 hatte zum Ziel, die Teilfrage 1 zu beantworten, welche wie folgt lautete:

Was ist Extreme Programming, welche Praktiken gibt es und wie häufig werden sie in der Praxis angewandt?

Zusammenfassend kann Extreme Programming als eine leichtgewichtige Softwareentwicklungsmethode bezeichnet werden, deren Einsatz in Projekten mit volatilen Anforderungen besonders zielführend sein kann (Beck & Andres, 2005, S. 3). Wie aus dem Kapitel 2.3 hervorgeht, beinhaltet die Methode zwölf Praktiken. Bis auf die Praktik der «Metapher», welche einer gemeinsamen, verständlichen Sprache im Projekt dient, sind alle Praktiken noch im Einsatz.

Zur Analyse der Verbreitung von XP als Methode und der XP-Praktiken wurden auf nationaler Ebene die Swiss Agile Studien näher betrachtet. Um einen internationalen Vergleich herzustellen, wurden die State Of Agile Reports ebenfalls analysiert. Die Recherche hat offengelegt, dass sich derselbe Sachverhalt sowohl auf nationaler als auch internationaler Ebene gleichermassen manifestiert. In den analysierten Reports gilt Scrum als am häufigsten verbreitet. Wie bereits im Kapitel 1.2 erwähnt, wird Scrum durch Schwaber und Sutherland (2020) im Scrum Guide als Rahmenwerk bezeichnet. Wie die Autoren schreiben, können innerhalb dieses Frameworks verschiedene Methoden zum Einsatz kommen. Scrum bildet somit eine Hülle zur agilen Produktentwicklung und «definiert klare Regeln bzgl. Projekt- und Teamorganisation sowie dem Requirements-Management» (Meier & Kropp, 2013b, S. 82). Auch Mushtaq und Qureshi (2012) erwähnen, dass Scrum keine Angaben dazu macht, wie die Software entwickelt werden kann (S. 39).

Kritisch festzuhalten gilt es, dass aus den Studien nicht hervorgeht, wie «extrem» die Praktiken angewendet wurden. Da diese jedoch unter agilen Praktiken aufgeführt wurden, hat die Autorin die Annahme getroffen, dass es sich dabei im Ursprung um XP-Praktiken handelt. Diese Annahme wird durch die Aussage der VersionOne Inc. (2016) bestärkt: « While the usage of XP as an independent methodology continues to decrease (<1%), the practices associated with XP are still prevalent» (S. 11). Die Höhe der Prozentzahl angewandter XP-Praktiken steht somit im Widerspruch zu dem auf dem Nullpunkt angelangten Einsatz der Methodik. Mögliche Gründe und Ursachen werden in den folgenden Kapiteln eruiert.

3 Methodisches Vorgehen

In diesem Kapitel wird das methodische Vorgehen erläutert. Die Unterkapitel teilen sich pro Teilfrage ein. Sowohl für die Auswertung der Literaturrecherche als auch für die Auswertung der Interviews wird eine qualitative Inhaltsanalyse durchgeführt. Der Forschungsansatz des Vorgehens ist qualitativ, da die Untersuchung in die Tiefe geht und den Sachverhalt detailliert beschreiben soll. Zudem hat die Untersuchung einen explorativen Charakter, was eine Offenheit gegenüber unerwarteten Befunden gewährleistet (Döring & Bortz, 2016, S. 192).

3.1 Literaturrecherche für Teilfrage 2

Für die Erarbeitung des literarischen Teils werden elektronische Ressourcen, welche die Fachbereiche Informationswissenschaft & IT sowie Wirtschaft & Management behandeln, konsultiert. Beim Vorgehen stützt sich die Autorin auf den empfohlenen Literaturrecherche-Prozess von vom Brocke et al. (2009). Für die Recherche im Rahmen dieser Thesis sind aus Sicht der Autorin das Festlegen der Datenbanken, die Definition von Suchstrings und die zeitliche Eingrenzung der Recherche relevant. Zusätzlich wird eine sogenannte Rückwärtsrecherche vorgenommen, um auch ältere relevante Literatur berücksichtigen zu können (Webster & Watson, 2002, S. xvi). Namentlich werden die Datenbanken Google Scholar, IEEE Xplore, Springer und WISO für die Literaturrecherche ausgewählt, da eine erste Analyse ergeben hat, dass Literatur mit Relevanz vorhanden ist. Ergänzend wird auf Swisscovery relevante Literatur bestellt, welche nicht in digitaler Form verfügbar ist.

Primär findet die Suche über Google Scholar statt. Scholar deckt ein breites Spektrum diverser wissenschaftlicher Quellen ab und liefert bei der Suche auch Resultate aus den weiteren definierten Datenbanken. Berücksichtigt wird bei der Recherche nur Literatur, auf welche der Zugriff via Open Source oder über die FH Graubünden möglich ist. Zeitlich wird die Recherche auf Resultate mit dem Publikationsjahr 2016–2021 eingeschränkt, sollten dabei keine oder sehr wenige Resultate gefunden werden, wird die Einschränkung gelockert oder für die entsprechende Suchabfrage gar entfernt. Mit der oben erwähnten Rückwärtsrecherche werden zusätzlich relevante Quellen aus den Suchresultaten ermittelt, in dem das Literaturverzeichnis analysiert wird.

Für die Recherche werden entsprechende Suchstrings definiert. Da die Autorin Ursachen und Gründe für den geringen Einsatz von XP in der Praxis ermittelt, ist der Begriff «Extreme Programming» fester Bestandteil jeder Suchabfrage. Die Suchstrings enthal-

ten Begriffe sowohl in der deutschen als auch in der englischen Sprache, dies, weil zum Thema vermehrt Literatur in der englischen Sprache vorhanden ist. Zusätzlich werden Suchstrings definiert, welche spezifisch die Herausforderungen der einzelnen XP-Praktiken ermitteln sollen. Ein Beispiel für einen solchen Suchstring ist:

```
"Extreme Programming" AND "Continuous Integration" OR "Kontinuierliche Integration" OR "Fortlaufende Integration" AND Herausforderung?
```

Die Autorin arbeitet bei der Definition der Suchstrings mit sogenannten booleschen Operatoren, diese spezifizieren die Suchabfrage und grenzen die Suchresultate ein. Das Fragezeichen, wie es im String-Beispiel ersichtlich ist, wird dabei als Platzhalter verwendet. So will die Autorin sowohl die Ein- als auch Mehrzahl des entsprechenden Begriffes abfangen. Da das Thema sehr praxisbezogen ist, wird bei Bedarf auch Graue Literatur in Form von Publikationen und Fachberichten herangezogen.

3.2 Teilstrukturierte Interviews für Teilfrage 3

Die Daten für die Teilfrage 3 werden mittels Leitfadeninterviews erhoben. Wie Gläser und Laudel (2010) erklären, gelten solche Interviews als nichtstandardisiert, kommen jedoch mit Vorgaben für die interviewende Person einher. Deshalb sind sie auch als teilstandardisierte Interviews bekannt (S. 41). Ein Leitfadeninterview enthält alle Fragen, die im Verlauf des Gesprächs beantwortet werden müssen. Dabei ist jedoch weder die Reihenfolge noch die exakte Formulierung der Fragen vorgegeben. Das Leitfadeninterview soll möglichst einem natürlichen Gesprächsverlauf angeglichen werden. Dabei kann es durchaus vorkommen, dass die interviewte Person ein Thema bereits aufgreift, bevor es laut Fragebogen an der Reihe wäre. In einem Leitfadeninterview kann darauf flexibel reagiert werden und die Person muss nicht unterbrochen werden (Gläser & Laudel, 2010, S. 42). Die Erkenntnisse aus der Teilfrage 2 sollen dabei helfen, den Leitfaden zu erstellen. In den Interviews werden Erfahrungen und Beobachtungen von Agile Coaches einzeln untersucht. Die Konzeption des Leitfadens sowie die Auswahl der interviewten Personen und die Durchführung der Interviews werden in den Kapiteln 5.1 und 5.2 näher erläutert.

3.3 Qualitative Inhaltsanalyse zur Auswertung für Teilfrage 2 und 3

Die relevanten Quellen sowie die Transkripte aus den Interviews unterzieht die Autorin einer qualitativen Inhaltsanalyse. Diese führt sie in Anlehnung an das Lehrbuch von Gläser und Laudel (2010) durch. Wie die Verfassenden dort schreiben, wird für das Vorgehen, in welchem den Texten relevante Daten entnommen werden, auch der Begriff

Extraktion verwendet. Durch die qualitative Inhaltsanalyse werden notwendige Informationen demnach aus den Quellen extrahiert und strukturiert (S. 199–200). Damit diese Informationen entsprechend zugeordnet werden können, wird, wie im Lehrbuch suggeriert, mit sogenannten Kategorien gearbeitet (S. 200–201).

Das Vorgehen bezeichnet die Autorin als deduktiv/induktiv. Die Kategorien werden deduktiv, also aus der Literatur, definiert. Das Kategoriensystem dieser Arbeit wird dabei basierend auf den Kernelementen von XP konzipiert. Hierbei wird pro XP-Praktik eine gleichnamige Kategorie erstellt. Nebst den daraus resultierenden zwölf Kategorien werden zwei weitere Kategorien definiert. Die eine trägt den Namen «XP-Methode», dieser werden Inhalte zugeordnet, die der Methode selbst und nicht einer einzelnen Praktik zugewiesen werden können. Die andere wird «Hybride Vorgehensmodelle» genannt. Letztere Kategorie hat sich bei der Erarbeitung der Theorie herauskristallisiert, wonach XP häufig in hybrider Form in der Praxis angewendet wird. Die 14 Kategorien bilden die Grundlage des Suchrasters für die Literaturrecherche. Bei Bedarf wird die Autorin im Verlauf der Analyse weitere Kategorien hinzufügen. Die Ausprägungen der einzelnen Kategorien werden induktiv, also direkt aus dem Datenmaterial, erhoben.

Damit eine Zuordnung der Texteinheiten zum analysierten Material möglich ist, braucht es einen Kodierleitfaden. Wie Mayring (2015) schreibt, muss festgelegt werden, wann eine Texteinheit einer Kategorie eindeutig zugeordnet werden kann. Dafür erwähnt der Autor das folgende Verfahren als zielführend:

1. «Definition der Kategorien
   Es wird genau definiert, welche Textbestandteile unter eine Kategorie fallen.
2. Ankerbeispiele
   Es werden konkrete Textstellen angeführt, die unter eine Kategorie fallen und als Beispiele für diese Kategorie gelten sollen.
3. Kodierregeln
   Es werden dort, wo Abgrenzungsprobleme zwischen Kategorien bestehen, Regeln formuliert, um eindeutige Zuordnungen zu ermöglichen.» (Mayring, 2015, S. 97)

Basierend auf diesem Verfahren wurde ein Kodierleitfaden erstellt. Dieser soll als Grundlage für die Zuordnung der Kategorien dienen und kann dem Anhang entnommen werden.

Wie Gläser und Laudel (2010) schreiben, bedeutet eine Extraktion «den Text zu lesen und zu entscheiden, welche der in ihm enthaltenen Informationen für die Untersuchung relevant sind. Diese Informationen werden den Kategorien des Suchrasters zugeordnet, das heisst unter der entsprechenden Kategorie eingetragen» (S. 200). Bei der Extraktion

werden die Textpassagen ausgelesen und in Form von sogenannten Merkmalsausprägungen der passenden Kategorie zugeordnet. «Zur Offenheit des Kategoriensystems gehört, dass die Merkmalsausprägungen frei verbal beschrieben werden» (Gläser & Laudel, 2010, S. 201). Die Autorin definiert die Merkmalsausprägungen fortlaufend während der Analyse.

Die qualitative Inhaltsanalyse wird mit dem Tool MAXQDA 2020 durchgeführt. Das vollständige Kategoriensystem mit den jeweiligen Ausprägungen sowohl für die Teilfrage 2 als auch für die Teilfrage 3 kann dem Anhang entnommen werden.

4 Gründe aus der Literatur für die geringe Verbreitung von XP

In diesem Kapitel werden mögliche Gründe aus der Literatur, die eine geringe Verbreitung von XP erklären, eruiert. Die im Rahmen der Datenerhebung für die Teilfrage 2 durchgeführte Literaturrecherche umfasst insgesamt 33 Suchabfragen. Das Total der Suchergebnisse beläuft sich auf 2450 Quellen. Um deren Relevanz für die Arbeit zu ermitteln, geht die Autorin wie von vom Brocke et al. (2015) beschrieben vor, in dem sie ein sogenanntes Screening der Literatur durchführt. «Such screening is usually done by reading the publications' abstracts» (S. 211). Die Autorin hat sich dafür entschiedenen, bei der Literaturrecherche nebst dem Abstract auch das Fazit durchzulesen. Erscheint die Literatur als relevant, wird zusätzlich das Quellenverzeichnis berücksichtigt und so allfällige ältere bedeutsame Literatur betrachtet. Nach diesem Vorgehen resultierten gesamt 34 Quellen als relevant, diese setzen sich aus 24 Quellen aus der direkten Suche und zehn Quellen aus der Rückwärtssuche zusammen. Das Vorgehen der Literaturrecherche erfolgt systematisch und ist im Anhang protokolliert. Die Autorin betrachtet die Literaturrecherche als beendet, sobald der Sättigungsgrad erreicht ist. Laut Webster und Watson (2002, S. xvi) und vom Brocke et al. (2015, S. 211) zeichnet sich dies ab, wenn bei der Recherche keine neuen Erkenntnisse mehr ermittelt werden können.

4.1 Ergebnisse der Literaturrecherche

Folgend werden die Ergebnisse der qualitativen Inhaltsanalyse näher erläutert. Die eruierten Gründe werden systematisch in den jeweils zugeordneten Kapiteln inhaltlich verdichtet und zusammenfassend dargestellt.

4.1.1 Ergebnisse einzelne Praktiken

Dieses Kapitel beinhaltet die Befunde zu den einzelnen Praktiken. Für einen besseren Lesefluss hat sich die Autorin entschieden, hier nicht mit Unterkapiteln, sondern mit Titeln zu arbeiten. In der Arbeit werden keine Überschriftsebenen tiefer als drei (X.X.X) verwendet. Damit für die Lesenden die möglichen Ursachen direkt ersichtlich sind, werden diese im Text fett hervorgehoben.

Ergebnisse Kategorie Planungsspiel

Die XP-Praktik Planungsspiel beinhaltet, wie der Name bereits antizipieren lässt, Aspekte der Planung und Organisation der agilen Softwareentwicklung. Dazu gehört auch das

Standup Meeting, welches für eine regelmässige Absprache sorgt. Dabei erläutert Proba (2021), dass spezifisch beim Standup Meeting Schwierigkeiten auftreten können, wenn die **Moderation** schlecht ist, da dadurch auch ineffiziente Gesprächsführungen die Folge sind. Weiter nennt der Autor die mangelnde **Disziplin** als Ursache für den fehlenden Informationsaustausch. Denn werden Standup Meetings unregelmässig oder mangelhaft diszipliniert durchgeführt, können wichtige Informationen entfallen (S. 298).

Auch kann die **Organisation** im Unternehmen den Einsatz des Planungsspiels erschweren. Wie Sfetsos et al. (2006) schreiben, sind vor allem intern ungenau definierte Prozesse und verteilte Teams Faktoren für eine erschwerte Kommunikation und einen erhöhten Koordinationsaufwand im Zusammenhang mit der XP-Praktik (S. 282).

Sowohl Kunwar (2019, S. 57) als auch Ibrahim et al. (2020, S. 166) nennen zudem hinsichtlich des **Anforderungsmanagements**, dass XP nicht-funktionale Anforderungen vernachlässigt und der Fokus auf funktionalen Anforderungen liegt.

Ergebnisse Kategorie Kurze Releasezyklen

Einzige Ursache, welche als Hinderungsgrund dieser Praktik in der Recherche eruiert werden konnte, waren die fehlenden **Kompetenzen**. Um funktionierende Software in kurzen Releasezyklen liefern zu können, bedarf es gemäss Sfetsos et al. (2006) erfahrenes und qualifiziertes Personal (S. 288).

Ergebnisse Kategorie Metapher

Aus der Literatur geht hervor, dass die Praktik der Metapher oft mit einer **Unklarheit** einhergeht (Donick, 2020, S. 28–29; Keller, 2019, S. 38; Sfetsos et al., 2006, S. 290). Wie Sfetsos et al. (2006) schreiben, wird die Metapher in der Praxistheorie unklar dokumentiert (S. 290). Dies führt dazu, dass das Konzept der Systemmetapher für Anwendende nebulös ist (Keller, 2019, S. 38). Die Metapher wird laut Donick (2020) unterschiedlich interpretiert, was konträr zu deren Intention ist, da diese ein gemeinsames Verständnis schaffen sollte (S. 28–29). Nebst dieser Unklarheit, nennen Sfetsos et al. (2006) auch die mangelnde **Erfahrung** sowie die **kulturellen Differenzen** zwischen agil und traditionell arbeitenden Teams als limitierende Faktoren der Metapher (S. 290).

Ergebnisse Kategorie Einfaches Design

Zur Praktik des einfachen Designs werden Faktoren wie **Zeit** und **Priorisierung** als Hindernisse für eine Implementierung genannt. Zeitliche Einschränkungen sowie Verschiebungen der Prioritäten im Entwicklungsprozess erschweren den Einsatz dieser Praktik. Hinzu kommt, dass bei verteilten Teams fehlende oder unzureichende **Dokumentationen** verhindern können, dass die Praktik angewendet werden kann. Auch fehlen **Anleitungen** aus der Theorie, was den Einsatz der Praktik zusätzlich erschwert (Sfetsos et al., 2006, S. 286–287).

Ergebnisse Kategorie Testen

Wie Ott (2018) schreibt, ist das Streben nach einer kompletten Testabdeckung ein grösseres Unterfangen. Deshalb wird empfohlen, zeitnah damit zu beginnen, um die Vorteile davon nutzen zu können. Dabei ist eine Automatisierung anzustreben, da der Aufwand ansonsten enorm ist (S. 49). Eine testgetriebene Entwicklung ist mit **Aufwand** verbunden (Qureshi, 2012, S. 151). Je mehr Tests fehlschlagen, desto grösser wird der Aufwand (Anwer et al., 2017, S. 4; Ibrahim et al., 2020, S. 165). Dieser Aufwand kann sich auch negativ auf die **Kosten** auswirken (Qureshi, 2012, S. 151). Zudem verlangt eine testgetriebene Entwicklung qualifizierte Personen (Sfetsos et al., 2006, S. 286). Die Praktik erfordert entsprechende **Kompetenzen** von Programmierenden, die normalerweise als Aufgabe der Testverantwortlichen betrachtet werden (Anwer et al., 2017, S. 4; Ibrahim et al., 2020, S. 164). Hinzu kommt, dass die Praktik laut Ibrahim et al. (2020) den Entwicklungsprozess verlangsamt (S. 164).

Ergebnisse Kategorie Refactoring

Die Praktik des Refactoring erfordert entsprechende **Kompetenzen** der Entwickelnden. Sie setzt fortgeschrittene Kenntnisse über Programmiertechniken voraus und bedingt ein Verständnis darüber, wie und weshalb Refactoring durchgeführt wird. Letzteres liegt nicht direkt auf der Hand, da die Praktik selbst keine zusätzlichen Funktionen generiert (Nanthaamornphong & Carver, 2017, S. 356). Dadurch, dass somit kein Fortschritt direkt sichtbar ist, fehlt dafür auch schneller die **Akzeptanz** der Kundin und des Kunden sowie des Managements (Fausten, 2016, S. 23).

Wie Sfetsos et al. (2006) schreiben, wurde in einem durchgeführten Interview bemängelt, dass die XP-Theorie keine **Anleitungen** und detaillierten Richtlinien zur Durchführung einer erfolgreichen Anwendung dieser Praktik liefert (S. 286). Daher ist laut Cockburn

(2002) erhöhte Disziplin gefragt, um Refactoring konsequent durchzuführen. Was den Einsatz der Praktik zusätzlich erschwert, ist, dass keine Mechanismen der XP-Methodik die Praktik verstärken. Diese Tatsache verlangt ein hohes Mass an **Durchhaltevermögen** (S. 141).

Ein weiterer limitierender Faktor dieser Praktik ist die **Abhängigkeit**. So nennen Nanthaamornphong und Carver (2017) Abhängigkeiten zu Unit-Tests und dem Architekturdesign als hindernde Faktoren für den Einsatz der Praktik. Der Prozess des Refactoring wird als schwierig bezeichnet, wenn die Unit-Tests nicht gut konzipiert sind. Gründliche Unit-Tests werden gar als Voraussetzung für diese Praktik genannt. Des Weiteren ist ein gutes Architekturdesign die Grundlage für das Refactoring. Ist dieses nicht gegeben, bedarf es einer Überarbeitung oder eines Neuentwurfs, damit ein Refactoring stattfinden kann (S. 356).

Auch fällt Refactoring schwer, wenn die Entwickelnden mit sogenanntem **Legacy Code** arbeiten und den Code nicht von Grund auf neu entwickeln. Fehlt bei dem bereits vorhandenen Code dann noch zusätzlich eine angemessene Testbasis, dann kann nicht sichergestellt werden, dass ein Refactoring keine Probleme in der bestehenden Codebasis verursacht hat (Nanthaamornphong & Carver, 2017, S. 356).

Ergebnisse Programmieren in Paaren

Diverse Faktoren wurden in der Literatur als hindernd für den Einsatz von Pair Programming genannt. Sfetsos et al. (2006) haben festgestellt, dass durch die Verteilung von Arbeit auf mehrere, **verteilte Teams** vor allem bei erhöhter Komplexität der Projekte Probleme entstehen können (S. 285). Wie Proba (2021) schreibt, kann auch eine unpassende Zusammenstellung der Programmierteams für eine erfolgreiche Programmierung in Paaren hinderlich sein (S. 285). Eine unpassende Zusammenstellung manifestiert sich dabei durch Differenzen in den **Kompetenzen**, dabei werden sowohl fachliche, wie die **Verfügbarkeit** der Programmierenden mit dem entsprechenden Know-How (Kunwar, 2019, S. 56; Tsyganok, 2016, S. 271), als auch sozial-kommunikative Kompetenzen, wie die Kommunikationsfähigkeit, in der Literatur erwähnt (Ibrahim et al., 2020, S. 166–167; Kunwar, 2019, S. 56; Sfetsos et al., 2006, S. 285). Kompetenzen letzterer Art scheinen für den Erfolg von Pair Programming eine entscheidende Rolle zu spielen. Aus der Literatur geht hervor, dass sozial-kommunikative Kompetenzen weniger greifbar sind. Deshalb gelten sie als am schwierigsten zu bewältigen (Tsyganok, 2016, S. 271). Der Mix der Persönlichkeiten innerhalb eines Teams ist ausschlaggebend für den Erfolg dieser Praktik (Dhoodhanath & Quilling, 2020; Kunwar, 2019, S. 58). Dabei verhindern

menschliche Probleme (Sfetsos et al., 2006, S. 285) wie antisoziale Persönlichkeiten den Einsatz von Pair Programming massgeblich (Kunwar, 2019, S. 56). Gemäss Kunwar (2019) kann paarweise zu programmieren auch als entmutigend empfunden werden, was sich demnach auf das **Wohlbefinden** auswirkt (S. 56). Tsyganok (2016) schreibt, dass das Wohlbefinden auch durch physische Einschränkungen beeinträchtigt sein kann. Darunter fallen der unzureichende Freiraum sowie unangemessene **Infrastruktur** für diese Praktik (S. 271). Hinzu kommt, dass laut Kunwar (2019) zwei Entwickelnde, die zusammen programmieren eine geringere **Produktivität** aufweisen, als wenn sie individuell arbeiten würden. Der Autor erwähnt in diesem Zusammenhang, dass statistische Beweise vorliegen, dass Pair Programming in etwa 15% höhere **Kosten** verursacht als die klassische Programmierung (S. 58). Gemäss Ibrahim et al. (2020) werden nebst den Kosten auch **Ressourcen** in Form von Zeit beansprucht; es dauert entsprechend länger, dieselben Features umzusetzen (S. 166–167). Auch können **Unterbrechungen** durch andere Entwickelnde dazu führen, dass die Pair Programming Session unproduktiv ist (Tsyganok, 2016, S. 271). Das Management betrachtet Programmierende als knappe Ressource und sieht daher vorzugsweise davon ab, die Anzahl der Mitarbeitenden für diese Praktik zu verdoppeln. Die **Managementakzeptanz** für diese Praktik ist demnach gering. Es benötigt eine Sensibilisierung des Managements, welches davon überzeugt ist, dass bei der Programmierung in Paaren eine Software in besserer Qualität entsteht (Kunwar, 2019, S. 58). Letztlich wurden auch Probleme, welche auf die **Kultur** zurückzuführen sind, als hindernd eruiert. Damit Pair Programming funktioniert, muss es «eine gute und faire Kultur der Kooperation bereits geben. Fehler müssen normal sein, der Austausch auf Augenhöhe üblich» (Hofert, 2018, S. 202). Auch kulturelle Probleme, die auf Unterschiede zwischen klassischer und agiler Entwicklung zurückzuführen sind, wurden als für diese Praktik störend ermittelt (Sfetsos et al., 2006, S. 285).

Ergebnisse Gemeinsame Verantwortlichkeit

Auer und Miller (2002) erwähnen, dass eine gemeinsame Verantwortlichkeit für den Code nicht der Norm in der Softwareentwicklung entspricht. Deswegen kann sich fehlendes **Verständnis** für diese Praktik im Widerstand der Entwickelnden äussern. Auch gibt es Entwickelnde, die fürchten, durch die Praktik ihren Code aufgeben zu müssen. In diesem Zusammenhang nennen die Autoren die Überwindung des Ego-Problems als schwierigste Hürde dieser Praktik. Eine gemeinsame Verantwortlichkeit für den Code funktioniert nicht ohne Vertrauen. Die Idee des exklusiven Eigentums am Code

muss überwunden werden. Damit diese Praktik erfolgreich angewendet werden kann, muss das **Mindset** entsprechend stimmen. Es muss daran geglaubt werden, dass gemeinsam bessere Ergebnisse produziert werden als alleine. Ist dies nicht der Fall, so ist die Praktik zum Scheitern verurteilt (S. 223).

Ergebnisse Fortlaufende Integration

Die Praktik der fortlaufenden Integration erweist sich laut Bass (2012) in grösseren Entwicklungsprojekten, welche eine erhöhte Komplexität aufweisen als herausfordernd. Wie eine befragte Person dem Autor erläutert, dauern bei grösseren, umfangreicheren Projekten die Regressionstests zu lange, um diese nach jeder Integration vollständig laufen zu lassen. Der **Aufwand**, welcher daher mit dieser Praktik in Verbindung gebracht wird, kann entsprechend erheblich sein (S. 6).

Ergebnisse 40-Stunden Woche

Hinsichtlich der 40-Stunden Woche nennen Sfetsos et al. (2006) die **Organisation** innerhalbdes Unternehmens sowie die **Projektart und -grösse** als limitierende Faktoren für eine erfolgreiche Anwendung der Praktik (S. 289).

Ergebnisse Kunde vor Ort

Diese Praktik setzt die **Anwesenheit** der Kundin oder des Kunden voraus. Aus der Literatur geht hervor, dass die Praktik wie in der Theorie beschrieben schwierig zu erreichen ist, da eine Vollzeit-Anwesenheit der Kundin oder des Kunden meist nicht gegeben ist (Sfetsos et al., 2006, S. 288). Vanhala und Kasurinen (2019) beobachteten jedoch, dass eine solche Vor-Ort-Präsenz nicht wie dokumentiert zwingend ist, wenn sich die Person über einen gemeinsamen digitalen Arbeitsbereich beteiligt. Eine solche **Beteiligung** hat laut den Autoren einen signifikanten Einfluss auf den Projekterfolg (S. 220). Sie wird jedoch gemäss Aichele und Schönberger (2016) öfters kritisiert (S. 86). Diese obligatorische Teamzugehörigkeit der Kundin oder des Kunden wird in XP als klare Herausforderung erkannt (Zykov, 2016, S. 67). Eine ständige **Verfügbarkeit** der Kundin oder des Kunden ist in Realität nicht immer möglich (Bibik, 2018, S. 9). Auch kann es schwierig sein, Zugang zu den geeigneten Personen zu finden (Sfetsos et al., 2006, S. 288), welche dann auch Vollzeit verfügbar sind (Kunwar, 2019, S. 55). Wenn die Kundin oder der Kunde nicht ständig verfügbar ist, kann sich das Projekt verlangsamen, da Entwickelnde auf Informationen warten oder aber sie raten, was die Anforderungen

sind und laufen Gefahr, etwas Falsches zu entwickeln. Beide Optionen fallen dabei negativ auf die Projektkosten zurück (Auer & Miller, 2002, S. 38). Ein weiterer Faktor, der als limitierend eruiert wurde, ist die **Interpretation** der Aussagen der designierten Person im Projekt, welche als repräsentativ für die Gesamtheit der Nutzenden betrachtet werden (Proba, 2021, S. 281). Die Voraussetzung der ständigen Verfügbarkeit schafft eine **Abhängigkeit** des Teams gegenüber der Kundin oder dem Kunden (Mushtaq & Qureshi, 2012, S. 40). Diese starke Abhängigkeit von Kundinnen und Kunden und Stakeholdern, kann für das Scheitern eines Projektes verantwortlich sein (Ibrahim et al., 2020, S. 167; Qureshi, 2012, S. 151; Qureshi & Bajaber, 2016, S. 5120). Kunwar (2019) nennt als Gefahr dieser Praktik, dass eine hohe Wahrscheinlichkeit besteht, dass unklare und fehlerhafte Anforderungen von einer einzelnen Person gesammelt werden. Diese Person ist anschliessend auch allein für die Entscheidungen des Unternehmens verantwortlich. Eine solche Entscheidungsbefugnis kann gemäss dem Autor als einschränkend für diese Praktik betrachtet werden (S. 55–56). Letztendlich ist diese Entscheidungsbefugnis mit einer grossen Verantwortung verknüpft, welche es zu übernehmen gilt. Wie Beck, Fowler und Kohnke (2001) schreiben, gibt es jedoch Kundinnen und Kunden, die eine solche **Verantwortung** nicht übernehmen möchten und sich weigern, Entscheidungen zu treffen. Extreme Programming ist darauf angewiesen, dass die Kundin oder der Kunde die Entscheidungen trifft, ansonsten funktioniert es nicht (S. 126). Letztendlich ist die Voraussetzung für eine Entscheidungskompetenz auch ein notwendiges, vertieftes **Domänenwissen** der designierten Person (Kunwar, 2019, S. 56).

Ergebnisse Programmierstandards

Für die Praktik der Programmierstandards werden Aspekte der **Erfahrung** und **Organisation** als hindernde Faktoren genannt. So nennen Sfetsos et al. (2006) organisatorische Probleme von grossen Unternehmen sowie die fehlende Erfahrung kleinerer Unternehmen als Gründe gegen diese Praktik (S. 289). Zudem kann ein «ineffizientes Handeln und hohe Aufwände bei **dogmatischer Verwendung** der Programmierrichtlinien» (Proba, 2021, S. 289) dazu führen, dass die Praktik scheitert. Darüber hinaus besteht die Möglichkeit, dass die Richtlinien im Team auf fehlende **Akzeptanz** stossen (Proba, 2021, S. 289).

4.1.2 Ergebnisse hybride Vorgehensmodelle

Aus der Literatur hat sich herauskristallisiert, dass Extreme Programming nicht selten in Kombination mit anderen agilen Vorgehensmethoden zum Einsatz kommt.

Vorwiegend werden XP-Praktiken in Scrum integriert, dies manchmal auch mit unterschiedlicher Namensgebung (Hanschke, 2017, S. 31). Signifikante Unterschiede zwischen Scrum und Extreme Programming erlauben es, die Methoden gleichzeitig erfolgreich anzuwenden: während sich XP auf die Entwicklungspraktiken konzentriert und die Managementpraktiken laut Smoczyńska et al. (2019) vernachlässigt, liegt in Scrum der Fokus auf den Managementpraktiken (S. 105). Dieser wesentliche Unterschied ermöglicht eine erfolgreiche **Kombination** der beiden Praktiken. Wie Krodel (2013) schreibt, sind einige der XP-Praktiken geeignet, Scrum inhaltlich zu präzisieren. Gemäss der Autorin «steuert XP eher mit ‹weichen Faktoren› Ergänzungspotenzial für den agilen Prozess Scrum bei» (S. 36). Krodel (2013) sieht darin die Begründung, «dass Scrum XP zunehmend verdrängt und nur die Techniken von XP erhalten bleiben» (S. 36). Auch durchlaufen die XP-Praktiken eine ständige **Evolution** (Kuhrmann et al., 2019, S. 27), entwickeln sich weiter und werden mittlerweile auch in anderen Vorgehensmodellen verwendet (Alam & Gühl, 2020, S. 140; Keller, 2019, S. 34). In sogenannten hybriden Modellen findet eine **Adaption** des Ursprungsmodells statt. Projektteams übernehmen dabei selektiv einige der Praktiken und passen diese an die Arbeitsumgebung an (Sfetsos et al., 2006, S. 272). Dies geschieht laut Kuhrmann et al. (2017) nach einem pragmatischen Ansatz (S. 39). Gemäss Aussagen einer Befragung der Autoren und der Autorin basiert eine solche Adaption nicht selten auf Erfahrungswerten aus vergangenen Projekten (S. 36). Kuhrmann et al. (2019) schreiben, dass es keine Einheitslösung für die Softwareentwicklung gibt und der Ansatz gewählt wird, welcher für die Herausforderungen im Projekt am geeignetsten ist. Dabei kommt es nicht selten vor, dass sich der geplante Ansatz entwickelt und dem Kernprinzip des ‹inspect and adapt› aus dem Agilen Manifest folgt (S. 26). Einige der Praktiken scheinen dabei öfter verwendet zu werden als andere. Dies begründet Dehn (2020) durch eine bessere Kombinationsmöglichkeit oder dadurch, dass bestimmte XP-Praktiken als Grundbaustein des Entwicklungsprozesses verwendet werden (S. 36). Hinzu kommt, dass die Teams eine gewisse **Flexibilität** aufweisen und daher den für sie am besten geeigneten Ansatz wählen (Kuhrmann et al., 2019, S. 27). Letzteres ist jedoch nicht immer der Fall. Bei der Auswahl des Vorgehens können **Einschränkungen** gegeben sein. Darunter fallen laut Kuhrmann et al. (2019) Aspekte wie die mangelnde Bereitschaft und das fehlende Verständnis für agile Entwicklung. Auch das Aufrechterhalten des bestehenden Geschäfts sowie der im Unternehmen bestehenden Philosophie und Standards können Restriktionen darstellen und die Auswahl des Vorgehens zunehmend einschränken. Ferner kann eine sogenannte **Neophobie**, bei welcher Personen Angst vor Neuem haben, auch als hindernd für den Einsatz von XP betrachtet werden (S. 27).

Wie Kuhrmann et al. bereits 2017 feststellen konnten, sind hybride Ansätze in der Praxis mittlerweile weit verbreitet. Sie sind zum **Mainstream** geworden und werden unabhängig von Unternehmensgrösse und Branche eingesetzt (S. 38–39).

4.1.3 Ergebnisse XP-Methode

In diesem Kapitel werden Aspekte thematisiert, welche nicht explizit einer Praktik oder einer hybriden Vorgehensweise zugeordnet werden konnten.

Pelrine et al. (2000) beschreiben XP als einen radikalen Ansatz der Softwareentwicklung. Die konsequente Anwendung von Extreme Programming bedarf demnach ein erhöhtes Mass an **Durchhaltevermögen** und Disziplin, um die Praktiken und Methode richtig zu erlernen (S. 1).

Wie bereits bei der Praktik der Gemeinsamen Verantwortlichkeit, setzt auch XP selbst ein entsprechendes **Mindset** voraus. So gehört laut Pelrine et al. (2000) das Verwerfen von Code zu einer der grössten Schwierigkeiten für Entwickelnde, da diese an ihrem Code hängen, dazu tendieren, besitzergreifend zu handeln und sich ungern davon trennen (S. 3). Hinzu kommt, dass ein Verständnis für agile Vorgehensweisen entwickelt werden muss, um erkennen zu können, wann der Einsatz einer Methode Sinn macht und wann nicht (Pelrine, 2011, S. 27). Weinrich et al. (2016) nennen als Herausforderung in diesem Zusammenhang das Schaffen eines Bewusstseins über agile Vorgehensweisen auch auf der Managementebene. Bei ihrer Befragung hat sich herausgestellt, dass möglicherweise die Rahmenbedingungen des Unternehmens für agile Vorgehensweisen ungeeignet sein können. Auch sehen nicht alle den Unterschied zwischen klassischen und agilen Vorgehensweisen (S. 87).

Auffallend ist hierbei, dass die **Entscheidungsgewalt** der Vorgehensweise nicht alleine beim Team zu liegen scheint, so weisen Kuhrmann et al. (2017) darauf hin, dass es durchaus bereits zu Beginn des Projektes Standards geben kann, die angewendet werden (S. 37). Auch wurde in der Literatur erwähnt, dass XP entwicklungszentriert ist und **Managementpraktiken** dabei vernachlässigt werden (Ibrahim et al., 2020, S. 167; Mushtaq & Qureshi, 2012, S. 40; Smoczyńska et al., 2019, S. 105). Wie Alpar et al. (2019) schreiben, ist aus Sicht des Managements die fehlende **Projektkontrolle** ein Nachteil von Extreme Programming (S. 371). Des Weiteren wird geäussert, dass der Einsatz der Methode, vor allem in kleineren Projekten, negative Auswirkungen auf die **Wirtschaftlichkeit** haben kann (Aichele & Schönberger, 2016, S. 86).

Die fehlende **Akzeptanz** für die Methode sowohl seitens Entwicklung als auch managementseitig kann als weiterer Grund für den geringen Einsatz der Methode betrachtet

werden. XP geht gegen Gewohnheiten von Programmierenden und zwingt sie, mit Menschen zu interagieren (Auer & Miller, 2002, S. 45). Auch hat sich in der Praxis gezeigt, dass das IT-Management den gruppenversierten Ansatz nicht unterstützt, da sie es bevorzugen, Entscheidungen zu kontrollieren (Hekkala et al., 2017, S. 5876). Bei einem Interview, welches Conboy et al. (2011) durchgeführt haben, hat ein Manager explizit die Angst vor Machtverlust einiger Manager in Zusammenhang mit XP genannt (S. 54). Wie Marquardt (2011) schreibt, verändert XP «die Zuständigkeiten und Rollen im Vergleich zu anderen Vorgehensmodellen und stösst damit in seiner Vollausprägung noch auf Skepsis» (S. 99). Der Autor schreibt, dass daher XP meist nur von Entwicklungsseite her weiterverfolgt wird. Es ist jedoch dieser Ansatz, der mehr Konflikte erzeugen kann als er löst, «da die wichtige Kommunikation und Rückkopplungsschleife zum Kunden nicht nur offen bleiben, sondern durch gegenseitiges Unverständnis noch zusätzlich behindert werden» (Marquardt, 2011, S. 99). Zudem befürchten Entwickelnde, dass durch die Methode eine **Transparenz** entsteht, welche persönliche Unzulänglichkeiten ans Licht bringen könnte (Conboy et al., 2011, S. 49– 51). Des Weiteren kann es auch sein, dass bei agilen Methoden, die top-down eingeführt werden die **Motivation** der Entwickelnden für die Anwendung der Methode geringer ausfällt. Einige empfinden die Einführung agiler Methoden als aufwändig, komplex und zeitintensiv (Conboy et al., 2011, S. 54).

Ein weiterer Faktor, der in der Literatur als hindernd genannt wird, ist die **Skalierbarkeit** der Methode. Mushtaq und Qureshi (2012) vertreten die Meinung, dass XP nicht für mittel bis grosse Projekte geeignet ist (S. 40). Da es schwierig ist, XP in grossen Organisationen und Teams zu implementieren, rentiert der Einsatz der Methode eher bei Projekten mit geringerem Risiko und einer reduzierten Teamgrösse (Ibrahim et al., 2020, S. 167). Cockburn (2002) spezifiziert diese Aussage und deutet darauf hin, dass XP in der Komplexität des Projektes sehr wohl skalierbar ist, jedoch nicht in der Teamgrösse (S. 142). Qureshi (2012, S. 151) sowie Qureshi und Bajaber (2016, S. 5120) begründen die eingeschränkte Skalierbarkeit auch dadurch, dass die **Dokumentation** in XP auf ein Minimum reduziert wird, waseine Wiederverwendbarkeit einschränken kann. Wie Cockburn (2002) schreibt, verlässt sich XP auf **implizites Wissen**. Wird mit mehr Programmierenden als empfohlen gearbeitet, so ist es notwendig, unter anderem die Verwendung der Dokumentation anzupassen, da sich der Koordinationsbedarf mit zunehmenden Teammitgliedern erhöhen wird (S. 142). Sind die **Teams** zusätzlich **verteilt**, zeichnet sich XP als eher ungeeignet aus (Ibrahim et al., 2020, S. 167; Qureshi & Bajaber, 2016, S. 5120). Wie Zykov (2016) erwähnt, ist XP abhängig von mündlicher **Kommunikation** (S. 67). Das spricht zusätzlich gegen eine Verteilung der Teams. Des

Weiteren behauptet Bibik (2018), dass die erhöhte **Produktivität** der Methode sich nur in speziellen Fällen manifestiert. In den anderen Fällen soll die Entwicklung langsamer sein als bei anderen Methoden (S. 8).

Eine weitere Herausforderung sind die **Abhängigkeiten** der einzelnen Praktiken. Deren Interdependenz kann zu Restriktionen führen, was kontraproduktiv für den Einsatz der Methode ist (Zykov, 2016, S. 67).

Entscheidender Faktor für den Einsatz der Methode ist auch die **Kompetenz**. So halten Conboy et al. (2011) fest, dass nicht jede Person dafür geeignet ist, mit einer Kundin oder einem Kunden zu interagieren. Eine gute Kommunikation ist dafür zentral, gleichermassen ist es wichtig, abwägen zu können, welche Informationen vertraulich behandelt werden müssen und welche nicht (S. 52). Nebst dem Erwerb von Fähigkeiten, die eine gute Kommunikation ermöglichen, bedarf es für XP auch fachlichen Wissens (Pelrine et al., 2000, S. 5). Ein agiles Umfeld weist die Tendenz auf die Grenzen zwischen den Rollen zu verwischen und erfordert deswegen ein breites Spektrum an Kompetenzen, gegensätzlich einer Spezialisierung. Dieser Aspekt birgt Schwierigkeiten, wie die Rekrutierung geeigneter Personen (Conboy et al., 2011, S. 51).

Zu guter Letzt kann eine **Eigeninterpretation** der Praktiken gekoppelt mit einem mangelnden Verständnis der Software-Architektur schnell zu einer inkonsistenten Entwicklung führen. Programmierende sollten daher reflektiert handeln und benötigen ein erfahrenes Urteilsvermögen (Pelrine et al., 2000, S. 4).

4.2 Zusammenfassung der Erkenntnisse und Beantwortung der Teilfrage 2

Das Kapitel 4 hatte zum Ziel, die Teilfrage 2 zu beantworten, welche wie folgt lautet:

Welche Gründe finden sich in der Literatur für die geringe Anwendung der Methodik?

Aus der Analyse resultiert, dass die Aspekte, welche durch die Recherche ermittelt werden konnten, vielfältig sind. Diese begründen eine unterschiedlich starke Anwendung der Methode und der einzelnen Praktiken. Da die Gründe aus der Literatur zahlreich sind, werden diese in Tabelle 1 und Tabelle 2 visualisiert. Dabei verzichtet die Autorin auf ergänzende Adjektive wie *«fehlende Akzeptanz»* oder *«mangelndes Verständnis»*, da in der Tabelle die Ausprägungen aufgelistet werden, die als hindernd hervorgegangen sind, wird implizit angenommen, dass eine Absenz der Ausprägung oder eine unzureichende Anwendung für das Erscheinen in der Literatur verantwortlich sind. Bei Ausprägungen,

bei denen diese Handhabung unangemessen ist, wird eine Ergänzung in kursiv eingefügt. Diese soll den entsprechenden Begriff präzisieren.

Kategorie (Praktik dt.)	Kategorie (Praktik eng.)	Ausprägung (Gründe)
Planungsspiel	Sprint/Iteration Planning	Anforderungsmanagement Disziplin Moderation Organisation
Kurze Releasezyklen	Short Iterations	Kompetenzen
Metapher	Metaphor	Erfahrung Kultur *bestehende* Unklarheit
Einfaches Design	Emergent Design	Anleitungen Dokumentationen Priorisierung Zeit
Testen	Test Driven Development	*erhöhter* Aufwand Kompetenzen *erhöhte* Kosten
Refactoring	Refactoring	*bestehende* Abhängigkeit (Architekturdesign) *bestehende* Abhängigkeit (Unit-Tests) Akzeptanz Anleitungen Durchhaltevermögen Kompetenzen *vorhandener* Legacy Code
Paarprogrammierung	Pair Programming	Infrastruktur Kompetenzen *erhöhte* Kosten Kultur Managementakzeptanz Produktivität Ressourcen *mögliche* Unterbrechung Verfügbarkeit *örtlich* verteilte Teams *beeinträchtigtes* Wohlbefinden
Gemeinsame Verantwortlichkeit	Collective Code Ownership	Mindset Verständnis

Fortlaufende Integration	Continuous Integration	*erhöhter* Aufwand
40-Stunden Woche	Sustainable Pace	Organisation *unpassende* Projektart und -grösse
Kunde vor Ort	On Site Customer	*bestehende* Abhängigkeit Anwesenheit Beteiligung Domänenwissen *nicht repräsentative* Interpretation *gescheute* Verantwortung Verfügbarkeit
Programmierstandards	Coding Standards	Akzeptanz *festhalten an* Dogma Erfahrung Organisation

Tabelle 1: Gründe aus der Literatur für die geringe Verbreitung einzelner XP-Praktiken

Aus der Tabelle kann entnommen werden, dass viele diverse Schwierigkeiten ermittelt werden konnten. Die Praktiken Paarprogrammierung, Kunde vor Ort und Refactoring resultierten in der Literatur als die am häufigsten diskutierten. Daher konnten für diese Praktiken am meisten Gründe erhoben werden. Auch geht aus der Literatur hervor, dass einige Aspekte über mehrere Praktiken hinweg wiederkehrend sind. So können unzureichende Kompetenzen, sowohl auf der sozialen als auch fachlichen Ebene, als hindernd für einen erfolgreichen Einsatz von Extreme Programming sein. Für die Praktiken Refactoring, Paarprogrammierung und Programmierstandards wird zudem jeweils eine fehlende Akzeptanz gegenüber der Praktik erwähnt. Dabei kann diese Akzeptanz von unterschiedlicher Seite herkommen. Während diese bei der Paarprogrammierung vorwiegend managementseitig ausbleibt, fehlt sie bei dem Refactoring und den Programmierstandards auch entwicklungsseitig.

Die folgende Tabelle visualisiert die Aspekte, welche praktik-übergreifend zugeordnet werden konnten.

Kategorie	Ausprägung (Gründe)
XP-Methode	*bestehende* Abhängigkeit *der Praktiken* Akzeptanz Dokumentation Durchhaltevermögen *verlagerte* Entscheidungsgewalt *vorausgesetztes* implizites Wissen Interpretation Kommunikation Kompetenzen Managementpraktiken Mindset Motivation Produktivität *bestehende* Projektkontrolle *unzureichende* Skalierbarkeit *ungewünschte* Transparenz *örtlich* verteilte Teams *negative Auswirkungen auf die* Wirtschaftlichkeit
Hybride Vorgehensmodelle	Adaption *des Ursprungsmodells* *bestehende* Einschränkungen Ergänzung *(Scrum/XP)* *ständige* Evolution *bestehende* Flexibilität *hybride Vorgehen als* Mainstream *beeinträchtigt durch* Neophobie

Tabelle 2: Gründe aus der Literatur für die geringe Verbreitung von XP

Wie die Tabelle zeigt, führen diverse Aspekte dazu, dass XP als Methode nicht erfolgreich zum Einsatz kommt. Wiederkehrend ist auch hier eine fehlende Akzeptanz über die Praktiken hinweg festzustellen. Gekoppelt mit der Entscheidungsgewalt, welche, wie aus der Literatur hervorgeht, nicht immer beim Team liegt, kann bei mangelnder Akzeptanz seitens Managements der Einsatz der Praktik verweigert werden. Dass XP gemäss der Literatur ein entwicklungszentrierter Ansatz ist, welcher dazu tendiert, Managementpraktiken zu benachteiligen, kann erklären, weshalb in der Praxis vermehrt hybride Vorgehensmodelle zum Einsatz kommen. Demnach lassen sich durch eine Adaption Scrum und XP kombinieren.

Letztendlich lässt sich die Teilfrage 2 durch die Gründe aus Tabelle 1 und Tabelle 2 ausführlich beantworten.

5 Gründe aus der Praxis für die geringe Verbreitung von XP

Dieses Kapitel hat zum Ziel, mittels Interviews die Gründe zu eruieren, weshalb lediglich einzelne Praktiken von XP noch angewendet werden. Wie Beck (2004) in seinem Buch schreibt, muss man mit Coaches sprechen, um Beispiele von XP-Durchführungen und Programmiergeschichten zu erfahren (S. 16). Die Autorin verfolgt in diesem Kapitel das Ziel, durch Befragungen genau solche Einblicke zu erlangen, die eine Beantwortung der Teilfrage 3 ermöglichen.

5.1 Konzeption des Interviewleitfadens

Als Erhebungsinstrument wird ein Interviewleitfaden gewählt. Dabei behält sich die Autorin eine situative Umstrukturierung des Interviewleitfadens vor, um den Gesprächsfluss aufrecht zu erhalten und die Fragen an der passenden Stelle zu stellen.

Der Interviewleitfaden wurde in drei Teile strukturiert. Im ersten werden Einstiegsfragen gestellt, die zur Klärung der richtigen Ansprechperson dienen sollen. Der zweite Teil enthält die Schlüsselfragen. Dieser Teil ist der Hauptteil des Leitfadens. Die Autorin hat den Interviewleitfaden absichtlich so konzipiert, dass die Fragen ein breites Spektrum an Themen abdecken. Es wurde gezielt darauf verzichtet, sich lediglich auf einzelne Aspekte aus der Literatur zu fokussieren, da der Forschungsgegenstand in seiner Breite erfasst werden soll. Der Interviewleitfaden schliesst mit einer Frage ab, welche die Thematik betreffende Aspekte, die nicht bereits thematisiert wurden, abfangen soll. Gesamt sind im Leitfaden 15 Fragen enthalten. Der konzipierte Leitfaden kann dem Anhang entnommen werden.

5.2 Auswahl der interviewten Personen

Gläser und Laudel (2010) bezeichnen in ihrem Lehrbuch eine Interviewperson für Experteninterviews, als eine Person, deren Kenntnisse für die Untersuchung von Bedeutung ist (S. 43). Im Kontext dieser Arbeit sieht die Autorin das spezifische Wissen von Agile Coaches über agile Methodologien als relevant. Ausgewählt wurden Agile Coaches, weil sie als Coach Einblicke in Organisationen und diverse agile Projektvorgehen erhalten und ein fundiertes Methodenwissen besitzen. Als Coach berät man in der Regel mehrere Teams gleichzeitig und kann so in kürzerer Zeit unterschiedliche Erfahrungen sammeln. Durch die Befragung von Agile Coaches erhofft sich die Autorin einen tieferen und vielfältigeren Einblick in die Projektvorgehen diverser Unternehmen trotz geringer Stichprobe.

Um die Gefahr eines Bias zu umgehen, wie er von Gläser und Laudel (2010, S. 118) beschrieben wird, hat die Autorin bewusst nicht Bekannte befragt. Um jedoch einen Zugang zu den interviewten Personen gewährleisten zu können, wurden Bekannte, die als Agile Coaches tätig sind, angefragt, ob sie Drittpersonen empfehlen könnten, die für die Untersuchung relevantes Wissen besitzen. Da die Verfügbarkeit der Interviewpersonen auch von ihrer Arbeitsbelastung abhängen kann (Gläser & Laudel, 2010, S. 117), hat die Autorin bereits früh Kontakt aufgenommen und Termine vereinbart. Die Anzahl der erforderlichen Interviewpersonen wurde initial auf sechs festgelegt, dabei standen nicht alle bereits zu Beginn der Erhebung fest. Nach Gläser und Laudel (2010) ist dies auch nicht notwendig, da es durchaus sein kann, dass man nach einem Interview auf neue Gesprächspersonen aufmerksam gemacht wird (S. 118). Die Tabelle 3 zeigt die Merkmale der Interviewpersonen. Während den Interviews hat sich abgezeichnet, dass Aussagen repetitiv waren und keine neuen Erkenntnisse mehr gewonnen werden können. Deswegen hat die Autorin entschieden, die Untersuchung nach fünf Interviews zu beenden und die gesammelten Informationen auszuwerten.

Die designierten Interviewpersonen werden nach definierten Kriterien ausgewählt, diese werden folgend näher erläutert. Erstes Kriterium sind die Kenntnisse agiler Methodologien. Als zweites Kriterium hat die Autorin definiert, dass die Person mindestens fünf Jahre Erfahrung als Coach besitzen muss, die Anzahl Jahre wurde dabei durch die Autorin festgelegt in der Annahme, dass in dieser Zeitspanne genügend Erfahrungen gesammelt werden konnten. Nebst der Coaching-Erfahrung werden auch Kenntnisse in der Softwareentwicklung vorausgesetzt. Da Extreme Programming entwicklungszentriert ist, steigert dieses Kriterium Beobachtungen und Erfahrungen mit der Methode oder einzelnen Praktiken. Die interviewten Personen müssen Berufserfahrung in der Schweiz haben, weitere Erfahrung im Ausland wird nicht ausgeschlossen, dennoch geht es primär darum, Erkenntnisse für die Schweiz zu erhalten. Zur vorgängigen Prüfung, ob die ausgewählten Personen den Kriterien entsprechen, hat die Autorin einen Background-Check via LinkedIn durchgeführt. Wie die Tabelle 3 zeigt, erfüllen die Interviewpersonen die definierten Auswahlkriterien. Als Teil der Anonymisierung, wie sie im Kapitel 5.4 beschrieben wird, werden die Interviewpersonen nachfolgend nicht in der Reihenfolge der Interviewdurchführung aufgelistet.

IP[3]	Erfahrung im Coaching	Erfahrung in der Softwareentwicklung	Erfahrung im Ausland	rekrutiert über	Interview-dauer
A	13 Jahre	20 Jahre	Nein	IP	0:20:17
B	9 Jahre	35 Jahre	Nein	Bekannte	0:28:15
C	25 Jahre	35 Jahre	Ja	Bekannte	0:32:44
D	18 Jahre	20 Jahre	Ja	IP	0:25:03
E	26 Jahre	34 Jahre	Ja	IP	0:57:10

Tabelle 3: Auswahlkriterien und Angaben der einzelnen Interviewpersonen

5.3 Durchführung der Interviews

Vier Interviews wurden virtuell durchgeführt und ein Interview fand physisch statt. Als Tool für die virtuelle Durchführung wurde Cisco Webex verwendet. Dieses bietet die Möglichkeit, Videokonferenzen durchzuführen und aufzuzeichnen. Vor dem Interview wird von den interviewten Personen das mündliche Einverständnis eingeholt, damit das Interview aufgezeichnet werden kann. Eine Aufzeichnung ist notwendig, damit die Autorin im Anschluss das Gesprochene transkribieren und auswerten kann. Das Interview wurde sowohl in Webex als auch mit dem Smartphone aufgezeichnet. Damit das Interview einem natürlichen Gespräch möglichst nahekommt, erfolgt die Durchführung in Mundart, also Schweizerdeutsch. Der Autorin ist bewusst, dass dies ein Mehraufwand für die Transkription bedeutet, da das Gesagte später in Schriftdeutsch festgehalten wird. Dieser Aufwand wird jedoch in Kauf genommen, um den Redefluss besser aufrecht erhalten zu können. Die Interviews dauerten zwischen 20 und 57 Minuten. Die exakte Dauer wurde bereits in Tabelle 3 antizipiert.

5.4 Transkription und Auswertung

Die Transkription der Interviews findet jeweils direkt im Anschluss an das Gespräch statt. Dafür wird die Audiodatei vom Smartphone in MAXQDA20 importiert. Das Tool bietet die Möglichkeit, die Wiedergabegeschwindigkeit des Gesprächs zu verringern und ermöglicht somit eine effizientere Transkription. Wie von Gläser und Laudel (2010) empfohlen, wurdedas Interview möglichst vollständig transkribiert (S. 193). Als Regeln für die Transkription hat die Autorin festgelegt, dass unverständliche Abschnitte mit (…) gekenn-

[3] Abkürzung für interviewte Person

zeichnet werden. Pausen werden, sofern sie für die Aussage als relevant erachtet werden, abhängig von ihrer Länge mit (), (-) oder (--) dokumentiert, wobei zwei Bindestriche bedeuten, dass die Pause am längsten ist. Dabei folgt die Autorin der Empfehlung von Gläser und Laudel (2010), die besagt, dass es lediglich Sinn macht Geräusche, Pausen und sogenannte paraverbale Äusserungen wie «hm» oder «äh» festzuhalten, wenn sie für die Untersuchung relevant sind (S. 193). Weiter werden während der Transkription, wo für den Lesefluss notwendig, durch die Autorin Ergänzungen zum Kontext in [ ] eingefügt. Obschon keine der Personen ausdrücklich darauf bestanden hat, werden die Transkripte anonymisiert. Die Transkripte können dem Anhang entnommen werden.

Im Anschluss werden diese analog dem in Kapitel 3.3 beschriebenen Vorgehen in MAXQDA20 einer qualitativen Inhaltsanalyse unterzogen. Dabei arbeitet die Autorin mit dem bestehenden Kodierleitfaden und erweitert das Kategoriensystem entsprechend mit neuen Erkenntnissen. Die Herausforderung bei der Auswertung wird es sein, sicherzustellen, dass die Äusserungen der Befragten intersubjektiv nachvollziehbar sind und objektive Schlussfolgerungen gezogen werden (Döring & Bortz, 2016, S. 358).

5.5 Ergebnisse der Leitfadeninterviews

In diesem Kapitel werden die Ergebnisse aus den Interviews präsentiert. Die Unterkapitel teilen sich in die vordefinierten Kategorien ein. Zusätzlich ist aus dem Interview noch eine weitere Kategorie entstanden, diese enthält Erkenntnisse, welche spezifisch für die Schweiz ermittelt wurden. Damit für die Lesenden deutlich wird, was als Grund ermittelt wurde, werden die Gründe im Text fett hervorgehoben. Zusätzlich werden die Ergebnisse durch Zitate aus den Interviews belegt. Diese Zitate sollen den Sachverhalt verständlicher machen und es den Lesenden ermöglichen, nachzuvollziehen, wie die Autorin zur entsprechenden Schlussfolgerung gelangt ist (Gläser & Laudel, 2010, S. 273–274). Damit diese Zitate für die Lesenden schnell sichtbar sind, werden sie unabhängig ihrer Länge eingerückt, in Anführungszeichen gesetzt und kursiv formatiert.

5.5.1 Ergebnisse einzelne Praktiken

Dieses Kapitel beinhaltet die Befunde zu den einzelnen Praktiken. Für einen besseren Lesefluss wurde entschieden, hier nicht mit Unterkapiteln, sondern mit Titeln zu arbeiten. In der Arbeit werden keine Überschriftsebenen tiefer als drei (X.X.X) verwendet. Die Praktiken Planungsspiel, kurze Releasezyklen, Refactoring, Fortlaufende Integration und

Programmierstandards werden in diesem Kapitel nicht erwähnt, da diese in den Interviews nicht zur Sprache kamen und somit auch keine Erkenntnisse erlangt wurden.

Ergebnisse Kategorie Metapher

Die Befunde aus der Literatur, dass die Metapher schwer verständlich ist, wurden durch das durchgeführte Interview mit der befragten Person E verhärtet.

> *«Das Problem ist, dass die meisten Leute den Begriff Metapher nicht verstehen.»* (inter- viewte Person E)

Die **Unklarheit,** welche somit gemäss der befragten Person E damit behaftet ist, wird als Grund für die fehlende Präsenz in der Praxis genannt. Obschon laut der interviewten Person E der Grundgedanke der Praktik war, dass durch eine Formulierung der Metapher die **Resonanz** erhöht und auch für weniger IT-affine Personen Verständlichkeit geschaffen wird.

Ergebnisse Kategorie Einfaches Design

Ein erfolgreicher Einsatz der Praktik des Einfachen Designs hängt gemäss der befragten Person B von der **Unternehmensgrösse** ab:

> «*Es gibt zum Beispiel unbequeme Elemente [in XP], die man nicht einfach so in einem Grossunternehmen anwenden kann. Zum Beispiel ‹Simplicity›, also Einfachheit. Das erlaubt einem jetzt niemand, dass man quasi sagt ‹he mach doch mal etwas, wo gar keine Datenbank dahinter ist, einfach weil es das Minimum ist, das du bauen könntest›.*» (interviewte Person B)

Wie die Person weiter ausführt, ist die Möglichkeit diese Praktik anzuwenden, beispielsweise in einem Startup durchaus gegeben. Anders eben in Grossunternehmen, in welchen das Vorhaben meist planbasiert durchgeführt wird.

Ergebnisse Kategorie Testen

Sowohl die interviewte Person B als auch die interviewte Person D sind der Meinung, dass die testgetriebene Arbeitsweise es nicht in den **Mainstream** geschafft hat:

> *«Also die technischen Praktiken, zum Beispiel testgetrieben zu arbeiten, das ist in der heutigen Zeit nach wie vor nicht gegeben.»* (interviewte Person B)

> *«Dinge wie Pair Programming und Test-Driven Development sind sicher nicht im Mainstream angekommen.»* (interviewte Person D)

Die Gründe dafür können unterschiedlicher Natur sein. Im Zusammenhang mit dem Einsatz der Praktik macht die interviewte Person B die folgende Aussage:

> *«Die Leute sind faul beziehungsweise Entwickler codieren gerne mal schnell etwas.»* (interviewte Person B)

Demnach wird der **Aufwand**, welcher die Praktik mit sich bringt, gescheut. Des Weiteren resultiert das richtige **Mindset** als notwendiger Faktor. Laut der interviewten Person E ist eine testgetriebene Entwicklung jedoch eher unbeliebt, denn:

> *«Streng genommen heisst testgetriebene Entwicklung, dass du keinen einzigen Buchstaben Code schreibst, wenn du keinen Test hast, der nicht läuft. Also das erste, was du machen musst, ist, einen Test zu schreiben, der dir sagt ‹du bist ein [zensiert], du kannst das nicht›,weil der Test noch nicht läuft. Stell dir vor, was das mit dem Ego eines Softwareentwicklers macht.»* (interviewte Person E)

Die interviewte Person B begründet das fehlende Mindset mit der Aussage, dass eine solches Vorgehen nicht in der Natur eines Entwickelnden liege. Wie die interviewte Person B weiter sagt, spielt auch die **Ausbildung** eine Rolle, denn eine testgetriebene Entwicklung, werde nach wie vor stiefmütterlich behandelt.

> *«Es wird nicht ermuntert, dass man erst einen Test schreibt und danach den Code dazu.»* (interviewte Person B)

Wie die interviewte Person E mitteilt, wurden in Projekten, in welchen sie gearbeitet hatte, neue Teammitglieder zu Beginn ausschliesslich mit dem Testing vertraut gemacht.

Ein weiterer Faktor, welcher als hindernd ermittelt wurde, war der Umgang mit **Legacy Code**.

> *«Viele Teams haben auch das Problem, dass sie eine bestehende Codebasis haben, bei welcher keine bestehenden oder nur wenige Tests vorhanden sind. Das dann noch testbar machen, dass man überhaupt Tests schreiben kann, ist natürlich das Schwierigste, was es überhaupt gibt.»* (interviewte Person D)

Die interviewte Person D führt weiter aus, dass es ideal ist, auf grüner Wiese starten zu können, da man sich so nicht mit einer bestehenden Codebasis auseinandersetzen muss. Der **Zeitpunkt** spielt, wie aus einem Interview hervorgeht, auch eine Rolle:

> *«Viele fangen mit automatisierten Tests zu spät an, machen es zu wenig, machen es zu wenigdiszipliniert. Und weil sie es nicht schaffen, eine Grundlage aufzubauen, bricht auch der ganze Rest zusammen.»* (interviewte Person B)

Letzter Faktor, welcher ermittelt wurde, war die **Akzeptanz**. So äussert die interviewte Person E, dass in der Einführung von XP der Widerstand der Entwickelnden gegen Pair Programming und testgetriebene Entwicklung als hindernd empfunden wurde.

Ergebnisse Kategorie Programmieren in Paaren

Die interviewte Person E hat im Zusammenhang mit Herausforderungen bei der Einführung von XP die folgende Aussage betreffend Paarprogrammierung geäussert:

> *«In der Einführung von XP war es vor allem der Widerstand der Entwickler, gerade gegen Pair Programming und testgetriebene Entwicklung.»* (interviewte Person E)

Demnach trifft nebst der testgetriebenen Entwicklung auch die Paarprogrammierung auf fehlende **Akzeptanz**. Dabei ist laut der Person D die Akzeptanz nicht einseitig.

> *«Es gibt immer noch sehr viele Entwickler, aber vor allem auch Manager, die sagen ‹was, zwei Leute vor einem PC? Das ist ja Zeitverschwendung!›»* (interviewte Person D)

Diese Auffassung von Pair Programming führt zu einer mangelnden **Managementakzeptanz** der Praktik. Wie die Person D erläutert, ist die Praktik umstritten, deswegen hat sie es ihrer Meinung nach auch nicht in den **Mainstream** geschafft.

> *«Pair Programming ist immer noch ein heisser Diskussionspunkt in der Schweiz, aber es sind nur Meinungen, es gibt niemanden, der eine empirische Studie gemacht hat, auch weltweit gibt es wenige empirische Studien. Alle empirischen Studien, die gemacht worden sind, sind nicht schlüssig. Sie sagen, es scheint etwas zu bringen, worauf die Diskussion Pair Programming kostet mehr, sich schon erledigt hat.»* (interviewte Person C)

Der Aussage kann entnommen werden, dass der Aspekt der vermeintlich damit verbundenen **Kosten** durch die interviewte Person C als hindernder Faktor für die Anwendung der Praktik angesehen wird. Die befragte Person E führt ausserdem den Nutzen der Praktik aus:

> *«Was ich gelernt habe; Pair Programming mit Testern bringt dir als Entwickler bei, wie man testbaren Code schreibt. Pair Programming mit einem Product Owner oder mit einem Kunden bringt dir bei, wie man lesbaren Code schreibt. Und das*

> *steht sonst kaum irgendwo geschrieben. Man muss Pair Programming nicht nur mit anderen Entwicklern machen.»* (interviewte Person E)

Laut ihr ist die **Paarbildung** bei dieser Praktik nicht auf die Rolle der Entwickelnden beschränkt. Als weitere Schwierigkeiten der Praktik nennt sie Aspekte der **Logistik** und **Infrastruktur**:

> *«In der Weiterführung von XP gab es eher logistische Schwierigkeiten. Wie beispielsweiseder Programmiersprache, der Netzwerklatenzzeit oder der Aufstellung der Tische, damit sie für Pair Programming geeignet sind.»* (interviewte Person E)

> *«Also wenn du die früheren Bücher liest, siehst du, dass die Firmen, die es gemacht haben, extra Tische bauen liessen. Und tatsächlich gerade am Anfang zeigt das, dass die ganze Infrastruktur nicht für XP ausgelegt worden ist.»* (interviewte Person E)

Hinzu kommt laut Person E, dass es durchaus möglich ist, dass diese Praktik das **Wohlbefinden** der betroffenen Personen beeinträchtigen kann.

> *«Es besteht das Problem des Beobachtet-Werdens.»* (interviewte Person E)

Bei unterschiedlichen **Kompetenzen** der Personen, die für die Praktik zusammensitzen, kann es zusätzlich zu Schwierigkeiten kommen.

> *«Pair Programming war unter Leuten mit demselben Wissenstand eher noch machbar, aber wenn du mal einen Experten hast und ein Anfänger, naja – aber das muss man tatsächlich unter Wissenstransfer abtun.»* (interviewte Person E)

Dieselbe Person nennt weiter, dass für eine erfolgreiche Anwendung der Praktik eine gewisse **Disziplin** erforderlich ist.

Ergebnisse Kategorie Gemeinsame Verantwortlichkeit

Zur Praktik der gemeinsamen Verantwortlichkeit resultierte aus dem Interview mit der Person A, dass diese ein entsprechendes **Mindset** voraussetzt.

> *«Collective Code Ownership ist eine Einstellungssache, die kann man nicht frontal ausbilden, das macht man einfach, das lernt man.»* (interviewte Person A)

Demnach liegt es an den Entwickelnden, diesbezüglich die Initiative zu ergreifen und eine gewisse Lernbereitschaft aufzuweisen. Wie die Person C äussert, macht sie regel-

mässig Code-Reviews bei Kundinnen und Kunden und empfindet dabei die **Qualität** des Codes selten als berauschend.

Ergebnisse Kategorie 40-Stunden Woche

Die Praktik der 40-Stunden Woche, auch als Sustainable Development oder Sustainable Pace bekannt, stellt gemäss der Person C ein Problem in der Schweiz dar. Dies begründet sie dadurch, dass die Praktik eine Spannung mit dem Management aufweist. Demnach sei es aus Sicht des Managements rentabler, das Entwicklungsteam Überstunden machen zu lassen und diese anschliessend auszuzahlen, als sich an eine nachhaltige Arbeitszeit zu halten. Demzufolge ist **Wirtschaftlichkeit** ein hindernder Faktor dieser Praktik.

Ergebnisse Kategorie Kunde vor Ort

Gemäss den Personen A, B und D ist im Zusammenhang mit dieser Praktik die **KundInnennähe** ein entscheidender Faktor. Dieser ist laut der Person B in die Ferne gerückt.

> *«Der enge Kundenkontakt zwischen Entwickler und Endkunde, also dem Benutzer sozusagen, und auch dem Sponsor vom Projekt der ist wieder verloren gegangen.»* (interviewte Person B)

Wie die Person D erläutert, ist es schwierig, die Nähe zur Kundin und zum Kunden zu behalten. Sie sagt, dass es auf das Organisationssetup ankommt.

> *«Je grösser die Firma, desto schlechter geht es. Also, desto weiter weg sind die Entwickler oder das Team von dem Benutzer. Bei kleineren Teams ist es häufig noch etwas besser, aber das ist – was ich persönlich finde – in der Schweiz grundsätzlich eher schlecht.»* (interviewtePerson D)

Die befragte Person A erzählt dabei, dass sie stets das Ziel verfolgt, möglichst direkt und häufig mit dem Endnutzenden in Kontakt zu treten. Eine Validierung der Ergebnisse erfolge somit ad hoc und schaffe einen Mehrwert ohne Zeitverluste.

5.5.2 Ergebnisse hybride Vorgehensmodelle

Aus den Interviews resultierte, dass XP in hybrider Form in der Praxis angewendet wird. Dabei äussert die Person A, dass ein Vorgehen häufig mit Scrum startet und XP hinzu-

genommen wird. Die **Adaption** von XP geschieht dabei situativ. Einer solchen Adaption stimmt auch die interviewte Person B zu:

> *«Ich denke, am Schluss spielt es nicht so eine Rolle, ob es XP oder Scrum oder so ist. Ich glaube, es geht mehr darum, welches Problem man lösen möchte und welche Praktik das Problem lösen kann.»* (interviewte Person B)

Auch die Person D ist der Meinung, dass ein Entscheid für den Einsatz der Praktik situativ gefällt werden sollte. Diese Adaption sollte jedoch auf einem fundierten Entscheid basieren.Wie die Person C erzählt, scheint dies nicht immer der Fall zu sein:

> *«Das grobe Problem mit XP ist, XP ist technisch orientiert und recht diszipliniert und viele Leute sind geflüchtet in Scrum, was eigentlich auch diszipliniert ist, aber deshalb machen alle Leute Scrum, aber ‹meine Firma hat's einfach angepasst› – was sie angepasst haben ist eigentlich das, was für sie störend ist, weg zu nehmen. Es ist nicht eine Anpassung im Sinne von ‹wir haben eine Hypothese gestellt, wenn wir es anders machen, wird's besser›, sondern eher ‹was immer stört, das nehmen wir weg›, was eigentlich der Tod vom Sinn von XP, Agil oder Lean ist.»* (interviewte Person C)

Die Person B ist der Meinung, dass zumindest zu Beginn das ganze Set der XP-Praktiken eingesetzt werden sollte. In einem zweiten Schritt können anschliessend Anpassungen gemacht werden, um Veränderungen gegenüber offen zu sein. Dabei betont sie:

> *«Ich denke, was wichtig ist, ist, dass man nicht von den technischen Empfehlungen, also die technischen Praktiken, wie man als Entwickler vorgeht, abweichen sollte.»* (interviewte Person B)

Veränderungen gegenüber flexibel zu reagieren ist ein wichtiger Bestandteil einer agilen Vorgehensweise. Die Person E erwähnt in diesem Zusammenhang den «apply, inspect and adapt»-Zyklus. Sie erläutert, dass wenn man nach agile-by-the-book arbeitet, nicht agil ist. Aus ihrer Sicht ist es entscheidend, auch auf Prozessebene agil zu sein, deshalb empfiehlt sie by-the-book zu beginnen und mittels dem «apply, inspect and adapt»-Zyklus zu reflektieren, was funktioniert und zielführend ist.

Fehlen bei einer Umstellung von einer klassischen zu einer agilen Vorgehensweise die Kenntnisse, so kann eine Adaption zu einem vermeintlich agilen Vorgehen führen.

> *«Ich sage mal, die guten Teams haben XP-Bestandteile mitgenommen und die Teams, die es noch nicht kannten, die haben einfach Scrum als Hülle genommen und dahinter einfach weitergearbeitet, wie sie vorher gearbeitet haben. So entsteht quasi ein Mini-Wasserfall [Vorgehen].»* (interviewte Person D)

Aus den Interviews geht hervor, dass sich **Scrum und XP** in der Praxis gut ergänzen. Eine solche **Ergänzung** funktioniert gemäss der Person C sehr gut, da das Scrum Framework XP affin ist. Wie die Person D äussert, haben Scrum und XP einen relativ grossen Overlap und Scrum sei gar auf die Engineering Practices von XP angewiesen:

> *«Also ich glaube, Scrum funktioniert nicht ohne XP. Also ohne einen gewissen Teil aus XP zumindest. Primär [kann Scrum] nicht ohne die Engineering Practices funktionieren. Sonst wäre Scrum einfach eine leere Hülle. Scrum kann für mich eigentlich fast nur mit den XP Practices wirklich gut funktionieren.»* (interviewte Person D)

Auch die Person A weist darauf hin, dass sie aus XP die Software Craftmanship Aspekte nutzt und Scrum und Kanban für Businessprozessthemen verwendet. Wie die Person B erwähnt, äussert sich Scrum nicht zu technischen Praktiken. Die Person E geht sogar einen Schritt weiter und zitiert Steve Freeman wie folgt:

> *«Steve Freeman hat mal gesagt, Scrum ist die Rache von XP. Wenn du Scrum richtig machst, merkst du, dass du unbedingt XP brauchst, wenn du an einem Softwareentwicklungsprojekt arbeitest.»* (interviewte Person E)

Aus der Sicht der Person E wird XP benötigt, um Scrum ein schnelleres Feedback im Sinne des Produktes und im Sinne der Akzeptanz zu geben. Sie führt weiter aus, dass es in den **Mainstream** lediglich einzelne Praktiken geschafft haben. Die Person E visualisiert ihre Aussage mit einer adaptierten Moore's Curve. Dabei plädiert sie darauf, dass XP in der Kurve bei den Innovatorinnen und Innovatoren liegt und es nie aus der Nische geschafft hat. Die Begründung dafür liefert sie wie folgt:

> *«XP ist schwer, es ist verdammt schwer.»* (interviewte Person E)

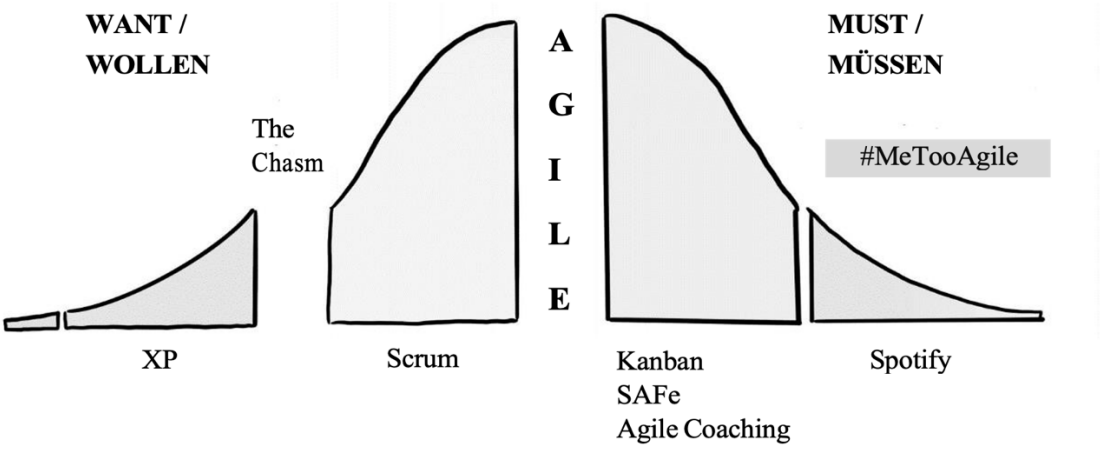

Abbildung 5: Darstellung der Moore's Curve von Pelrine, 2020 (in Anlehnung an Moore, 1995; Pietri, 2011)

Die Person E erklärt ihre Visualisierung mit der folgenden Aussage:

> *«Moore hat gesagt, dass es bei disruptiven Technologien diesen Chasm gibt. Nur die, die im Mainstream [rechts vom Chasm] arbeiten, sind überlebensfähig. Will Pietri, ein alter agiler Knacker, hat mal gesagt, dass es bei agile auch noch einen [Chasm] hier [in der Mitte] gibt und zwar zwischen Firmen, die agile machen, weil sie es wollen und Firmen, die es machen, weil sie müssen. Und wir sind jetzt ganz deutlich hier [rechts vom mittleren Chasm] also bei Firmen, die es tun, weil sie es müssen und es eigentlich gar nicht wollen. Was sie wollen ist – ich nenne es #MeTooAgile – sie wollen sagen, dass sie es tun, obwohl sie eigentlich möglichst wenig verändern möchten.»* (interviewte Person E)

Die **Änderungsbereitschaft** von Unternehmen im Mainstream ist gemäss Aussage der Person E relativ gering. Auch ist sie der Meinung, dass viele Unternehmen zwar versuchen agil zu sein, jedoch dabei die **Komfortzone** nicht verlassen möchten und deswegen der Ansatz scheitert.

Die Person A hat in ihrem Interview als weiterer hindernder Faktor für den Einsatz von XP den Aspekt der **Regulierungen** genannt. Sie erzählt, dass sie in einem regulierten Umfeld tätig ist und sich daher nicht jede XP-Praktik für den Einsatz eignet.

> *«Je nach Kontext funktionieren halt gewisse Dinge nicht. Ich arbeite in einem regulierten Umfeld und das heisst, ich muss meine Spezifikationen irgendwann freezen, ablegen und sagen ‹das ist jetzt das Produkt, jetzt kann ich es auf den Markt geben› und ab dem Punkt fällt XP beispielsweise auseinander. In XP würde man sagen ‹das gibt's, nicht, dass man Spezifikationen fixt, die sind immer anpassbar›. Aber der Kontext in der Medizin oder im regulierten Umfeld ist einfach so, dass man irgendwann die Spezifikation fixen muss, damit man das Gerät auf den Markt bringen kann, weil ich muss das irgendwann dem TÜV oder FDA übergeben und die können damit nicht umgehen, wenn ich sage ‹das ist ein wachsendes Ding, wir schauen schon für gute Qualität›.»* (interviewte Person A)

Die Person C spricht zusammenhängend mit Regulierungen darüber, dass in der Schweiz der Bund die HERMES-Methode den Kantonen verordnet hat. Bürki et al. (2020) beschreiben HERMES in ihrem Referenzhandbuch als «Projektmanagementmethode für Projekte im Bereich der Informatik, der Entwicklung von Dienstleistungen und Produkten sowie der Anpassung der Geschäftsorganisation.» (S. 5). Die Person C erläutert ihre Erfahrungen mit der Methode wie folgt:

> *«Die HERMES-Methode ist eine Krücke. Ich habe es noch knapp geschafft über HERMES() mit anderen Leuten Scrum einzuführen, aber es ist eine Water-Scrum-*

Fall [Methode]. Darin können Sie weder mit XP noch mit Scrum arbeiten und das ist die offizielle Methode.» (interviewte Person C)

5.5.3 Ergebnisse XP-Methode

Aus den Interviews resultierten diverse Aspekte, welche die Autorin der Methode selbst zugeordnet hat. Einer dieser Aspekte ist das **Marketing**. Laut Person C waren die Ideen von XP gut, aber Scrum war marketingtechnisch überlegen. Derselben Meinung ist auch die Person D:

«Scrum ist natürlich marketingtechnisch wesentlich stärker gewesen. XP hat eher Entwickler angesprochen. Scrum hat eher Manager angesprochen oder Projektleiter, Produktmanager, die sagten sich dann ‹ja, jetzt habe ich endlich etwas, wie ich alles koordinieren kann› und das ist natürlich wie eine Bombe eingeschlagen. Es klingt einfach, ist verständlich und es gab auch das Versprechen von Jeff Sutherland: ‹Das Doppelte in der Hälfte derZeit› und so etwas hören Manager natürlich wahnsinnig gern.» (interviewte Person D)

Ein wesentlicher Faktor im Zusammenhang mit dem Marketing sind gemäss Aussagen der Personen B und D die **Zertifizierungen**. Anders als Scrum bietet XP keine Zertifizierungen an und gemäss der Person B fehlt folgend die Lobby für XP.

«Die ganze Zertifiziererei hat Scrum einen riesen Boost gegeben, nun konnte man sagen, dass man zertifizierter Scrum Master ist. Das scheint in der Branche einfach unglaublich gut zu funktionieren und so hat Scrum eigentlich XP total verdrängt.» (interviewte PersonD)

Die Personen A, D und E nennen Faktoren, die eine fehlende **Akzeptanz** signalisieren. So erzählt Person E beispielsweise von einem Mandat, in welchem sie XP eingeführt hatte. Das auftraggebende Unternehmen empfand XP aus der Sicht der erzählenden Person nach der erfolgreichen Umsetzung als Bedrohung:

«Die hatten geschätzt, dass wir 6–9 Monate für eine Anwendung brauchen. Wir haben aber beide in vier Monaten geliefert. Fully refactored, alles lief sauber durch den Testbetrieb, wie geschmiert mit Olivenöl. Das hat dazu geführt, dass XP als Methode in der Bank verboten wurde. Und wir haben aufgedeckt; In der 40-jährigen Informatikgeschichte der Bank: das erste Projekt, welches on-time, on-budget, on-spec geliefert hat und das war für sie eine Bedrohung.» (interviewte Person E)

Auf die Frage hin, bei wem die **Entscheidungsgewalt** für ein Vorgehen liegt, führt sie weiter aus:

> *«Das Management entscheidet. Es werden politische Entscheide getroffen, die darauf abzielen ein möglichst geringes Risiko für das Management zu haben.»* (interviewte Person E)

Dass die Entscheidungsgewalt beim Management liegt, empfinden auch die Personen B, C und D. Lediglich die Person A äussert, dass solche Entscheidungen durch das Team gefällt werden. Wie die Person B erläutert, liegt der Schwerpunkt von XP stark auf den technischen Empfehlungen. Was daraus gemäss ihrer Aussage eine **entwicklungszentrierte Methode** macht. Sie führt ihre Erzählung fort und erklärt, dass die technischen Praktiken in anderen Methoden zu kurz kommen, weshalb dann auf die XP-Praktiken zurückgegriffen wird.

> *«Ich glaube, XP als Methode ist, würde ich mal behaupten – bis auf ein paar kleine ‹Orte› – tot beziehungsweise es haben nur noch die technischen Praktiken überlebt. Die Methode als Ganzes mit all den Dingen, die nicht direkt mit der Programmierung zu tun haben, überlebt im Geist, aber nicht als Methode.»* (interviewte Person B)

Dass die Methode **eine technische Orientierung** aufweist, bestätigen auch die Person C und E. Die Person E sieht XP als der technische Teil eines gesamt agilen Ansatzes:

> *«Also es geht darum zu sagen, was wir machen, warum wir es machen, wann wir es machen und wie wir es machen. Das Wie ist XP, das Wann ist Scrum. Und das Was und Warum sollten eigentlich gute Projektmanagement- und Marketingpraktiken sein, welche oft einfach aus dem Fenster geworfen werden.»* (interviewte Person E)

Ein weiterer hindernder Faktor für den Einsatz von XP kann die unzureichende **Dokumentation** sein. Wie die Person D erzählt, sind Dinge verloren gegangen, weil sie nicht beschrieben wurden. Die Person E, welche an der Entstehung von XP beteiligt war, bestätigt diese Aussage:

> *«Und die Tatsache ist, dass die meisten Dinge nirgendwo geschrieben stehen. Die meisten Dinge sind irgendwo spät abends in einer Beiz entstanden.»* (interviewte Person E)

Gemäss der Person A benötigen jedoch unerfahrene Softwareentwicklerinnen und Softwareentwickler unter Umständen eine Schritt-für-Schritt-Anleitung, eine solche bietet XP jedoch nicht. Wie sie weiter ausführt, kann man in XP nicht einfach eine Checkliste abar-

beiten, sondern benötigt ein entsprechendes Mass an **Domänenwissen**. Auch ist sie der Meinung, dass Personen, die frisch vom Studium kommen und wenig Berufserfahrung haben, sich aufgrund der mangelnden **Erfahrung** mit XP schwertun. Hinzu kommt laut der Person C eine gewisse **Unkenntnis** im Zusammenhang mit den verschiedenen agilen Ansätzen. Aus den Interviews geht hervor, dass es für einen erfolgreichen Einsatz gewisser **Kompetenzen** bedarf.

> *«XP ist zu schwer. XP verlangt Disziplin.»* (interviewte Person E)

Nebst der notwendigen **Disziplin** erfordert die Methode auch ein entsprechendes **Durchhaltevermögen**.

> *«Die grösste Schwierigkeit sehe ich in der mangelnden Zeit für Reflexion und dem mangelnden Durchhaltewillen, damit sich das etablieren könnte. Im Sinne von man probiert es mal, es funktioniert nicht wie erwartet beim ersten Mal und dann sagt man direkt ‹ah, das funktioniert nicht› und wirft es wieder weg.»* (interviewte Person A)

Die Haltung, welche die Person A erläutert, kann laut der Person B dazu führen, dass das gesamte Konstrukt von XP auseinanderfällt.

> *«Und weil sie es nicht schaffen, eine Grundlage aufzubauen, bricht auch der ganze Rest zusammen. Das zeigt, dass die Praktiken alle irgendwie zusammengehören und wenn man sie nicht durchgängig anwendet, dann bricht es zusammen und wenn man keinen Erfolg hat, dann hört man auch mit dem Rest auf. Das zieht einen runter, anstatt dass es aufbauend ist und dort braucht es ein relativ grosses Durchhaltevermögen.»* (interviewte Person B)

Wie die Person C anbringt, spielen auch Faktoren wie die **Kultur** und **Anpassungsfähigkeit** eine Rolle. Demnach soll in XP jede Person, die sich nicht anpassen kann oder möchte, das Team verlassen. Sie bezeichnet XP als Meritokratie, in welcher sie neue Ideen und Änderungen einbringen können, wenn sie gut sind.

> *«Und diese Meritokratie ist im Spannungsfeld mit der Kultur der Deutschschweizer. Wenn Sie in der Deutschschweiz jemanden in einem Meeting kritisieren, dann haben Sie einen lebenslangen Feind im schlimmsten Fall. Bei Meritokratie dürft ihr sagen ‹den Code, den ich da gesehen habe, der ist nicht umwerfend›. Man darf nicht die Leute kritisieren aber die Resultate schon. Der Druck, auf Meritokratie zu gehen, als hinterschwelligen Wert von XP, ist in unserer Kultur nicht speziell ausgeprägt. Aber das ist wieder eine subjektive Interpretation, warum XP sich nicht durchgesetzt hat und Scrum sich eher durchsetzt.»* (interviewte Person C)

Laut der Person E wird XP in seiner **Extremität** praktiziert. Eine solche Extremität kann der Person D zufolge den Bedarf an einem Coaching oder einer **Begleitung** erfordern. Wie die Person B erzählt, stellt auch die Beliebtheit einen Faktor dar, denn laut ihr sind gute Entwickelnde immer extrem gerne mit neuen Technologien unterwegs, scheuen jedoch gute Engineering Practices. Diese sind aber ein wesentlicher Bestandteil von XP. Sich die nötigen Kompetenzen anzueignen, liegt dabei aus Sicht der Person C in der Verantwortung der Entwickelnden selbst. Sie nennt eine entsprechende **Eigeninitiative** als Voraussetzung für den Einsatz der Methode:

> *«Es gibt eine implizite, teilweise explizite Erwartung, dass die Leute fähig sind und wenn es ihnen an etwas fehlt, dass sie es ziemlich zügig lernen.»* (interviewte Person C)

Aus der Aussage der Person B zeichnet sich zudem ab, dass eine entsprechende **Motivation** notwendig ist:

> *«Man sagt so ‹der gute Softwareengineer entsteht von alleine›, man muss es sich selbstgetrieben aneignen. Da muss man selbst Motivation zeigen.»* (interviewte Person B)

Wie die Person A erzählt, kann auch der **externe Druck** dazu führen, dass XP nicht wie angedacht angewendet wird. Laut ihr verhindern Businessdruck, Termindruck und Deadlines, dass gewisse Dinge noch im geplanten Ausmass gemacht werden können. Hinzu kommt, dass eine falsche **Priorisierung**, wie sie von Person B beschrieben wird, dazu führen kann, dass Entwickelnde an Dingen arbeiten, die allenfalls obsolet sind. Sie plädiert dabei auf die Eigeninitiative der Entwickelnden.

> *«‹Aber ist es wirklich das, was den Kunden weiterbringt?› Auch ein Software Engineer dürfte sich das mal von Zeit zu Zeit fragen. Und die besten Software Engineers, die ich kenne, die machen das. Die fragen dann auch den Kunden ‹Wofür brauchst du das genau? Was für ein Problem willst du damit lösen? Was bringt dir das?›»* (interviewte Person B)

Die Person C äussert weiter, dass aus ihrer Sicht XP den Zeitpunkt verpasst hat und historisch verschwinden wird. Wie sie erzählt, kam die Methode zu einem **Zeitpunkt**, an dem es zu früh war für die meisten Leute. Beifügend ist laut den Personen B und E auch die **Unternehmensgrösse** ein Faktor, der berücksichtigt werden muss.

> *«Ich glaube, in der Grossunternehmenswelt hat XP keine Chance und das macht halt auch einen grossen Anteil der Entwicklungsjobs, die wir so haben, aus.»* (interviewte Person B)

5.5.4 Ergebnisse spezifisch für die Schweiz

In diesem Kapitel werden Aspekte aus den Interviews erläutert, welche explizit über die Schweiz geäussert wurden. Dabei handelt es sich nebst XP spezifischen Aussagen auch um solche, die allgemein den IT-Fortschritt in der Schweiz betreffen und somit impliziten Einfluss auf die Anwendung von XP haben können.

Die Person C erzählt über die Steigerung der Komplexität der IT-Welt und erwähnt in diesem Zusammenhang, dass es umstritten ist, dass die IT-**Entwicklungsfähigkeiten** der Schweiz mit dem Rhythmus des IT-Fortschritts mithalten können. Dem fügt sie anschliessend bei:

> *«Und dann kann man nicht eine Kultur wie XP einführen, da muss man auch ehrlich sein, das ist zwei Generationen weiter.»* (interviewte Person C)

Hinzu kommt gemäss der Person E, dass die Schweiz einen gewissen **Konservatismus** pflegt, wenn es um neue Technologien geht.

> *«IT-technisch sind wir Schweizer so konservativ, wir warten bis etwas rauskommt und erfolgreich ist, und dann werfen wir einfach eine Unmenge Geld raus, um den technologischen Rückstand wieder einzuholen.»* (interviewte Person E)

Wie die Person C weitererzählt, gibt es gemäss ihr keine grossen IT-Projekte mehr in der Schweiz. Diese werden ins Ausland verlagert. Dass eine solche **Auslagerung** stattfindet, sei jedoch vielen Leuten nicht bewusst. Weiter nennt sie eine fehlende **Wahrnehmung** der Gesellschaft für Software als Kernelement als hindernder Faktor für technologische Fortschritte.

> *«Die Wahrnehmung in unserer Gesellschaft, in der Industrie und so, dass Software ein Kernelement geworden ist, ist noch nicht da. Wir haben in der Schweiz nicht mal eine IT-Software-Assoziation, es gibt drei oder vier, die sich streiten und versuchen, mal zu fusionieren,aber im Prinzip ist es nicht klar, wo IT, Softwareengineering, Software im Kanton oder in Bern angegliedert ist.»* (interviewte Person C)

Diese fehlende Wahrnehmung sieht sie als einen Grund, weshalb auch die **Ausbildung** der IT-Fachkräfte in der Schweiz hintanbleibt. Sie plädiert darauf, dass der Pfad, wie die Ausbildung gemacht wird, sei es Fachhochschule, Lehre oder in einer Firma, verbesserungsbedürftig ist. Auch ist sie der Meinung, dass eine Optimierung dieser Ausbildung langfristig geplant werden muss:

> *«Viele der Entwickler, die jetzt mit agil arbeiten, die sogenannten Late-Adapter, was 60–90% der Entwickler ausmacht, haben sehr oft keine Ahnung von dem und da sie sich nicht speziell weiterbilden, kommen sie wenig damit in Berührung. Ich hoffe, dass sich das dreht, aber das muss langfristig geplant werden, dass Fachhochschulen und Uni die Studenten schulen, dass neue [Studienabgänger] das kennen. Aber das ist eine Hoffnung.»* (interviewte Person C)

Wie die Personen B, C und E äussern, erlangen die meisten Fachkräfte in der Schweiz die notwendigen Skills, um agil arbeiten zu können, ‹on the job›. Hierbei könne sich jedoch die Qualität der internen Trainings unterscheiden.

> *«Dann [mittels] Training ‹on the job›. Und es ist sehr variabel, es gibt sehr gute Training ‹on the job›, wo ich sagen würde ‹perfekte Lösung› und der Rest ist dann eine Alibiübung, um sich die Kosten zu sparen.»* (interviewte Person C)

Gemäss der Person C spielt auch der Aspekt der **Reglementierung** eine Rolle. Sie bezeichnet die Schweiz als reglementorientiert und macht mit ihrer Aussage darauf aufmerksam, dass implizite Erwartungen an eine agile Arbeitsweise dadurch weniger wahrgenommen werden:

> *«Die Methode, die in Scrum empfohlen worden sind, die XP-Ansätze, sind in Scrum integriert. Es wurde auf die Seite geschoben ‹ja, ihr könnt es machen›. Und wenn man mit Experten redet, sogar Ken Schwaber und so steht dahinter, ‹ja, macht's›, aber es steht nicht mehr explizit drin. Worauf wir in unserem Land sehr reglementorientiert sind, dann rutscht es wieder raus. Worauf die Qualität der Entwicklungsteams eher abnimmt, aktuell nicht zuviel, aber es nimmt eher ab.»* (interviewte Person C)

5.6 Zusammenfassung der Erkenntnisse und Beantwortung der Teilfrage 3

Das Kapitel 5 hatte zum Ziel, die Teilfrage 3 zu beantworten, welche wie folgt lautet:

Welche Gründe werden durch Personen aus der Praxis für die geringe Anwendung der Methodik genannt?

Durch die Interviews konnten Beobachtungen aus der Praxis erhoben und dadurch neue Erkenntnisse gewonnen werden. Dafür wurde der bestehende Kodierleitfaden für die Datenerhebung verwendet. Das aus der Literaturanalyse resultierende Kategoriensystem konnte mit neuen Ausprägungen erweitert werden. Bei der Auswertung wurde eine neue Kategorie hinzugefügt, welcher Aspekte zugeordnet werden, die spezifisch für die

Schweiz geäussert wurden. Zu den Praktiken Planungsspiel, kurze Releasezyklen, Refactoring, Fortlaufende Integration und Programmierstandards konnten keine neuen Erkenntnisse erlangt werden, da diese im Gespräch nicht thematisiert wurden. Demnach werden sie in der Tabelle 4 nicht aufgeführt. Eine Verbindung der Erkenntnisse aus der Literatur und der Praxis wird im Kapitel 6 vorgenommen.

Durch die offenen Fragen im Leitfaden konnten gezielt mehr Gründe ermittelt werden, die einen geringen Einsatz der XP-Methode selbst erklären. Diese können der Tabelle 5 entnommen werden. Die vollständige Liste mit einer Visualisierung, welche Gründe aus welchem Datenmaterial identifiziert wurden, kann dem Anhang entnommen werden. Aus der Analyse resultiert, dass die Aspekte, welche durch die Recherche ermittelt wurden, vielfältig sind. Diese begründen eine unterschiedlich starke Anwendung der Methode und der einzelnen Praktiken. Da die Gründe aus den Interviews zahlreich sind, werden diese in Tabelle 4 und Tabelle 5 visualisiert. Dabei verzichtet die Autorin analog der Visualisierung für die Teilfrage 2 auf ergänzende Adjektive wie *«fehlende Akzeptanz»* oder *«mangelndes Verständnis»*. Bei Ausprägungen, bei denen diese Handhabung unangemessen ist, wird eine Ergänzung in kursiv eingefügt. Diese soll den entsprechenden Begriff präzisieren.

Kategorie (Praktik dt.)	Kategorie (Praktik eng.)	Ausprägung / Gründe
Metapher	Metaphor	*bestehende* Unklarheit *niedrige* Resonanz
Einfaches Design	Emergent Design	*ungeeignete* Unternehmensgrösse
Testen	Test Driven Development	Akzeptanz *erhöhter* Aufwand Ausbildung *vorhandener* Legacy Code *nicht im* Mainstream Mindset *falscher* Zeitpunkt
Paarprogrammierung	Pair Programming	Akzeptanz Disziplin Infrastruktur Kompetenzen *erhöhte* Kosten Logistik *nicht im* Mainstream Managementakzeptanz *unklare* Paarbildung *beeinträchtigtes* Wohlbefinden
Gemeinsame Verantwortlichkeit	Collective CodeOwnership	Mindset Qualität
40-Stunden Woche	Sustainable Pace	Wirtschaftlichkeit *nicht gegeben*
Kunde vor Ort	On Site Customer	KundInnennähe

Tabelle 4: Gründe aus den Interviews für die geringe Verbreitung einzelner XP-Praktiken

Aus den Interviews geht hervor, dass sowohl XP als Methode als auch diverse einzelne Praktiken es nicht in den Mainstream geschafft haben. Aussagen zufolge sind Faktoren wie eine fehlende Disziplin oder Akzeptanz als Gründe dafür aufzuführen.

Speziell zu erwähnen gilt es, den Aspekt der Wirtschaftlichkeit hinsichtlich der Praktik 40-Stunden Woche. Wie die Person C erläutert, ist es aus Sicht des Managements rentabler, bei Bedarf Überstunden anzuordnen, anstatt eine nachhaltige Arbeitsweise zu fördern.

Wie aus der folgenden Tabelle hervorgeht, wurden in den Interviews diverse Aspekte genannt, die als Gründe für eine geringe Verbreitung von XP als Methode interpretiert werden.

Kategorie	Ausprägung / Gründe
XP-Methode	Akzeptanz Anpassungsfähigkeit Begleitung Beliebtheit Disziplin Dokumentation Domänenwissen Durchhaltevermögen Eigeninitiative *verlagerte* Entscheidungsgewalt *starker* Entwicklungsfokus Erfahrung *vorhandener* externer Druck Extremität Kompetenzen Kultur Managementpraktiken Marketing Motivation Priorisierung Reflexion *starke* technische Orientierung Unkenntnis *der Methode* *ungeeignete* Unternehmensgrösse *falscher* Zeitpunkt Zertifizierung
Hybride Vorgehensmodelle	Adaption *des Ursprungmodells* Änderungsbereitschaft Ergänzung *(Scrum/XP)* *festhalten an* Komfortzone *hybride Vorgehen als* Mainstream *bestehende* Regulationen
Schweiz	Ausbildung *vorhandene* Auslagerung *der Entwicklung* Entwicklungsfähigkeit *vorhandener* Konservatismus *bestehende* Reglementierung Wahrnehmung

Tabelle 5: Gründe aus den Interviews für die geringe Verbreitung von XP

Die Autorin wird aus Platzgründen nicht auf alle einzeln eingehen. Speziell zu erwähnen gilt es jedoch, dass sich aus den Interviews herauskristallisiert hat, dass die Befragten der Meinung sind, dass sich Scrum im Gegensatz zu XP durchgesetzt hat, weil die Methode besser vermarktet wurde und das Marketing durch Zertifizierungen entsprechend auf eine erhöhte Resonanz stösst.

Eine hybride Vorgehensweise wird oftmals bevorzugt. Aus dem Interview mit der Person B geht hervor, dass eine mögliche Begründung darin liegen kann, dass die technischen Empfehlungen, wie sie in XP vorhanden sind, in den meisten anderen Methoden eher zu kurz kommen. Die Person E begründet eine geringe Verbreitung von XP dadurch, dass es lediglich einzelne Praktiken in den Mainstream geschafft haben. Die Methode selbst jedoch nicht, da sie in ihrer Gänze zu schwer ist.

Spezifisch für die Schweiz resultierten in den Interviews Aspekte, die auf eine verbesserungsbedürftige Ausbildung, weniger speziell auf XP als eher auf den Werdegang einer entwicklungsverantwortlichen Person hindeuten (interviewte Person B, C). Hinzu kommt, dass die Wahrnehmung der Relevanz von Software laut der Person C nicht in der Gesellschaft angekommen ist. Wie die Person E erläutert, erhöht ein gewisser Konservatismus in der Schweiz den technologischen Rückstand. Wird zudem berücksichtigt, dass, wie laut der Person C, vielerorts eine reglementorientierte Einstellung herrscht, wird der Einsatz von XP verhindert.

Letztendlich lässt sich die Teilfrage 3 durch die Gründe aus Tabelle 4 und Tabelle 5 ausführlich beantworten.

6 Diskussion der Ergebnisse

In diesem Kapitel wird eine Synthese der Befunde aus der Literatur und der Praxis, gefolgt von einer Interpretation, gemacht. Die Autorin hat als Grundlage dieses Kapitels eine Quantifizierung der Ausprägungen vorgenommen. Dafür wurden analog der Agile Onion, wie sie in der theoretischen Grundlage visualisiert wurde (Abbildung 1), die Cluster *being agile* und *doing agile* definiert. Dabei beinhaltet der Cluster *being agile* Aspekte wie das Mindset, dieKultur und Werte. Dem Cluster *doing agile* hingegen werden Ausprägungen zugeordnet, welche auf Prinzipien, Praktiken, Tools und Prozesse zurückzuführen sind. Einem weiteren Cluster wurden alle Ausprägungen zugeordnet, welche extrinsischer Natur sind, demnach wurde der Cluster *externe Einflussfaktoren* genannt. Die komplette Zuordnung der Ausprägungen und die prozentuale Berechnung können dem Anhang entnommen werden. Die Angaben in Prozente sind dabei jeweils gerundet. Nachfolgend geht die Autorin auf die zentralen Punkte ein. Dabei erfolgt eine Strukturierung dieses Kapitels nach Aspekten, die eine Anwendung einzelner Praktiken verhindern können und solchen, die eine Anwendung der XP-Methode selbst verhindern. Letzteres berücksichtigt auch hybride Vorgehensweisen sowie Beobachtungen, die spezifisch für die Schweiz gelten. Zur Unterstützung der Lesbarkeit werden die Resultate zusätzlich visualisiert.

Aspekte, die eine Anwendung einzelner XP-Praktiken verhindern

Aus der Zuordnung der Praktiken in ihrer Gänze geht hervor, dass die Ausprägungen zu 45% dem Cluster *doing agile* zugeordnet werden können. Gefolgt vom Cluster *being agile* mit 29% und externen Einflussfaktoren mit 27%. Die Summe der Prozentzahlen übersteigt in diesem Fall 100, da die Ausprägung *Kompetenzen* doppelt gezählt wurde; Soziale Kompetenzen wurden *being agile* zugeordnet, fachliche hingegen *doing agile*.

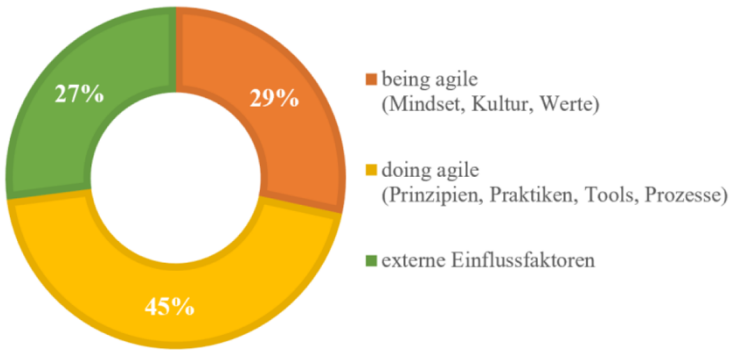

Abbildung 6: Hindernde Faktoren für die einzelnen Praktiken (in Summe)

Wie aus der Literatur hervorgeht, erfordern vor allem Praktiken wie Pair Programming oder Test-Driven-Development fachliches Wissen (Nanthaamornphong & Carver, 2017; Sfetsos et al., 2006), doch auch soziale Kompetenzen wie eine starke Kommunikation (Kunwar, 2019) werden bedingt, was im Interview mit der Person E bestätigt wird.

Bezugnehmend auf die Grafik in der Abbildung 4 geht die Autorin nachfolgend lediglich auf die Praktiken ein, welche mit weniger als 50% in der Praxis im Einsatz sind, da die Autorin davon ausgeht, dass diese Praktiken mehr Schwierigkeiten bereiten und deshalb weniger angewendet werden. Namentlich handelt es sich dabei um die Praktiken Kunde vor Ort (41%), 40-Stunden Woche (26%), Gemeinsame Verantwortlichkeit (31%), Paarprogrammierung (30%), Refactoring (48%), Testen (38%), Einfaches Design (15%) und Metapher (0%). Die Prozentangaben in den Klammern weisen dabei auf, wie häufig die Praktik gemäss den State of Agile Reports in der Praxis durchschnittlich angewendet wird.

Wie aus der Literatur hervorgeht, leidet die Praktik **Kunde vor Ort** unter starken Abhängigkeiten zur Kundin und zum Kunden (Ibrahim et al., 2020; Kunwar, 2019; Mushtaq & Qureshi,2012; Qureshi, 2012), diese bedingte KundInnennähe wird auch in den Interviews mit den Personen A, B und D thematisiert. Werden diese Ausprägungen nun in die definierten Cluster eingeordnet, so wird ersichtlich, dass diese Praktik zu 75% mit Schwierigkeiten zu kämpfen hat, die auf *doing agile* zurückzuführen sind.

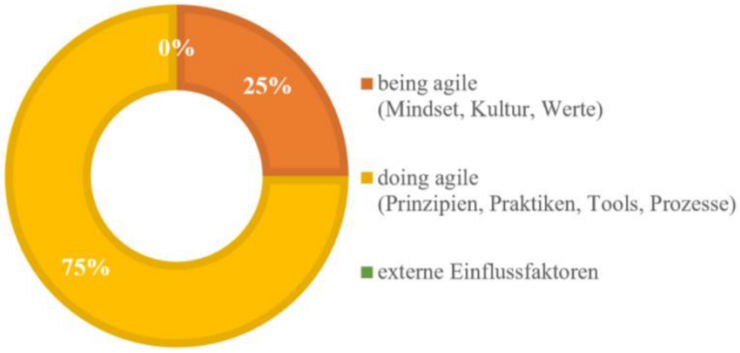

Abbildung 7: Hindernde Faktoren für die Praktik Kunde vor Ort

Demnach verhindern vorwiegend definierte Prinzipien, Praktiken, Prozesse und Tools einen einwandfreien Einsatz dieser Praktik.

Bei der Praktik **40-Stunden Woche** sind zu 67% externe Einflüsse wie die Wirtschaftlichkeit (interviewte Person C), wonach es aus Managementsicht nicht rentiert, nachhaltig zu entwickeln, und die definierte Projektart und -grösse in einem Unternehmen, als hindernd zu betrachten. Letzterer Aspekt erschwert mit zunehmender Grösse des Unternehmens den Einsatz der Praktik (Sfetsos et al., 2006).

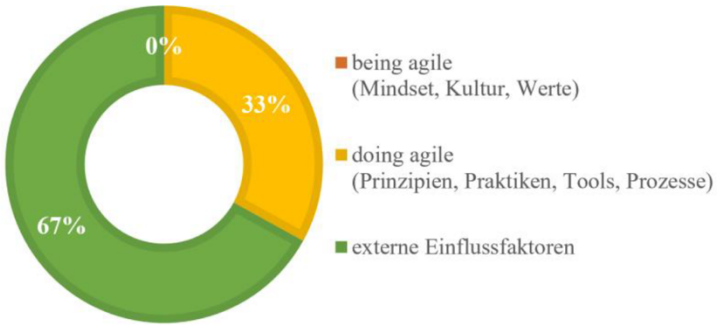

Abbildung 8: Hindernde Faktoren für die Praktik 40-Stunden Woche

Eine **gemeinsame Verantwortlichkeit** scheitert zu 75% an Aspekten, welche *being agile* zugeordnet werden. Demnach fehlt es laut der interviewten Person A oftmals am richtigen Mindset, was auch bereits durch Auer & Miller (2002) festgestellt wurde.

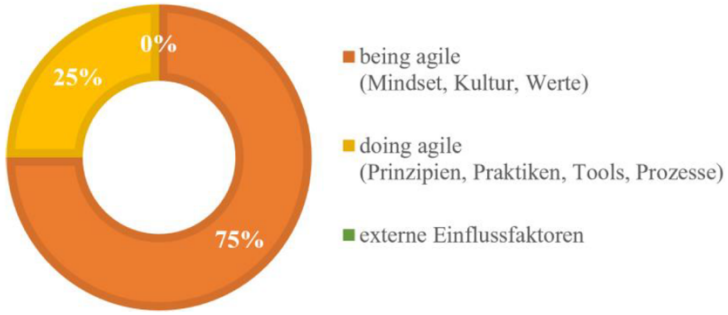

Abbildung 9: Hindernde Faktoren für die Praktik gemeinsame Verantwortlichkeit

Die Praktik der **Programmierung in Paaren** wurde sowohl in der Literatur als auch in den Interviews mehrfach diskutiert. Aus der Analyse geht hervor, dass sie mit 43% am häufigsten mit externen Einflussfaktoren zu kämpfen hat, gefolgt von 38% *being agile* und 24% *doingagile*.

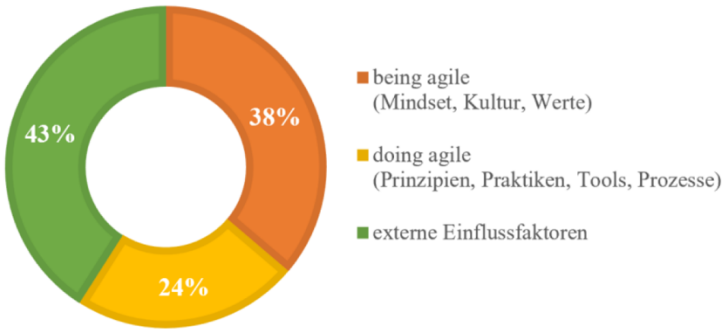

Abbildung 10: Hindernde Faktoren für die Praktik Paarprogrammierung

Unter externe Einflussfaktoren fallen in diesem Kontext Unterbrechungen durch Drittpersonen (Tsyganok, 2016), sowie Themen der Logistik (interviewte Person E). Ein weiterer Aspekt ist die fehlende Akzeptanz gegenüber der Praktik. Diese kann sowohl von

der Entwicklung her kommen (interviewte Person E) als auch managementseitig sein (Kunwar, 2019). Bei letzteren fehlt sie meistens aufgrund der Annahme, dass die Kosten sich durch diese Praktik erhöhen (Kunwar, 2019), was auch durch die interviewte Person C geäussert wurde. Entgegen einigen Quellen aus der Literatur war es nicht angedacht, dass die Paarprogrammierung nur unter Programmierenden stattfindet. Wie die interviewte Person E erzählt, ist es durchaus möglich, dass beispielsweise die Kundin und der Kunde mit einem Programmierenden zusammensitzt.

Das **Refactoring** wurde in den Interviews nicht erwähnt, wonach keine Vergleichsbasis vorliegt. Aus der Literatur geht jedoch hervor, dass mit 57% am häufigsten Aspekte, die auf Prozesse und Prinzipien (*doing agile*) zurückzuführen sind, als hindernd betrachtet werden.

So sind diverse Abhängigkeiten, sei es zu Unit-Tests als auch zum Architekturdesign als solche Aspekte zu betrachten. Hinzu kommen notwendige fachliche Kompetenzen, um diese Praktik zielgerichtet anwenden zu können (Nanthaamornphong & Carver, 2017).

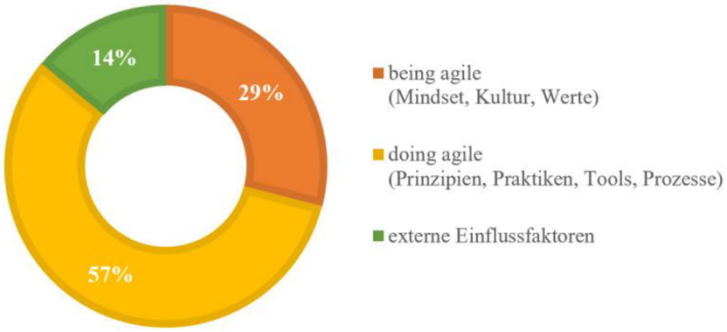

Abbildung 11: Hindernde Faktoren für die Praktik Refactoring

Eine testgetriebene Entwicklung, wie sie in der Praktik **Testen** empfohlen wird, scheitert zu jeweils 40% am häufigsten an Aspekten, welche Prozesse und Prinzipien (*doing agile*) sowie externe Einflüsse betreffen.

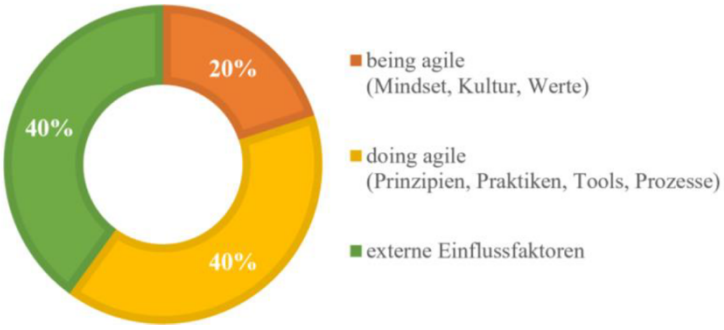

Abbildung 12: Hindernde Faktoren für die Praktik Testen

Nebst der mangelnden Akzeptanz (interviewte Person E) und dem fehlenden Mindset (interviewte Personen E, B) für die Praktik ist der damit verbundene Aufwand der am meisten diskutierte Aspekt (Anwer et al., 2017; Ibrahim et al., 2020; Ott, 2018; Qureshi, 2012). Aus dem Interview mit der Person B geht hervor, dass dieser Aufwand nicht selten durch Entwickelnde gescheut wird.

Die Praktik des **einfachen Designs** scheitert zu 80% an organisatorischen Themen, welche mit *doing agile* einhergehen. Aus den Interviews wurde ergänzend zu den Aspekten der Literatur, welche die fehlenden Anleitungen und mangelnde Diskussion betreffen (Sfetsos et al., 2006), auch der Einfluss der Unternehmensgrösse (interviewte Person B) als hindernd geäussert. Letzteres wurde jedoch als extrinsischer Faktor eingestuft und bildet somit die restlichen 20% in der Zuordnung der Cluster.

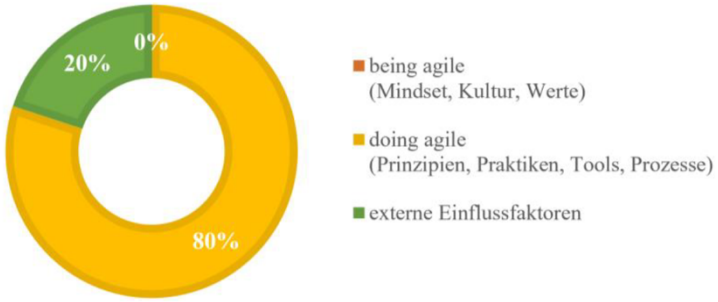

Abbildung 13: Hindernde Faktoren für die Praktik einfaches Design

Die Praktik der **Metapher** resultierte im Vergleich der State of Agile Reports mit 0% als einzige Praktik, die überhaupt nicht verwendet wird. Zu 60% geht aus der Analyse hervor, dass dafür externe Faktoren verantwortlich sind. Wie die Person E im Interview geäussert hat, wurde die Metapher von den Leuten nicht verstanden. Diese Aussage verhärtet sich durch die Literatur, denn darin wird die Metapher als nebulös und unzureichend verstanden beschrieben (Donick, 2020; Keller, 2019; Sfetsos et al., 2006). Wie aus dem technischen OOPSLA Programm von 2002 hervorgeht, war Beck bereits damals diese Unklarheit bekannt.

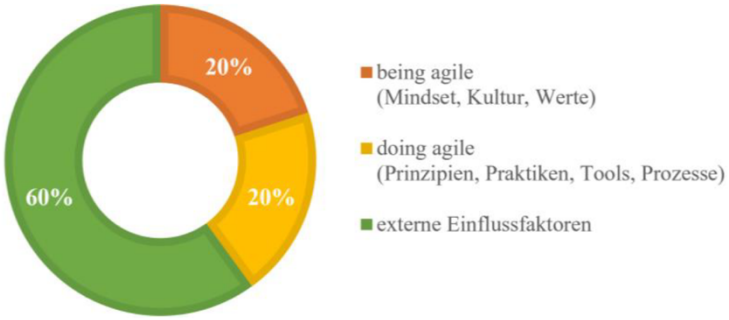

Abbildung 14: Hindernde Faktoren für die Praktik Metapher

Summierend lässt sich feststellen, dass die Praktiken, die reduzierter angewendet werden, zu 45% an Aspekten scheitern, die dem Cluster *doing agile* zugeordnet werden können. Mit jeweils 28% sind die Cluster *being agile* und externe Faktoren weiter dafür verantwortlich. Demnach sind Themen, die Prozessen und Prinzipien zugewiesen werden können, die häfigsten hindernden Ursachen für eine Anwendung der XP-Praktiken.

Aspekte, die eine geringe Anwendung von XP begründen

Zur XP-Methode selbst konnten insgesamt 44 Ausprägungen erfasst werden. Die Zuweisung zu den drei Clustern zeigt, dass die Methode selbst zu 48% aufgrund von Aspekten, die *being agile* zugeordnet werden können, scheitert. Die Kultur und das Mindset sind demnach entscheidend für die Anwendung von XP. Gefolgt von 41% Aspekten, die *doing agile*, also Prozesse und Prinzipien, betreffen. Am geringsten dafür verantwortlich scheinen externe Einflussfaktoren mit 14%.

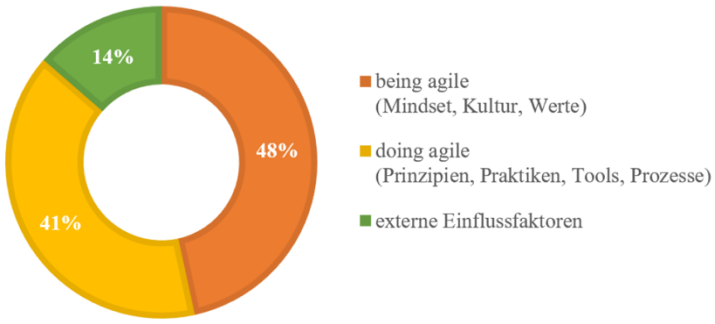

Abbildung 15: Hindernde Faktoren für die XP-Methode

XP geht aus den Interviews als Methode mit starkem Entwicklungsfokus hervor (interviewte Person B) und spricht demnach besonders Personen aus der Entwicklung an (interviewte Person D). Dies wird auch durch Smoczyńska et al. (2019) bestätigt. Aufgrund des starken Fokus auf die Entwicklung vernachlässigt XP laut mehreren Quellen sogenannte Managementpraktiken (Ibrahim et al., 2020; Mushtaq & Qureshi, 2012; Smoczyńska et al., 2019).

Wie von den interviewten Personen A, C und D geäussert, hat Scrum marketingtechnisch überragend dominiert. Während XP laut der Person D ein ‹grassroots-approach› geblieben ist, entstand für Scrum eine Lobby, in welcher durch Zertifizierungen eine erhöhte Resonanz erzielt werden konnte (interviewte Personen B, D). Wie in vier von fünf Interviews erwähnt,liegt die Entscheidungsgewalt über das gewählte Vorgehen vorwiegend beim Management (interviewte Personen B, C, D, E). Wird nun berücksichtigt, dass Marketingmassnahmen vorwiegend auf Managementebene ausgerichtet sind und die

Entscheidungsgewalt beim Management liegt, werden demnach Vorgehensweisen bevorzugt, die managementorientiert sind. XP ist keine solche Vorgehensweise. Dadurch kann sich eine stärkere Verbreitung von Scrum erklären und implizit offengelegt werden, weshalb XP, wie in der Literatur angedeutet, mit einer deutlich reduzierten Akzeptanz zu kämpfen hat (Conboy et al., 2011; Hekkalaet al., 2017; Marquardt, 2011).

Hinsichtlich hybrider Vorgehensweisen und -modellen wurden gesamt 13 Ausprägungen ermittelt. 38% davon können dem Cluster *doing agile* zugeordnet werden. Gefolgt von jeweils31% *being agile* und externe Einflussfaktoren. Wonach XP in hybriden Konstellationen somit am häufigsten an Aspekten der Organisation, wie Prozesse und Prinzipien, scheitert.

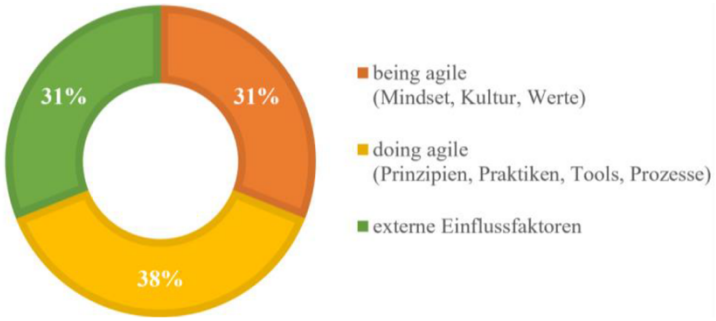

Abbildung 16: Hindernde Faktoren für hybride Vorgehensmodelle

Hybride Vorgehensmodelle entstehen durch eine Kombination unterschiedlicher Methoden. Wie sich herausgestellt hat, ergänzen sich Scrum und XP gut, da der Fokus in Scrum auf Managementpraktiken liegt und in XP hingegen die Entwicklung im Zentrum steht (Smoczyńska et al., 2019). Wie die interviewte Person D äussert, funktioniert Scrum in Softwareentwicklungsprojekten nicht ohne XP, da es sich ansonsten um eine Kombination von Scrum mit einem klassischen Vorgehen handeln würde, wonach das Vorgehen in der Softwareentwicklung nicht agil ist und demnach der Arbeitsweise auch kein agiler Ansatz zugrunde liegt. Häufig werden Vorgehensweisen adaptiert, dabei können die Gründe dafür unterschiedlich sein. Wie Kuhrmann et al. (2017) erwähnen, folgt eine solche Adaption einem pragmatischen Selektionsprozess. Auch Sfetsos et al. (2006) machen dieselbe Aussage. Wie aus den Interviews hervorgeht, bedarf diese Adaption jedoch eines fundierten Entscheides basierend auf Kenntnissen und Erfahrungen der adaptierten Methoden. Anders können häufig Kombinationen von klassischen und agilen Vorgehensweisen entstehen, diese sind aber nicht im Sinne einer agilen Softwareenentwicklung (interviewte Person C, D). Bezugnehmend auf die Auswertung der State of Agile Reports in Abbildung 3, aus welcher Scrum als Spitzenreiter hervorgeht, werfen diese Erkenntnisse die Frage auf, nach welchem Vorgehen – klassisch oder agil – die

Software in den befragten Unternehmen entwickelt wird. Im Zusammenhang mit dieser Frage kann die Moore's Curve, wie sie in der Abbildung 5 ersichtlich ist, berücksichtigt werden. Denn darin spricht die interviewte Person E von einer Scheinagilität diverser Unternehmen, die sich dem Trend anschliessen, jedoch strukturell keine Veränderungen vornehmen möchten. Feststeht, dass eine Mischung von Scrum und XP situativ als zielführend identifiziert werden kann. Wonach die beiden Methoden nicht als konkurrierend, sondern sich ergänzend betrachtet werden können.

In den Interviews wurden Ausprägungen erhoben, die spezifisch zur Anwendung von XP in der Schweiz geäussert wurden. Gleichermassen dominierend gelten in der Schweiz Aspekte des *being agile* sowie externe Einflussfaktoren mit jeweils 50% als hindernd für XP.

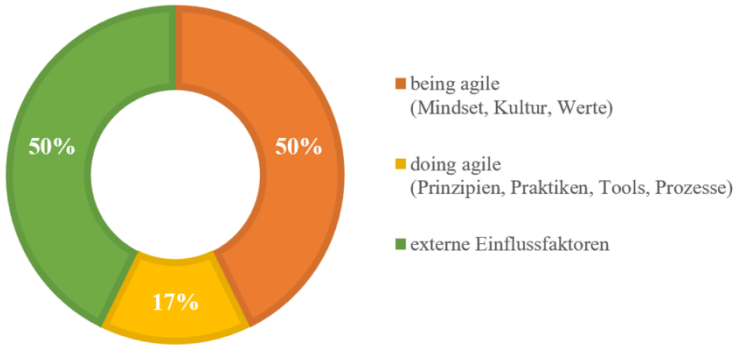

Abbildung 17[4]: Hindernde Faktoren für eine Anwendung von XP und XP-Bestandteilen in der Schweiz

Demnach sind die Kultur, das Mindset und die Werte, welche XP voraussetzt, in der Schweiz laut Aussagen nicht gegeben (interviewte Personen B, C, E). Aspekte wie eine Lernbereitschaft der Entwickelnden, die auf Eigeninitiative basiert, sieht die Person C als nicht immer vorhanden. Diese mangelnde Bereitschaft zur Ausbildung gekoppelt mit einem Wissenstransfer durch Training ‹on the job›, wie er von den Personen C und E erläutert wird, kann zu unzureichenden Kompetenzen und Qualifikationen führen, um agil arbeiten zu können. Hinzu kommt, dass laut der Person C in der Schweizer Gesellschaft Software noch nicht alsKernelement wahrgenommen wird. Auch wird das Ausbildungsangebot bemängelt (interviewte Personen B, C). Durch ein unzureichendes Ausbildungsangebot kombiniert mit einerreduzierten Bereitschaft zur Ausbildung leiden die Programmierfähigkeiten. Wird nun berücksichtigt, dass Programmierende ihre Kompetenzen teilweise ‹on the job› erlangen, so kann eine zunehmend sinkende Qualität der Software, wie sie durch die Person C erwähnt wird, begründet werden. Zudem wird die Frage aufge-

[4] Die Prozentzahl übersteigt hier im Total 100, da die Ausprägung der Ausbildung doppelt zugewiesen wurde. Das Ausbildungsangebot wurde externen Einflussfaktoren zugeordnet. Die Bereitschaft zur Ausbildung wurde *being agile* zugeordnet.

worfen, ob die fehlende Wahrnehmung der Gesellschaft eine Begründung dafür sein kann, dass auch das Ausbildungsangebot in der Schweiz nicht so ausgelegt ist, dass Methoden wie XP stärker thematisiert werden.

Wie durch die Person C geäussert, spielt auch die Kultur und Reglementorientierung der Schweiz eine entscheidende Rolle. Aus der Literatur geht hervor, dass XP mit einer schlechten Dokumentation einher geht (Qureshi & Bajaber, 2016). Dadurch können entsprechende Artefakte fehlen, auf die sich eine reglementorientierte Person stützen kann. Anders als in Scrum, wo ein Scrum Guide als Anleitung betrachtet werden kann, bleibt in XP eine solch kompakte Orientierung aus. Personen, die dazu neigen, sich an Vorschriften zu halten, tendieren daher, auf eine besser dokumentierte Methode auszuweichen.

Folgend werden im Fazit in Kapitel 7 die Schwerpunkte der Ausprägungen konkludiert, welche einen geringen Einsatz von XP und den Praktiken, die gemäss der Abbildung 4 mit weniger als 50% in der Praxis zur Anwendung kommen, begründen.

7 Fazit

Diese Arbeit untersucht die Ursachen, die dazu geführt haben, dass XP als agile Softwareentwicklungsmethode kaum noch im Einsatz ist, obschon die Praktiken daraus vermehrt zur Anwendung kommen. Die definierte Forschungsfrage der Untersuchung lautet wie folgt:

Weshalb findet XP als agile Entwicklungsmethode in der Schweiz nur noch geringe Anwendung, obwohl die Praktiken in der Praxis etabliert sind?

Die Forschungsfrage beinhaltet drei Teilfragen. Diese werden im Fazit jedoch nicht thematisiert. Deren Beantwortung findet sich in den Kapiteln 2.6 (TF1), 4.2 (TF2) und 5.6 (TF3).

Das Ziel dieser Arbeit ist es, den in der Forschungsfrage erläuterten Sachverhalt zu erklären und Schwierigkeiten offenzulegen, bei deren Berücksichtigung eine agile Arbeitsweise optimiert werden kann. Es ist nicht das Ziel, Handlungsempfehlungen auszusprechen oder nach Möglichkeiten zu suchen, die Methode wieder bekannter zu machen.

Wie aus der Literaturrecherche hervorgeht, reichen relevante Quellen zu dieser Fragestellung bis ins Jahr 2000 zurück. Daraus lässt sich schlussfolgern, dass bereits vor 21 Jahren Schwierigkeiten im Zusammenhang mit XP zum Vorschein kamen. Die Forschungsfrage ist auf die Schweiz eingegrenzt, aus der Literatur kristallisiert sich jedoch heraus, dass die Problematik auch über die Landesgrenze hinweg besteht. Bei der Synthese der Ergebnisse aus der Literatur und den Interviews hat sich die Komplexität der Thematik offenbart. Die ermittelten Ursachen, die eine geringe Anwendung begründen, weisen eine erhöhte Heterogenität auf, die keine Aussage zu einer spezifisch verantwortlichen Ursache ermöglicht. Vielmehr wird ein komplexes Phänomen mit soziologischen, methodischen, fachlichen und kulturellen Aspekten sichtbar. Speziell zu erwähnen, gilt es dabei das Spannungsfeld zwischen Management und Entwicklung, welches oftmals für die Entstehung hybrider Vorgehensweisen verantwortlich ist, da XP alleine sehr entwicklungszentriert ist.

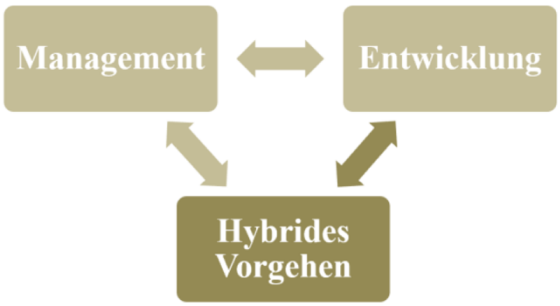

Abbildung 18: Spannungsfeld zwischen Management und Entwicklung

Einzelne Praktiken kommen demnach weiterhin aus der Entwicklung zur Anwendung. Dabei findet bestenfalls ein situativ selektiver Prozess statt, in welchem entschieden wird, worauf verzichtet und was weiterhin inkludiert wird. Schlimmstenfalls wird bei der Adaption lediglich entfernt, was als störend empfunden wird.

Im Kontext der Schweiz resultierte eine mangelnde Wahrnehmung der Gesellschaft für IT als Kernstück als eine potenzielle Ursache für die geringe Verbreitung. Es gilt anzunehmen, dass bei einer fehlenden Wahrnehmung die Thematik auch im Bildungswesen als weniger relevant eingestuft wird. Demnach wird auch das Ausbildungsangebot nicht so ausgelegt, dass es eine agile Arbeitsweise unterstützt. Hierbei handelt es sich aber nicht um XP-spezifische Befunde, sondern vielmehr um Aspekte, die einen gesamt agilen Ansatz erschweren. Wie die Analyse ergeben hat, sind es nämlich überwiegend Angelegenheiten der Kultur, des Mindsets und der Werte, die einen Einsatz von XP verhindern. Daher scheitert die Methode, weil das Fundament für eine agile Arbeitsweise nicht gegeben ist. XP gilt als äusserst schwierig und anspruchsvoll, es gilt anzunehmen, dass die Methode ohne starkes Fundament scheitert. Die fehlende Akzeptanz, mangelnde Fehlerkultur sowie eine Entscheidungsgewalt, die selten bei der ausführenden Instanz liegt, tragen zusätzlich zu einer niedrigen Präsenz von XP bei.

In Summe scheitern die Praktiken mit 41% am häufigsten an Themen, welche Prinzipien, Praktiken, Tools und Prozesse betreffen. XP als Methode hingegen scheitert mit 48% am häufigsten an Aspekten, die das Mindset, die Kultur und die Werte anbelangen. Wird die Schweiz gesondert betrachtet, so ergeben sich mit jeweils 50% sowohl Aspekte, die das Mindset, die Kultur und Werte anbelangen, als auch externe Faktoren als für XP hindernd.

Ungeachtet der Methode im Einsatz geht aus der Arbeit hervor, dass eine Kultur der Veränderung und ein für Agilität förderndes Mindset zentrale Erfolgsfaktoren für eine gesamt agile Arbeitsweise sind.

8 Kritische Reflexion

Die gewählte Untersuchungsmethode mit dem Mix von Literatur und Praxis erwies sich für die Arbeit als zielführend. Dadurch konnten vielfältige Aspekte ermittelt werden, die als Bereicherung für die Klärung des Sachverhalts wahrgenommen werden. Die Vielfältigkeit der Resultate erhöht jedoch massgeblich die Komplexität der Thematik. Diese Komplexität und Heterogenität der Befunde im Anschluss auf den Umfang der Bachelorthesis zu reduzieren, gestaltete sich als äusserst herausfordernd. Eine Simplifizierung zu Gunsten der Lesenden war daher unumgänglich. Detailliert werden die Befunde im Anhang aufgeführt. Durch die Erhebungen kann aufgezeigt werden, wie umfangreich der Sachverhalt tatsächlich ist.

In der Schweiz gibt es wenig wissenschaftliche Literatur zu der Thematik. Die Aussagen im theoretischen Teil der Arbeit, welche die Schweiz betreffen, basieren auf drei Studien, deren Aktualität nicht gegeben ist. Die Aussagekraft für die Schweiz ist somit reduzierter möglich als auf internationaler Ebene, wo mit dem State of Agile Report jährlich der Stand präsentiert wird. Durch diese Arbeit konnte die Autorin einen qualitativen Beitrag zum Thema Agilität mit dem Schwerpunkt auf dem Einsatz von XP in der Schweiz leisten.

Die Auswahl der Interviewpersonen war für die offene Fragestellung dienlich. Die Befragten verfügen allesamt über ein fundiertes Wissen im Bereich agiler Methodologien und haben auch persönlich Erfahrungen mit XP gesammelt. Da es sich bei den Befragten um Personen mit Herkunft aus der Entwicklung handelt, konnte nicht ausgeschlossen werden, dass XP nicht von vornherein favorisiert wird, da die Methode als entwicklungszentriert gilt. Die Aussagen in den Interviews deuten jedoch weniger auf eine Favorisierung von XP als vielmehr auf eine gänzlich wünschenswerte agile Arbeitsweise hin. Obschon in der Literatur bereits diverse Aspekte, die eine geringe Verbreitung von XP begründen, erhoben wurden, ist es der Autorin gelungen, aus den Interviews weitere spezifische Erkenntnisse zu erlangen.

Eine der interviewten Personen war Joseph Pelrine. Joseph gilt als einer der Pioniere von XP. Er hat Kent Beck bei der Entwicklung der Methode assistiert. Einerseits erfreut es die Autorin ausgesprochen, eine Person dieses Kalibers interviewt zu haben, andererseits erhöhte die Expertise den Bearbeitungsaufwand, da die Aussagen im Transkript nicht anonymisiert werden konnten, ohne dass Rückschlüsse möglich sind. Die Autorin hat initial angedacht, die Aussagen von Joseph als solche zu kennzeichnen, hat sich jedoch später dagegen entschieden und dessen Aussagen analog der anderen befragten Personen im Fliesstext nicht mit dem Namen, sondern lediglich mit ‹interviewte Person X› gekennzeichnet. Grund dafür war, dass die Aussagen von Joseph im Kontext durch die

Lesenden nicht stärker gewichtet werden sollten. Schlussendlich geht es um den Sachverhalt und die Lesenden sollten nicht durch die Erfahrung einer bestimmten Person vorbelastet werden.

Folgend thematisiert die Autorin Limitationen der Arbeit und gibt eine Empfehlung zur weiteren Forschung.

Da es sich um ein qualitatives Vorgehen handelt, können keine quantitativen Aussagen über die Signifikanz der einzelnen Aspekte gemacht werden. Es gilt jedoch anzunehmen, dass Faktoren, die sowohl in der Literatur als auch in den Interviews erwähnt wurden, mit höherer Relevanz einzustufen sind.

Die XP-Praktiken werden in der Praxis auch situativ adaptiert. Inwiefern eine solche Adaption eine Assimilierung mit dem Grundgedanken von XP zulässt, steht offen. In der Arbeit konnte nicht eindeutig ermittelt werden, inwiefern die Praktiken im Einsatz tatsächlich in ihrer – nach XP vorgesehenen – Extremität angewendet werden.

Aus der Analyse der Daten hat sich ergeben, dass ein Spannungsfeld zwischen Management und Entwicklung besteht. Diese Arbeit berücksichtigt nicht die Sicht der Managementebene. In einem weiteren Schritt wäre es möglich, das Management bezüglich der Thematik abzuholen und dessen Beweggründe zu ermitteln, die zum Entscheid einer agilen Vorgehensweise führen.

Die Thematik ist mit verhaltenspsychologischen Aspekten behaftet, wonach die Autorin empfiehlt, dass sich Personen mit einem Hintergrundwissen in Psychologie und Kenntnissen im Umfeld der Wirtschaft und des Managements einer fortführenden Untersuchung, in welcher die Aspekte des agilen Mindsets untersucht werden, annehmen.

9 Literaturverzeichnis

Aichele, C. & Schönberger, M. (2016). Mit Struktur und Methode in die projektindividuelle App-Entwicklung. In C. Aichele & M. Schönberger (Hrsg.), *App-Entwicklung – effizient und erfolgreich* (S. 77–133). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-13685-7_5

Alam, D. & Gühl, U. (2020). *Projektmanagement für die Praxis*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-62170-7

Alpar, P., Alt, R., Bensberg, F. & Weimann, P. (2019). Phasenmodelle in der Softwareentwicklung. In P. Alpar, R. Alt, F. Bensberg & P. Weimann (Hrsg.), *Anwendungsorientierte Wirtschaftsinformatik* (S. 347–402). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658- 25581-7_13

Andersson, F. & Sandahl, E. (2021). Agile project management practises for digital production: Bachelor's thesis in Industrial management and production engineering. Göteborg. https://odr.chalmers.se/bitstream/20.500.12380/302202/1/E2020_122.pdf

Anwer, F., Aftab, S., Waheed, U. & Muhammad, S. S. (2017). Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey. In INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY SCIENCES AND ENGINEERING (Hrsg.), *Volume 8, Issue 2, March 2017* (Vol. 8, No. 2, S. 1–10). http://www.ijmse.org/Volume8/Issue2.html

Auer, K. & Miller, R. W. (2002). *Extreme programming applied: Playing to win* (1. Aufl.). *The XP series*. Addison-Wesley.

Bass, J. M. (2012). Influences on Agile Practice Tailoring in Enterprise Software Development. In *2012 Agile India* (S. 1–9). IEEE. https://doi.org/10.1109/AgileIndia.2012.15

Beck, K. (2002). The Metaphor Metaphor. In OOPSLA (Vorsitz), *17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications,* Seattle. http://www.oopsla.org/2002/fp/files/spe-metahpor.html

Beck, K. (2004). *Extreme Programming: Die revolutionäre Methode für Softwareentwicklung in kleinen Teams*. Das Manifest. *Programmer's choice*. Addison-Wesley.

Beck, K. & Andres, C. (2005). *Extreme programming explained: Embrace change* (2nd ed.). Addison-Wesley. http://proquest.safaribooksonline.com/0321278658

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001a). *Manifest für Agile Softwareentwicklung*. https://agilemanifesto.org/iso/de/manifesto.html

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S.,Schwaber, K., Sutherland, J. & Thomas, D. (2001b). *Prinzipien hinter dem Agilen Manifest*. https://agilemanifesto.org/iso/de/principles.html

Beck, K., Fowler, M. & Kohnke, J. (2001). *Planning extreme programming*. *The XP series*. Addison-Wesley.

Bibik, I. (2018). *How to Kill the Scrum Monster: Quick Start to Agile Scrum Methodology and the Scrum Master Role*. Apress; Imprint: Apress. https://link-springer-com.ezproxy.fhgr.ch/con-tent/pdf/10.1007%2F978-1-4842-3691-8.pdf https://doi.org/10.1007/978-1-4842-3691-8

Böhm, J. (2019). *Erfolgsfaktor Agilität*. Springer Fachmedien Wiesbaden.https://doi.org/10.1007/978-3-658-25085-0

Brandt-Pook, H. & Kollmeier, R. (2020). *Softwareentwicklung kompakt und verständlich*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-30631-1

Bürki, A., Mourgue d'Algue, H., Kruschitz, B. & Stoupa, L. F. (01/2020). *HERMES 5.1: Projektmanagementmethode für alle Projekte Inklusive Agilität*. Referenzhandbuch. https://www.bundespublikationen.admin.ch/cshop_mimes_bbl/48/48DF3714B1101EEA88B560F0070A7B82.pdf

Cockburn, A. (2002). *Agile software development*. *The Agile software development series*. Addison-Wesley. https://www.researchgate.net/publication/235616359_Agile_Software_Develop-ment/stats

Conboy, K., Coyle, S., Wang, X. & Pikkarainen, M. (2011). People over Process: Key Challenges in Agile Development. *IEEE Software*, *28*(4), 48–57. https://doi.org/10.1109/MS.2010.132

Dams, C. M. (2019a). Agiles Mindset. In C. M. Dams (Hrsg.), *essentials. Agiles Event Management* (S. 5–7). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-25500-8_2

Dams, C. M. (2019b). Transformation vom agilen Projektmanagement zum Agile Event Management. In C. M. Dams (Hrsg.), *essentials. Agiles Event Management* (S. 9–11). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-25500-8_3

Dehn, N. (2020, 28. August). *Umfragestudie zur Nutzung von Methoden und Praktiken in unterschiedlichen Softwareprojektdisziplinen: Bachelorarbeit im Studiengang Informatik.* Institut für Praktische Informatik. https://www.pi.uni-hannover.de/fileadmin/pi/se/Stud-Arbei- ten/2020/Dehn2020.pdf

Dhoodhanath, P. & Quilling, R. (2020). Case study: Factors that hinder and support the adoption of Pair Programming in an agile software development company. In *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)* (S. 1–7). IEEE. https://doi.org/10.1109/icABCD49160.2020.9183869

Donick, M. (2020). Kommunikation bei der Softwareentwicklung. In M. Donick (Hrsg.), *Nutzerverhalten verstehen – Softwarenutzen optimieren* (S. 23–45). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-28963-8_2

Döring, N. & Bortz, J. (2016). *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-41089-5

Fausten, L. (2016). *Architektur-Diskussionen in Scrum Prozessen: Eine empirische Studie* [Masterarbeit]. Hochschule für Angewandte Wissenschaften Hamburg, Hamburg. https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/fausten.pdf

Freedman, R. (2016). *The Agile Consultant*. Apress. https://doi.org/10.1007/978-1-4302-6053-0

Freyth, A. (2019). *Persönliche Veränderungskompetenz und Agilität stärken*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-22848-4

Gläser, J. & Laudel, G. (2010). *Experteninterviews und qualitative Inhaltsanalyse als Instrumente rekonstruierender Untersuchungen* (4. Aufl.). *Lehrbuch*. VS Verlag. http://d-nb.info/1002141753/04

Halstenberg, J., Pfitzinger, B. & Jestädt, T. (2020). *DevOps*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-31405-7

Hanschke, I. (2017). *Agile in der Unternehmenspraxis*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-19158-0

Hanser, E. (2010). Das Agile Manifest. In E. Hanser (Hrsg.), *eXamen.press. Agile Prozesse: Von XP über Scrum bis MAP* (S. 9–11). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642- 12313-9_2

Hekkala, R., Stein, M.-K., Rossi, M. & Smolander, K. (2017). Challenges in Transitioning to an Agile Way of Working. In HICSS – Hawai'i International Conference on System Sciences (Hrsg.), *Proceedings of the Annual Hawaii International Conference on System Sciences, Proceedings of the 50th Hawaii International Conference on System Sciences (2017)* (5869-5878). Hawaii International Conference on System Sciences. https://doi.org/10.24251/HICSS.2017.707 Highsmith, J. (2001). *History: The Agile Manifesto*. https://agilemanifesto.org/history.html

Hofert, S. (2018). *Agiler führen*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-18561-9

Ibrahim, M., Aftab, S., Bakhtawar, B., Ahmad, M., Iqbal, A., Aziz, N., Javeid, M. S. & Ihnaini, B. N. S. Dr. (2020). Exploring the Agile Family: A Survey. In IJCSNS (Hrsg.), *International Journal of Computer Science and Network Security* (Vol. 20 No. 10, S. 163–179). https://doi.org/10.22937/IJCSNS.2020.20.10.22

Keller, H. B. (2019). Herstellungsprozesse für Software. In H. B. Keller (Hrsg.), *Entwicklung von Echtzeitsystemen* (S. 27–51). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-26641-7_2

Krodel, F. (2013). Potenziale und Grenzen agiler Vorgehensmodelle: Eine Best Practice-Analyse. In GPM Deutsche Gesellschaft für Projektmanagement (Hrsg.), *projekt-MANAGEMENT aktuell* (1. Aufl., 01/2013, S. 34–40). https://www.wisonet.de/document/PM7a8db2fbbc3247d569f65ba0dc32bdb9e7b11474

Kuhrmann, M., Diebold, P., Munch, J., Tell, P., Trektere, K., McCaffery, F., Garousi, V., Felderer, M., Linssen, O., Hanser, E. & Prause, C. R. (2019). Hybrid Software Development Approaches in Practice: A European Perspective. *IEEE Software*, *36*(4), 20–31. https://doi.org/10.1109/MS.2018.110161245

Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Garousi, V., Felderer, M., Trektere, K., McCaffery, F., Linssen, O., Hanser, E. & Prause, C. R. (2017). Hybrid software and system development in practice: waterfall, scrum, and beyond. In R. Bendraou, D. Raffo, H. LiGuo & F. M. Maggi (Hrsg.), *Proceedings of the 2017 International Conference on Software and System Process* (S. 30–39). ACM. https://doi.org/10.1145/3084100.3084104

Kunwar, S. (2019). Enabling and Limiting factors in eXtreme Programming (XP) with Evaluation Framework. *SCITECH Nepal*, *14*(1), 50–62. https://doi.org/10.3126/scitech.v14i1.25534

Marquardt, K. (2011). Extreme Programming. In Berleb Media GmbH (Hrsg.), *Projekt Magazin: Agiles Projektmanagement* (10. Aufl., Bd. 10, S. 80–100). https://www.wisonet.de/document/PROJ_5ad8494e548c0915698e99423fb8b4af98bd7c80

Mayring, P. (2015). *Qualitative Inhaltsanalyse: Grundlagen und Techniken* (12., überarbeitete Auflage). Beltz Verlag. http://d-nb.info/1063369835/04

Meier, A. & Kropp, M. (01.2013a). *Swiss Agile Study 2012: Agile Software-Entwicklung in der Schweiz*. www.swissagilestudy.ch. http://blogs.fhnw.ch/swissagilestudy/files/2015/05/SwissAgileStudy2012.pdf

Meier, A. & Kropp, M. (2013b). Vermittlung von agiler Softwareentwicklung im Unterricht. In A. Spillner & H. Lichter (Vorsitz), *SEUH 2013,* RWTH Aachen. http://ceur-ws.org/Vol-956/S3_Pa-per2.pdf

Meier, A. & Kropp, M. (05.2015). *Swiss Agile Study 2014: Software-Entwicklung in der Schweiz*. www.swissagilestudy.ch. http://blogs.fhnw.ch/swissagilestudy/files/2015/05/SwissAgileStudy2014.pdf

Meier, A. & Kropp, M. (08.2017). *3. Swiss Agile Study: Agile und hybride Software-Entwicklung in der Schweiz*. www.swissagilestudy.ch. http://blogs.fhnw.ch/swissagilestudy/files/2017/09/3.SwissAgileStudy.pdf

Michl, T. (2018). Das agile Manifest – eine Einführung. In M. Bartonitz, V. Lévesque, T. Michl, W.Steinbrecher, C. Vonhof & L. Wagner (Hrsg.), *Agile Verwaltung* (S. 3–13). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-57699-1_1

Mushtaq, Z. & Qureshi, M. R. J. (2012). Novel Hybrid Model: Integrating Scrum and XP. *International Journal of Information Technology and Computer Science*, *4*(6), 39–44. https://doi.org/10.5815/ijitcs.2012.06.06

Nanthaamornphong, A. & Carver, J. C. (2017). Test-Driven Development in scientific software: a survey. *Software Quality Journal*, *25*(2), 343–372. https://doi.org/10.1007/s11219-015-9292-4

Ott, M. (2018). Deployment, Testing und Management. In M. Ott (Hrsg.), *X.media.press. Apps effektiv managen und vermarkten* (S. 45–63). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-22296-3_5

Pelrine, J. (2011). On Understanding Software Agility – A Social Complexity Point Of View. In *E:CO Issue* (Bd. 13, S. 26–37). https://www.researchgate.net/profile/Joseph-Pelrine/publication/295174037_On_Understanding_Software_Agility_-_A_Social_Complexity_Point_Of_View/links/5e8b09e1a6fdcca789f83bf1/On-Understanding-Software-Agility-A- Social-Complexity-Point-Of-View.pdf

Pelrine, J., Uhtes, R. & Noack, C. (2000). *Why is it hard to learn eXtreme Programming? Philosophical and psychological aspects of learning a new methodology*. https://citese-erx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.3139&rep=rep1&type=pdf

Preußig, J. (2020). *Agiles Projektmanagement: Agilität und Scrum im klassischen Projektumfeld* (2. Auflage). Haufe Group. https://doi.org/10.34157/9783648137789

Proba, D. J. (2021). *Technikkompositionen für das situative Methoden-Engineering agiler Methoden* (5068) [Qualifikationsarbeit (Doctoral)]. Universität St. Gallen, St. Gallen. https://www.al-exandria.unisg.ch/262411/

Qureshi, M. R. J. (2012). Empirical Evaluation of the Proposed eXScrum Model: Results of a Case Study. *International Journal of Computer Science and Issues* (Vol. 8, No. 2), Artikel Issue 3, 150–157. http://arxiv.org/pdf/1202.2513v1

Qureshi, M. R. J. & Bajaber, F. (2016). *COMPARISON OF AGILE PROCESS MODELS TO CONCLUDE THE EFFECTIVENESS FOR INDUSTRIAL SOFTWARE PROJECTS*. https://www.researchgate.net/publication/311987124_COMPARISON_OF_AGILE_PROCESS_MODELS_TO_CONCLUDE_THE_EFFECTIVENESS_FOR_INDUSTRIAL_SOFTWARE_PRO-JECTS

Rowell, A. (2020). *Agile Leadership – the missing piece*. https://cybercom.com/insights/blog-posts/blogs/blogs/2020-09-23-agile-leadership---the-missing-piece

Sauter, R., Sauter, W. & Wolfig, R. (2018). *Agile Werte- und Kompetenzentwicklung*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-57305-1

Schwaber, K. & Sutherland, J. (2020). *Der Scrum Guide: Der gültige Leitfaden für Scrum: Die Spielregeln*. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-German.pdf

Sfetsos, P., Angelis, L. & Stamelos, I. (2006). Investigating the extreme programming system – An empirical study. *Empirical Software Engineering*, *11*(2), 269–301. https://doi.org/10.1007/s10664-006-6404-6

Smoczyńska, A., Pawlak, M. & Poniszewska-Marańda, A. (2019). Hybrid Agile Method for Management of Software Creation. In P. Kosiuczenko & Z. Zieliński (Hrsg.), *Advances in Intelligent Systems and Computing. Engineering Software Systems: Research and Praxis* (Bd. 830, S. 101– 115). Springer International Publishing. https://doi.org/10.1007/978-3-319-99617-2_7

Swiss Agile Research Network. (o.J.). *About*. https://www.swissagileresearchnetwork.ch/?page_id=68

SwissQ Consulting AG. (2020). *Trends & Benchmarks Report Switzerland: Where are we andwhere are we heading*. https://report20.swissq.it. https://report20.swissq.it

SwissQ Consulting AG. (2021). *Trends & Benchmarks Report: Digital Product Development*. https://report.swissq.it.

Tsyganok, I. (2016). Pair-Programming from a Beginner's Perspective. In H. Sharp & T. Hall (Hrsg.), *Lecture Notes in Business Information Processing. Agile Processes, in Software Engineering, and Extreme Programming* (Bd. 251, S. 270–277). Springer International Publishing. https://doi.org/10.1007/978-3-319-33515-5_25

Vanhala, E. & Kasurinen, J. (2019). The Role of the Customer in an Agile Project: A Multi-case Study. In S. Hyrynsalmi, M. Suoranta, A. Nguyen-Duc, P. Tyrväinen & P. Abrahamsson (Hrsg.),*Lecture Notes in Business Information Processing. Software Business* (Bd. 370, S. 208–222). Springer International Publishing. https://doi.org/10.1007/978-3-030-33742-1_17

VersionOne Inc. (2006). *Survey: The State of Agile Development* (Nr. 1). stateofagile.com. https://stateofagile.com/#ufh-i-613556022-1st-annual-state-of-agile-report/7027494

VersionOne Inc. (2007). *2nd Annual Survey: The State of Agile Development* (Nr. 2). stateofagile.com. https://stateofagile.com/#ufh-i-613555833-2nd-annual-state-of-agile-re-port/7027494

VersionOne Inc. (2008). *3rd Annual Survey: 2008: The State of Agile Development* (Nr. 3). stateofagile.com. https://stateofagile.com/#ufh-i-613555659-3rd-annual-state-of-agile-re-port/7027494

VersionOne Inc. (2009). *State of Agile Survey: The State of Agile Development* (Nr. 4). stateofagile.com. https://stateofagile.com/#ufh-i-613555629-4th-annual-state-of-agile-re-port/7027494

VersionOne Inc. (2010). *State of Agile Survey: The State of Agile Development* (Nr. 5). stateofagile.com. https://stateofagile.com/#ufh-i-613555476-5th-annual-state-of-agile-re-port/7027494

VersionOne Inc. (2011). *State of Agile Survey: The State of Agile Development* (Nr. 6). stateofagile.com. https://stateofagile.com/#ufh-i-613555398-6th-annual-state-of-agile-re-port/7027494

VersionOne Inc. (2012). *7th Annual State of Agile Development Survey* (Nr. 7). stateofagile.com. https://stateofagile.com/#ufh-i-613555113-7th-annual-state-of-agile-report/7027494

VersionOne Inc. (2013). *8th Annual State of Agile Survey* (Nr. 8). stateofagile.com. https://stateofagile.com/#ufh-i-613554825-8th-annual-state-of-agile-report/7027494

VersionOne Inc. (2014). *9th Annual State of Agile Survey* (Nr. 9). stateofagile.com. https://stateofagile.com/#ufh-i-613554519-9th-annual-state-of-agile-report/7027494

VersionOne Inc. (2015). *10th Annual State of Agile Report* (Nr. 10). stateofagile.com. https://stateofagile.com/#ufh-i-613554198-10th-annual-state-of-agile-report/7027494

VersionOne Inc. (2016). *11th Annual State of Agile Report* (Nr. 11). stateofagile.com. https://stateofagile.com/#ufh-i-613554036-11th-annual-state-of-agile-report/7027494

VersionOne Inc. (2017). *12th Annual State of Agile Report* (Nr. 12). stateofagile.com. https://stateofagile.com/#ufh-i-613553652-12th-annual-state-of-agile-report/7027494

VersionOne Inc. (2018). *13th Annual State of Agile Report* (Nr. 13). stateofagile.com. https://stateofagile.com/#ufh-i-613553418-13th-annual-state-of-agile-report/7027494

VersionOne Inc. (2019). *14th Annual State of Agile Report* (Nr. 14). stateofagile.com. https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494

vom Brocke, J., Simons, A., Niehaves, B., Reimer, K., Plattfaut, R. & Cleven, A. (2009). RECONSTRUCTING THE GIANT: ON THE IMPORTANCE OF RIGOUR IN DOCUMENTING THE LITERATURE SEARCH PROCESS. In EUROPEAN CONFERENCE ON INFORMATION SYSTEMS (Vorsitz), *ECIS 2009 PROCEEDINGS,* Verona, Italy. https://www.researchgate.net/publication/259440652_Reconstructing_the_Giant_On_the_Importance_of_Rigour_in_Documenting_the_Literature_Search_Process/stats

vom Brocke, J., Simons, A., Riemer, K., Niehaves, B., Plattfaut, R. & Cleven, A. (2015). Standingon the Shoulders of Giants: Challenges and Recommendations of Literature Search in Information Systems Research. *Communications of the Association for Information Systems*, *37.* https://doi.org/10.17705/1CAIS.03709

Webster, J. & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. In Management Information Systems Research Center, University of Minnesota (Hrsg.), *MIS Quarterly Vol. 26* (Bd. 2, S. xiii–xxiii). https://www.jstor.org/stable/4132319

Weinrich, T., Volland, A. & Muntermann, J. (2016). Herausforderungen bei der Einführung agiler Vorgehensmodelle für Finanzdienstleister – eine Fallstudie. In M. Engstler, M. Fazal-Baqaie, E.Hanser, O. Linssen, M. Mikusz & A. Volland (Hrsg.), *GI-Edition – lecture notes in informatics (LNI) Proceedings: volume P-263. Projektmanagement und Vorgehensmodelle 2016, PVM 2016: Arbeiten in hybriden Projekten: das Sowohl-als-auch von Stabilität und Dynamik: gemeinsame Tagung der Fachgruppen Projektmanagement (WI-PM) und Vorgehensmodelle (WI-VM) im Fachgebiet Wirtschaftsinformatik der Gesellschaft für Informatik e.V. ; 6. und 7. Oktober 2016 in Paderborn* (S. 79–91). Gesellschaft für Informatik. https:// dl.gi.de/handle/ 20.500.12116/591

Zykov, S. V. (2016). *Crisis Management for Software Development and Knowledge Transfer* (Bd. 61). Springer International Publishing. https://doi.org/10.1007/978-3-319-42966-3

10 Anhang

10.1 Anhang A: Datengrundlage zur Visualisierung der Methoden in Anwendung

International				
Jahr	**Report Nr.**	**Scrum**	**XP**	**Hybrid**
2006	1	40%	23%	77%
2007	2	37%	12%	23%
2008	3	49.10%	8%	22.30%
2009	4	50%	6%	24%
2010	5	58%	4%	17%
2011	6	52%	2%	14%
2012	7	54%	2%	11%
2013	8	55%	1%	11%
2014	9	56%	<1%	10%
2015	10	58%	1%	10%
2016	11	58%	<1%	10%
2017	12	56%	1%	6%
2018	13	54%	1%	10%
2019	14	58%	1%	8%
Ø		**52.51%**	**5%**	**18.09%**

Schweiz				
Jahr	**Studien Nr.**	**Scrum**	**XP**	**Hybrid**
2012	1	51%	5%	n/a
2014	2	59%	3%	n/a
2016[5]	3	52%	0%	17%
Ø		**54%**	**3%**	**17%**

Tabelle 4: Datengrundlage für die Abbildung 3 zuzüglich der Visualisierung der Daten für die Schweiz, welche im Text erwähnt wurden. (Eigene Darstellung in Anlehnung an Meier & Kropp, 2013a; 2015, 2017; VersionOne Inc., 2006; 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019)

[5] Es wurden nur die Werte von agilen Unternehmen berücksichtigt.

10.2 Anhang B: Berechnungsgrundlage Praktiken im Einsatz

		Planungs-spiel	**Kurze Re-leasezyklen**	**Metapher**	**Einfaches Design**	**Testen**	**Refactoring**	**Paarprogr.**	**Gemeins. Verantw.**	**Fortl. Inte-gration**	**40-Stunden Woche**	**Kunde vor Ort**	**Progr.-Standards**
Bezeichnung der Praktik in Report		Sprint/ Iteration Planning	Short Itera-tions	n/a	Emergent Design	Test Driven Development	Refactoring	Pair Program-ming	Collective Code Owner-ship	Continuous Integration	Sustainable Pace	Dedicated Cus-tomer/ Product Own-er[6] On Site Customer	Coding Standards
Jahr	**Report**												
2006	1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
2007	2	65%	n/a	n/a	n/a	38%	59%	24%	36%	50%	n/a	27%	n/a
2008	3	86%	n/a	n/a	n/a	49%	59%	31%	41%	65%	n/a	34%	57%
2009[7]	4	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
2010	5	83%	n/a	n/a	n/a	46%	57%	33%	36%	65%	n/a	29%	56%
2011	6	74%	n/a	n/a	n/a	38%	48%	30%	28%	54%	n/a	24%	51%
2012	7	75%	n/a	n/a	n/a	40%	48%	30%	32%	56%	n/a	n/a	57%
2013	8	75%	n/a	n/a	n/a	38%	47%	30%	29%	58%	n/a	n/a	55%
2014	9	71%	79%	n/a	n/a	34%	36%	21%	27%	50%	n/a	n/a	43%
2015	10	69%	79%	n/a	n/a	33%	37%	24%	25%	50%	n/a	n/a	44%
2016	11	90%	71%	n/a	15%	40%	52%	32%	31%	61%	31%	n/a	56%
2017	12	88%	69%	n/a	16%	35%	45%	36%	31%	54%	25%	63%	64%
2018	13	80%	67%	n/a	14%	33%	41%	34%	31%	53%	25%	57%	58%
2019	14	79%	64%	n/a	13%	30%	43%	31%	29%	55%	23%	54%	58%
Ø Anwendung		**78%**	**72%**	**n/a**	**15%**	**38%**	**48%**	**30%**	**31%**	**56%**	**26%**	**41%**	**54%**

Tabelle 5: Berechnungsgrundlage für die Abbildung 4 (Eigene Darstellung in Anlehnung an VersionOne Inc., 2006; 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019)

[6] Daten wurden nur mit «Dedicated Customer/Product Owner» berücksichtigt. Bei «Dedicated Product Owner» ist nicht klar, ob es sich dabei um eine Kundin oder einen Kunden handelt.

[7] Keine Daten in Report, die ausgewertet werden können.

		Planungsspiel	**Kurze Release-zyklen**	**Metapher**	**Einfaches Design**	**Testen**	**Refactoring**	**Paarprogram-mierung**	**Gemeinsame Verantwortlich-keit**	**Fortlaufende Integration**	**40-Stunden Woche**	**Kunde vor Ort**	**Programmier-standards**
Bezeichnung der Praktik in Report		Siehe separate Tabelle	n/a	n/a	n/a	Test Driven Development	Refactoring	Pair Programming	Collective Code Ownership	Continuous Integration	n/a	On-Site Customer	Coding Standards
Jahr	Studie												
2012	1	66%[8]	n/a	n/a	n/a	44%	51%	31%	30%	57%	n/a	23%	75%
2014	2	74%[9]	n/a	n/a	n/a	40%	61%	37%	37%	68%	n/a	38%	76%
2016[10]	3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Durchschnittliche Anwendung		**70%**	**n/a**	**n/a**	**n/a**	**42%**	**56%**	**34%**	**34%**	**63%**	**0%**	**31%**	**76%**

Tabelle 6: Praktiken im Einsatz in der Schweiz (Eigene Darstellung in Anlehnung an Meier & Kropp, 2013a; 2015; 2017)

[8] Mittelwert von 66% und 65% (aus der Tabelle mit der Berechnungsgrundlage für die XP-Praktik Planungsspiel in der Schweiz)

[9] Mittelwert von 77% und 71% (aus der Tabelle mit der Berechnungsgrundlage für die XP-Praktik Planungsspiel in der Schweiz)

[10] Datengrundlage nicht verfügbar, da Visualisierung ohne prozentuale Werte

		Planungsspiel	
Bezeichnung der Praktik in Report		Iteration Planning	User Stories
Jahr	**Studie**		
2012	1	66%	65%
2014	2	77%	71%
*2016	3	n/a	n/a
Durchschnittliche Anwendung		**72%**	**68%**

Tabelle 7: Berechnungsgrundlage für die XP-Praktik Planungsspiel in der Schweiz (Eigene Darstellung in Anlehnung an Meier & Kropp, 2013a; 2015; 2017)

10.3 Anhang C: Kodierleitfaden

Kategoriename	Definition	Ankerbeispiel	Kodierregel
40-Stunden-Woche	«Man arbeitet prinzipiell nicht mehr als 40 Stunden in der Woche. Überstunden werden nie länger als eine Woche geleistet.» (Beck, 2004, S. 54)	«9.10. 40-Hour Week […] They considered that company's organizational issues and type|volume of projects are the most important limiting factors especially for large companies.» (Sfetsos et al., 2006, S. 289)	Faktoren, welche der XP-Praktik 40-Stunden Woche zugeordnet werden können. Dabei werden auch englische Schreibweisen wie 40-Hour Week/Sustainable Pace berücksichtigt.
Einfaches Design	«Das System sollte zu jedem Zeitpunkt so ein-fach wie möglich strukturiert sein. Lösen Sie unnötig komplexe Strukturen auf, sobald Sie diese entdecken.» (Beck, 2004, S. 54)	«9.5. Simple Design […] Developers named time restrictions in development process and the shift of priorities in development process, as twoother limiting factors for practice implementation in large companies.» (Sfetsos et al., 2006, S. 286–287)	Faktoren, welche der XP-Praktik Einfaches Design zugeordnet werden können. Dabei werden auch englische Schreibweisen wie Simplicity berücksichtigt.
Fortlaufende Integration	«Das System wird mehrmals täglich integriert und erstellt und zwar immer dann, wenn eine Aufgabe erledigt worden ist.» (Beck, 2004, S. 54)	«Continuous integration was found to be challenging where enterprise software products required time consuming regression testing and elaborate code release processes.» (Bass, 2012, S. 1)	Faktoren, welche der XP-Praktik Fortlaufende Integration zugeordnet werden können. Dabei werden auch englische Schreibweisen wie Continuous Integration berücksichtigt.

Gemeinsame Verantwortlichkeit	«Jeder kann jederzeit beliebigen Code im System ändern.» (Beck, 2004, S. 54)	«Collective code ownership doesn't work without trust. You have to give up control. You have to give up the idea of exclusive ownership. You have to believe that ‹we› produce vastly better results than ‹I›. […] The hardest hurdle to clear is the ego problem.» (Auer & Miller, 2002, S. 224)	Faktoren, welche der XP-Praktik Gemeinsame Verantwortlichkeit zugeordnet werden können. Dabei werden auch englische Schreibweisen wie Collective (Code) Ownership berücksichtigt.
hybride Vorgehensmodelle	Basierend auf den jährlichen State of Agile Reports von (VersionOne Inc.) werden in der Praxis hybride Vorgehen gewählt.	«We categorized and analyzed the processes used and found hybrid approaches to be widely used in practice. Our study revealed that hybrid approaches have become mainstream and are used by companies regardless of company size and industry sector.» (Kuhrmann et al., 2017, S. 38–39)	Faktoren, welche der Kombination von XP mit einer anderen Vorgehensweise zugeordnet werden können.
Kunde vor Ort	«Dem Team sollte ein echter, lebender Benutzer angehören, der während der gesamten Arbeitszeit zur Beantwortung von Fragen zur Verfügung steht.» (Beck, 2004, S. 54)	«XP fully depends upon customer thatmay become a risk to fail a project.» (Mushtaq & Qureshi, 2012, S. 40)	Faktoren, welche der XP-Praktik Kunde vor Ort zugeordnet werden können. Dabei werden auch englische Schreibweisen wie Customer on Site berücksichtigt.
kurze Releasezyklen	«Gehen Sie mit einem einfachen System schnell in Produktion und bringen Sie dann innerhalb sehr kurzer Zeit die nächste Version heraus.» (Beck, 2004, S. 53)	«9.9. Short Release Cycles […] The need for experienced and skilled personnel is one of the needing factors proposed by both managers and developers, especially for small companies.» (Sfetsos et al., 2006, S. 288)	Faktoren, welche der XP-Praktik kurze Releasezyklen werden können. Dabei werden auch englische Schreibweisen wie Small Releases berücksichtigt.

Metapher	«Sämtliche Entwicklungen werden an einer einfachen gemeinsamen Metapher ausgerichtet, die die Funktionsweise des gesamten Systems veranschaulicht.» (Beck, 2004, S. 53)	«XP ist bisher in seiner Konsequenz unzureichend dokumentiert und das Konzept der Systemmetapher ist etwas nebulös.» (Keller, 2019, S. 38)	Faktoren, welche der XP-Praktik Metapher werden können. Dabei werden auch englische Schreibweisen wie Metaphor berücksichtigt.
Paarprogrammierung	«Der gesamte Produktionscode wird von zwei Programmierern geschrieben, die gemeinsam an einem Rechner sitzen.» (Beck, 2004, S. 54)	«Managers view programmers as a scarce resource, and are reluctant to ‹waste› such by doubling the number ofpeople needed to develop a piece of code.» (Kunwar, 2019, S. 58)	Faktoren, welche der XP-Praktik Paarprogrammierung werden können. Dabei werden auch englische Schreibweisen wie Pair Programming berücksichtigt.
Planungsspiel	«Legen Sie den Umfang der nächsten Versionrasch fest, indem Sie Geschäftsprioritäten mittechnischen Aufwandsschätzungen kombinieren. Wenn die Realität den Plan einholt, dann muss der Plan aktualisiert werden.» (Beck, 2004, S. 53)	«The lack of consideration of non-functional requirements from the standpointof the business.» (Kunwar, 2019, S. 57)	Faktoren, welche der XP-Praktik Planungsspiel werden können. Dabei werden auch englische Schreibweisen wie Planning Game berücksichtigt.
Programmierstandards	«Programmierer schreiben sämtlichen Code entsprechend Regeln, die die Kommunikationmithilfe des Codes erleichtern.» (Beck, 2004, S. 54)	«Ineffizientes Handeln und hohe Aufwände bei dogmatischer Verwendung der Programmierrichtlinien möglich.» (Proba, 2021, S. 289)	Faktoren, welche der XP-Praktik Programmierstandards werden können. Dabei werden auch englische Schreibweisen wie Coding Standards berücksichtigt.
Refactoring	«Die Programmierer strukturieren das System neu, ohne sein Verhalten zu ändern, um Redundanzen zu entfernen, die Kommunikation zu verbessern, das System zu vereinfachen oder flexibler zu gestalten.» (Beck, 2004, S. 54)	«The places where XP is still exposed with respect to being high-discipline are coding standards, acceptance tests, and aggressive refactoring. Of those, aggressive refactoring probably will remain the most difficult, because it requires consistency,	Faktoren, welche der XP-Praktik Refactoring werden können.

		energy, and courage, and no mechanisms in the methodology reinforce it.» (Cockburn, 2002, S. 141)	
Testen	«Die Programmierer schreiben fortwährend Komponententests, die fehlerfrei ausgeführt werden müssen, damit die Entwicklung voranschreiten kann. Kunden schreiben Tests, um zu zeigen, dass Leistungsmerkmale fertig gestellt sind.» (Beck, 2004, S. 54)	«TDD is a disciplined approach requires some special skills (like writing test cases which is duty of testers usually) that programmers feel difficult to practice.» (Anwer et al., 2017, S. 4)	Faktoren, welche der XP-Praktik Testen werden können. Dabei werden auch englische Schreibweisen wie Test Driven Development (TDD) berücksichtigt.
XP-Methode	*Kein Literaturverweis.* *Diese Kategorie ist für Ausprägungen gedacht, nicht einer einzelnen Praktik, sondern der Methode selbst zugeordnet werden können.*	«XP lacks in project management practice and depends upon customer support that may become a risk of project failure.» (Ibrahim et al., 2020, S. 167)	Faktoren, welche XP als Methode zugeordnet werden können. Diese Faktoren gelten Praktikübergreifend

10.4 Anhang D: Auswertung Literatur (Kategoriensystem)

Quelle	Kategorie	Ausprägung	Segment
Cockburn, 2002, S. 141	Refactoring	Durchhaltever-mögen	The places where XP is still exposed with respect to being high-discipline are coding standards, acceptance tests, and aggressive refactoring. Of those, aggressive refactoring probably will remain the most difficult, because it requires consistency, energy, and courage, and no mechanisms in the methodology reinforce it.
Cockburn, 2002, S. 142	XP-Methode	Skalierbarkeit	Many people exclaim: «XP doesn't scale!» At this point, you should review, if you don't recall it, the graphs of problem size versusteam size in the last section. A well-structured, 10-programmer team using XP properly may be able to solve a larger problem than a 30-person team using a larger methodology. In fact, on the first official XP project, an 8-person XP team delivered in one year what the previous, 26-person team had failed to deliver in the previous year. So be aware of what the statement «XP doesn't scale» really means. XP scales quite well in problem size (up to its limit); at the same time, it does not scale in staff size.
Cockburn, 2002, S. 142	XP-Methode	Implizites Wissen	XP, as written, has been demonstrated on projects with up to twelve programmers and four on-site customers. It may have trouble with larger teams due to its reliance on tacit knowledge. It is difficult to build extensive tacit knowledge without good osmotic communication, and that is hard to do with more people than conveniently fit in a room. A larger project team trying XP will have to adjust the teaming structures, interfaces, and use of documentation to accommodate the greater coordination needs of the larger group and the thinner communication lines.
Pelrine et al., 2000, S. 1	XP-Methode	Durchhaltever-mögen	Many of the techniques and methods of XP are highly intuitive and, one would think, easy to learn. Nevertheless, XP is a radical approach to software development, and the discipline required to pro-perly learn and apply the techniques and methods is itself extreme. We maintain that the use of a coach is necessary to successfully learn the methods of XP, and to guarantee the success of the development process.

Pelrine et al., 2000, S. 3	XP-Methode	Mindset	Throwing away code is a big thing to request from a developer. We have mentioned above the difficulties in the learning process in cases where the corporate culture stigmatizes making mistakes, or (in our case) «building one to throw away». This is not the only problem, and not the biggest problem. A normal developer becomes attached to his code, to his product, becomes possessive, and extremely reluctant to let it go.
Pelrine et al., 2000, S. 4	XP-Methode	Interpretation	«On the other hand, I think that programmers who aren't as good as Beck and his colleagues will view this book as a license to be reckless. ‹Analyze requirements? Bah – we'll just start coding.› ‹Design the class hierarchy to be flexible? Bah – we'll just through some code down, then refactor it when we need to.› This might be productive for someone who has Becks insight into software architecture, and who has internalized good practices to the point where they are instinctual. Most of the programmers I know, however, would quickly find themselves in a morass of inconsistent class interfaces, passed-through parameters, and switch statements that don't quite cover all possible cases. I realize that XP practices such as prophylactic testing are intended to prevent this, but these have to be backed up by experienced judgement to be effective, and it is just such judgement that less reflective programmers lack.»
Pelrine et al., 2000, S. 5	XP-Methode	Kompetenzen	Certain characteristics are required of the learner – not only for XP, but especially there. Especially in Pair Programming «there must be a great deal of mutual respect, tolerance, understanding [...]» (see [7]). Certainly not every developer is «blessed with the kind of strong communication and social skills» (see [4]). It is part ofthe learning process to build up these social skills as much as it is to teach the theoretical and practical aspects of a method.
Beck, Fowler & Kohnke, 2001, S. 126	Kunde vor Ort	Verantwortung	Sometimes customers simply refuse to make decisions. They won't pick the stories for an iteration. They won't specify acceptance tests. They won't answer little questions about scope. Extreme Programming cannot work without a customer making decisions.

Auer & Miller, 2002, S. 38	Kunde vor Ort	Verfügbarkeit	If you want to move at full speed, you can't afford not to have a customer available to answer questions for the development team immediately, whenever those questions come up. We don't know of any research that's been done in this area, but we've been on many projects. When developers have to wait for answers, two things tend to happen: 1. The project slows way down or grinds to a halt while developers wait. 2. Developers get tired of waiting and guess at what the requirements are. The first costs money by delaying the realization of value from the project. The second costs money because developers often guess wrong and have to redo lots of work, producing still more delay.
Auer & Miller, 2002, S. 39	Kunde vor Ort	Interpretation	It will often be difficult to have a customer on site or available all of the time, but that is no reason to ignore the problem. Even if we do all the other XP practices well, without direct customer input we're still guessing to fill the holes.
Auer & Miller, 2002, S. 45	XP-Methode	Akzeptanz	XP goes against deeply ingrained programmer habits by forcing these folks to interactwith people most of the time. Developers resist.
Auer & Miller, 2002, S. 223	Gemeinsame Verantwortlichkeit	Verständnis	Collective code ownership isn't the norm in software development. Developers tend toresist it, usually because they don't understand what it means, so they assume it's like no code ownership at all, or they fear giving up control of «their» code.
Auer & Miller, 2002, S. 224	Gemeinsame Verantwortlichkeit	Mindset	Collective code ownership doesn't work without trust. You have to give up control. You have to give up the idea of exclusive ownership. You have to believe that «we» produce vastly better results than «I». […] The hardest hurdle to clear is the ego problem.
Sfetsos et al., 2006, S. 272	Hybride Vorgehensmodelle	Adaption	Results have shown that none of the companies adopted agile practices in a «pure» form. Project teams adopted selectively some practices and tailored some others to operate in their working environment.
Sfetsos et al., 2006, S. 282	Planungsspiel	Organisation	Company’s organizational issues (mixed and well defined processes, distributed work across multiple teams, communication and coordination problems)

Sfetsos et al., 2006, S. 285	Paarpro-grammierung	Kultur	Cultural problems highlighted by differences between agile and traditional teams, and problems caused by distribution of work across multiple teams in large and complex projects are the most significant limiting factors for large companies.
Sfetsos et al., 2006, S. 285	Paarpro-grammierung	Verteilte Teams	Cultural problems highlighted by differences between agile and traditional teams, and problems caused by distribution of work across multiple teams in large and complex projects are the most significant limiting factors for large companies.
Sfetsos et al., 2006, S. 285	Paarpro-grammierung	Verteilte Teams	Small companies also experienced problems associated with distributed teams, but human problems (unpleasant conditions or relations among staff) and the need for experienced developers (skill-factor: efficiency in modifying code) are considered as strong limiting factors.
Sfetsos et al., 2006, S. 285	Paarpro-grammierung	Kompetenzen	Small companies also experienced problems associated with distributed teams, but human problems (unpleasant conditions or relations among staff) and the need for experienced developers (skill-factor: efficiency in modifying code) are considered as strong limiting factors.
Sfetsos et al., 2006, S. 286	Testen	Kompetenzen	The need for skilled developers-customers and the lack of automated-effective testing tools were reported by managers as limiting factors for both large and small companies.
Sfetsos et al., 2006, S. 286–287	Einfaches Design	Priorisierung	Developers named time restrictions in development process and the shift of priorities in development process, as two other limiting factors for practice implementation in large companies.
Sfetsos et al., 2006, S. 286–287	Einfaches Design	Zeit	Developers named time restrictions in development process and the shift of priorities in development process, as two other limiting factors for practice implementation in large companies.
Sfetsos et al., 2006, S. 286	Refactoring	Anleitungen	In interviews, managers and developers from both large and small companies complained about the lack of effective refactoring tools and that existing XP theory does not provide detailed guidelines on how to accomplish successful refactoring
Sfetsos et al., 2006, S. 286	Einfaches Design	Anleitungen	Limiting factors are the same for large and small companies as DA results have shown, namely the lack of theoretical guidance of the practice and the lack of documentation in distributed development.
Sfetsos et al., 2006, S. 286	Einfaches Design	Dokumentatio-nen	Limiting factors are the same for large and small companies as DA results have shown, namely the lack of theoretical guidance of the practice and the lack of documentation in distributed development.

Sfetsos et al., 2006, S. 288	Kunde vor Ort	Verfügbarkeit	For both large and small companies, a business representative the «Customer», being constantly near the team, helps in: - Requirements' understanding, - Decisions' taking, - Development process (increasing team's experience), and - Projects' success. However, in reality, such access to customers is often difficult.
Sfetsos et al., 2006, S. 286	Kunde vor Ort	Anwesenheit	Besides, they also assured that the full use of the practice, as described in theory, was difficult to achieve and only some large projects provided a full time on-site customer
Sfetsos et al., 2006, S. 286	kurze Re-leasezyklen	Kompetenzen	The need for experienced and skilled personnel is one of the needing factors proposed by both managers and developers, especially for small companies.
Sfetsos et al., 2006, S. 289	40-Stunden Woche	Organisation	They considered that company's organizational issues and type|volume of projects are the most important limiting factors especially for large companies.
Sfetsos et al., 2006, S. 289	40-Stunden Woche	Projektart und -grösse	They considered that company's organizational issues and type|volume of projects are the most important limiting factors especially for large companies.
Sfetsos et al., 2006, S. 289	Programmier standards	Erfahrung	Large company's organizational issues and the lack of experience for small companies were suggested as limiting factors
Sfetsos et al., 2006, S. 289	Programmier standards	Organisation	Large company's organizational issues and the lack of experience for small companies were suggested as limiting factors
Sfetsos et al., 2006, S. 290	Metapher	Erfahrung	However, they also verified that metaphor was the hardest to apply practice, especially in large companies. They defined as limiting factors, for large and small companies, the cultural problems (traditional agile teams), the lack of experience, and the unclear practice theory.
Sfetsos et al., 2006, S. 290	Metapher	Unklarheit	However, they also verified that metaphor was the hardest to apply practice, especially in large companies. They defined as limiting factors, for large and small companies, the cultural problems (traditional agile teams), the lack of experience, and the unclear practice theory.

Sfetsos et al., 2006, S. 290	Metapher	Kultur	However, they also verified that metaphor was the hardest to apply practice, especially in large companies. They defined as limiting factors, for large and small companies, the cultural problems (traditional agile teams), the lack of experience, and the unclear practice theory.
Conboy et al., 2011, S. 49–51	XP-Methode	Transparenz	In all 17 companies, developers feared that the agile process could bring their own deficiencies to light. Interviewees outlined how procedures such as stand-up meetings, on-site customers, and the use of storyboards and whiteboards made developer shortcomings visible to the rest of the team because these practices require direct and constant communication and collaboration. For example, storyboards track the status of user stories and make a developer's lack of progress obvious. White-boards, which agile teams use to communicate design issues, can highlight developers' technical and communication challenges because they must regularly present their ideas in front of their peers. In addition, continuous integration and automated testing mean that developers can't hide poor, low-quality code. [...] To address this challenge, developers need an environment where they feel safe to expose their weaknesses. [...] Developers should also know that they can get help to improve their skills. In at least nine cases, pair programming teamed weaker developers with more experienced developers; thus, joint responsibility dissolved the public exposure of any potential weaknesses.
Conboy et al., 2011, S. 51	XP-Methode	Kompetenzen	In all 17 companies, agile environments seem to blur the boundaries among developers' roles and require competence in a broad range of skills, as opposed to specialization in one. To be a successful agile [developer] you need to be a coder, a tester, an architect, a customer, a quality assurance expert, and a multitude of other things software-related. (manager, company M) As a manager in company D described, rather than being a «jack of all trades, master ofnone», a developer in an agile team must be a «master of all trades». This multifaceted skill set caused nu-merous problems. First, almost all project managers had difficulty finding developers with all the necessary agile skills, either externally or in their organization. Training was also more difficult. In four cases, management sent its entire team to all available training courses, incurring high expense.

Conboy et al., 2011, S. 52	XP-Methode	Kompetenzen	Agile development's customer-facing aspect also caused significant problems in eight companies. It was clear that with certain people, «you should never, ever put them in front of a client» (director, company M). In fact, «being a good communicator is one thing. Knowing what not to communicate is much more important» (manager, company O). Managers cited examples of developers revealing politically sensitive and confidential information to customers regarding contracts, salaries, and opinions regarding development teams' weaknesses.
Conboy et al., 2011, S. 54	XP-Methode	Motivation	Five companies encountered developers who lacked motivation to use agile methods. This was more prominent in companies that adopted agile methods top-down. A manager in company G observed, «sometimes they have the competence but are not convinced it [agile development] will work». Many respondents perceived process innovations such as adopting agile methods as overly onerous, complex, and time-consuming.
Conboy et al., 2011, S. 54	XP-Methode	Akzeptanz	In company L, the manager cited anxiety over losing power as a «problemamong some managers».
Marquardt, 2011, S. 99	XP-Methode	Akzeptanz	Extreme Programming verändert die Zuständigkeiten und Rollen im Vergleich zu anderen Vorgehensmodellen und stösst damit in seiner Vollausprägung noch auf Skepsis. Oftwird XP dann nur von Entwicklerseite verfolgt. Dieser Ansatz erzeugt leider gelegentlich mehr Konflikte als er löst, da die wichtige Kommunikation und Rückkopplungsschleife zum Kunden nicht nur offen bleiben, sondern durch gegenseitiges Unverständnis noch zusätzlich behindert werden kann.
Pelrine, 2011, S. 27	XP-Methode	Mindset	Ultimately, models are only as good as the people applying them. Too many teams havecome to regard Agile as something like a cookery recipe – follow this set of instructions and procedures for a tasty result. But in software development, as in cooking, what you get out is not simply the sum of what you put in. We need to develop an understanding of what makes Agile work, and indeed what makes it fail. This understanding is a prerequisite for sustaining and scaling Agile efforts. Acknowledging that Agile is not working in a particular situation is an inherent part of Agile practice, but it's one that's often ignored.

Bass, 2012, S. 1	Fortlaufende Integration	Aufwand	Continuous integration was found to be challenging where enterprise software products required time consuming regression testing and elaborate code release processes.
Bass, 2012, S. 6	Fortlaufende Integration	Aufwand	Continuous integration is challenging in enterprise software development projects because of the size, scale and complexity of the system under development. According to the Director of Engineering at Company E, «regression tests take too long to do it everysprint. So we don't run a regression suite, it is like three and a half days to run a full regression suite so we don't run that [every Sprint]».
Mushtaq & Qureshi, 2012, S. 40	XP-Methode	Management-praktiken	XP lacks in project management practices.
Mushtaq & Qureshi, 2012, S. 40	Kunde vor Ort	Abhängigkeit	XP fully depends upon customer that may become a risk to fail a project.
Mushtaq & Qureshi, 2012, S. 40	XP-Methode	Skalierbarkeit	XP is not suitable for medium and large scale projects.
Qureshi, 2012, S. 151	XP-Methode	Skalierbarkeit	recommends less documentation making it suitable only for small projects and limiting the opportunities and advantages of reusability;
Qureshi, 2012, S. 151	Testen	Kosten	Test driven approach is more time and cost consuming
Qureshi, 2012, S. 151	Testen	Aufwand	Test driven approach is more time and cost consuming
Qureshi, 2012, S. 151	Kunde vor Ort	Abhängigkeit	teams fully depend upon customer that may sometime become a cause for the failure of projects

Krodel, 2013, S. 36	Hybride Vorgehensmodelle	Ergänzung (Scrum/XP)	Einige dieser Techniken, wie das Streben nach kurzen Release-Zyklen, regelmässiges Testen und Kundeneinbindung sind in ähnlicher Form in Scrum bereits durch das Prozess- und Rollenmodell umgesetzt. Andere hingegen sind geeignet, Scrum inhaltlich weiter zu präzisieren. Beispielsweise empfiehlt sich für Scrum-Projekte die Einhaltung einer 40-Stunden-Woche, um Demotivation und Erschöpfung zu vermeiden. Auch die teamorientierten Aspekte der XP-Techniken, wie das Pair Programming und ein gemeinsames Verantwortungsgefühl für das Projekt, können in Scrum integriert werden. Somit steuert XP eher mit «weichen Faktoren» Ergänzungspotenzial für den agilen Prozess Scrum bei. Hinsichtlich des Prozess- und Rollenmodells beinhaltet Scrum bereits wesentliche Elemente von XP. Hierin liegt begründet, dass Scrum XP zunehmend verdrängt und nur die Techniken von XP erhalten bleiben, auf die sich die meisten Veröffentlichungen beschränken.
Aichele & Schönberger, 2016, S. 86	Kunde vor Ort	Beteiligung	An der Vorgehensweise des XP bestehen zahlreiche Kritikpunkte, die sich auf die fehlende Entwurfsplanung sowie die ständige Beteiligung des Kunden beziehen.
Aichele & Schönberger, 2016, S. 86	XP-Methode	Wirtschaftlichkeit	Balzert formuliert weiterhin den Einwand, dass der Einsatz des Extreme Programming vor allem in kleineren Projekten negative Auswirkungen auf die Wirtschaftlichkeit des gesamten Projektes hervorbringt
Fausten, 2016, S. 23	Refactoring	Akzeptanz	Eine Architektur-Refaktorisierung beinhaltet im Normalfall eine Code-Refaktorisierung. Der Kunde und das Management sehen keinen direkten Fortschritt, deshalb werden diese schnell unzufrieden.
Qureshi & Bajaber, 2016, S. 5120	XP-Methode	Dokumentation	Main limitations of XP are poor documentation
Qureshi & Bajaber, 2016, S. 5120	XP-Methode	Verteilte Teams	It is also inappropriate for distributed teams, reuse and subcontracting and development of average and complex software.
Qureshi & Bajaber, 2016, S. 5120	Kunde vor Ort	Abhängigkeit	The success of XP mainly depends on the support of its stakeholders.

Tsyganok, 2016, S. 271	Paarpro-grammierung	Infrastruktur	Physical challenges are concerned with physical comfort. They can take the form of unsuitable equipment or inadequate personal space and can result in poor posture and discomfort
Tsyganok, 2016, S. 271	Paarpro-grammierung	Wohlbefinden	Physical challenges are concerned with physical comfort. They can take the form of unsuitable equipment or inadequate personal space and can result in poor posture and discomfort
Tsyganok, 2016, S. 271	Paarpro-grammierung	Unterbrechung	Session-management challenges are interruptions to the session caused by developers without consideration of the schedule of their pairing partner.
Tsyganok, 2016, S. 271	Paarpro-grammierung	Verfügbarkeit	For junior/senior pairs, the biggest challenge is frequent unavailability of senior developers.
Tsyganok, 2016, S. 271	Paarpro-grammierung	Kompetenzen	Social challenges are less tangible and are therefore, the hardest to deal with. They are dependent on the personality traits and emotional intelligence of both partners. Physical and session management challenges can be more easily resolved if the social challenges are eliminated first.
Weinrich et al., 2016, S. 87	XP-Methode	Mindset	Weiterhin besteht die Herausforderung des Schaffens eines Bewusstseins über agile Vorgehens-modelle auch auf der Ebene der Projektleiter: «Also ich sehe da nicht so einen riesen Unterschied [zwischen klassisch und agil].» Ein möglicher Grund hierfür ist, dass die Rahmenbedingungen des Unternehmens nur bedingt für agile Vorgehensmodelle geeignet sind. Ein weiterer Interviewpartner bestätigt: «So wie wir es hier bei uns im Hause tun [agiles Vorgehen], kann man es auch mit her-kömmlichen Projekt-Management-Methoden machen.»
Zykov, 2016, S. 67	XP-Methode	Kommunikation	Among XP challenges are: dependence on oral communication, client's obligatory team membership, interdependence of the practices (some of which, such as pair programming, are human factor-dependent)
Zykov, 2016, S. 67	XP-Methode	Abhängigkeit	Among XP challenges are: dependence on oral communication, client's obligatory team membership, interdependence of the practices (some of which, such as pair programming, are human factor-dependent)
Zykov, 2016, S. 67	Kunde vor Ort	Beteiligung	Among XP challenges are: dependence on oral communication, client's obligatory team membership

Anwer et al., 2017, S. 4	Testen	Kompetenzen	TDD is a disciplined approach requires some special skills (like writing test cases which is duty of testers usually) that programmers feel difficult to practice
Anwer et al., 2017, S. 4	Testen	Aufwand	Sometimes, TDD become more time consuming because of repeated test failure.
Hanschke, 2017, S. 31	Hybride Vorgehensmodelle	Ergänzung (Scrum/XP)	Aus diesen Techniken werden bei Scrum und anderen Ansätzen Anleihen mit ggf. unterschiedlicher Namensgebung genommen.
Hekkala et al., 2017, S. 5876	XP-Methode	Akzeptanz	There were severe managerial challenges already at the beginning; the choices made by the management group did not resonate with the ideas of project members working in the project and their different backgrounds. In addition to this, the IT managers did not support the more group intensive approach, as they wanted to control, for example, technological choices. The management group and software developers, thus, had very different conceptions about what self-organizing teams and distributed leadership meant in practice. Some people on the project level made assumptions that agile means anything goes, whereas the management group still needs status reports, even within sprints.
Kuhrmann et al., 2017, S. 36	Hybride Vorgehensmodelle	Adaption	We asked the participants how their particular combination of the different development approaches was developed, and a majority of 83.9% (Table 10) stated that the development approach emerges from experience and learning from past projects.
Kuhrmann et al., 2017, S. 37	XP-Methode	Entscheidungsgewalt	Finally, we were interested in the decision-making process, i.e., how are development approaches selected and by whom. The results summarized in Table 11 show that morethan the half of the participants (53.6%) state selecting the methods and practices in the project on demand. Furthermore, 35.7% state that the project-specific tailoring follows defined rules and that a project manager carries out the tailoring in the beginning of a project (33.9%). Hence, this shows that standards exist and that they are (at least initially) applied to a project. However, during the project, the configuration of methods and practices might change.
Kuhrmann et al., 2017, S. 38–39	Hybride Vorgehensmo-	Mainstream	We categorized and analyzed the processes used and found hybrid approaches to be widely used in practice. Our study revealed that hybrid approaches have become mainstream and are used by com-

	delle		panies regardless of company size and industry sector.
Kuhrmann et al., 2017, S. 39	Hybride Vorgehensmodelle	Adaption	Hybrid approaches used in practice today emerge from pragmatic process selection and evolve over time.
Nanthaamornphong & Carver, 2017, S. 356	Refactoring	Abhängigkeit (UnitTests)	Because the implementation is often driven by the tests rather than by a requirements specification, the respondents indicated that the process of refactoring is difficult if the unit tests are not well designed. However, software is nearly impossible to refactor without thorough unit tests that are readily available
Nanthaamornphong & Carver, 2017, S. 356	Refactoring	Abhängigkeit (Architekturdesign)	Although testing is essential to refactoring, poor architecture design makes refactoring difficult or impossible. Furthermore, the respondents indicated that if the initial architecture or system design is poor, it is occasionally necessary to redesign the system and revisiting large portions of it.
Nanthaamornphong & Carver, 2017, S. 356	Refactoring	Kompetenzen	One common problem is developers' failure to understand refactoring and recognize its benefits. There is lack of knowledge of how to refactor (at all and/or efficiently) or why refactoring is conducted (because it does not add functionality). Refactoring also may require knowledge of advanced programming techniques, e.g., template code and functional programming.
Nanthaamornphong & Carver, 2017, S. 356	Refactoring	Legacy Code	Refactoring is difficult when the respondents are working with a diverse legacy code base. More specifically, for legacy code, the developers may not actually have adequate tests to ensure that a refactoring does not cause problems.
Bibik, 2018, S. 8	XP-Methode	Produktivität	Despite the claims that XP is more productive in XP-oriented resources, development can be slower than other methodologies. I believe it can be more productive only insome specific cases.
Bibik, 2018, S. 9	Kunde vor Ort	Verfügbarkeit	Risk of lack of customer availability: Since customers can't always commit to becoming part of the team, it can become an issue for XP methodology
Hofert, 2018, S. 202	Paarprogrammierung	Kultur	Ohne klare Regeln besteht die Gefahr, dass im Pairing Konfikte entstehen. Es muss eine gute und faire Kultur der Kooperation bereits geben. Fehler müssen normal sein, der Austausch auf Augenhöhe üblich.

Ott, 2018, S. 49	Testen	Aufwand	Eine komplette Testabdeckung für den gesamten Quellcode zu erreichen, ist ein grösseres Unterfangen. Je früher damit jedoch gestartet wird, desto schneller können die Vorteile genutzt werden und desto weniger Zeit muss investiert werden, um alten Code mit Tests abzudecken.
Ott, 2018, S. 49	Testen	Aufwand	Sollten die Tests nicht automatisiert sein, entsteht ein riesiger Aufwand, jeden Build manuell zu testen oder dies wird schlicht ignoriert und Bugs werden mit neuen Versionen deployt.
Alpar et al., 2019, S. 371	XP-Methode	Projektkontrolle	Extreme Programming ist eher auf die Entwicklung kleinerer Software ausgelegt. Es geht von relativ kurzen Entwicklungszeiten aus und verfolgt ein inkrementelles Vorgehen. Nachteilig ist bei dieser Art der Softwareentwicklung die mangelhafte Projektkontrolle.
Keller, 2019, S. 34	Hybride Vorgehensmodelle	Adaption	XP wurde von Beck, Cunningham und Jeffries entworfen [Bec2000]. Es beinhaltet einige Elemente, welche auch in allen anderen Vorgehensmodellen sinnvoll eingesetzt werden können.
Keller, 2019, S. 38	Metapher	Unklarheit	XP ist bisher in seiner Konsequenz unzureichend dokumentiert und das Konzept der Systemmetapher ist etwas nebulös.
Kuhrmann et al., 2019, S. 20	Hybride Vorgehensmodelle	Adaption	AS THERE IS no one-size-fits-all software development approach, teams and organizations use different approaches to address the manifold challenges of software development projects. They rarely follow the pure approach by implementing a process by the book
Kuhrmann et al., 2019, S. 26	Hybride Vorgehensmodelle	Adaption	On the basis of earlier studies, we expected a combined use, so we asked the participants how a particular combination of approaches was developed. Only 19.6% of the responses stated that the used approach was an outcome of a planned process improvement program. For 83.9% of the responses, the majority of the approaches emerged from experiences collected throughout time.
Kuhrmann et al., 2019, S. 26	Hybride Vorgehensmodelle	Adaption	That is, the actual approach evolved and was usually not subject to a controlled development – a practice in line with the core principle from the «Agile Manifesto» often referred to as inspect and adapt. In this context, 23.2% of the participants also stated that they adapted the approach in response to a specific situation, which is in line withthe 27.5% who selected their development approach individually.

Kuhrmann et al., 2019, S. 27	Hybride Vorgehensmodelle	Neophobie	Improving the stability of the team, the project type, product lifecycle management, integration of high-level waterfall-like and low-level agile approaches, the lack of readiness of customers (e.g., regarding contracting or pricing), the lack of management buy-in, and business people who are scared of new things
Kuhrmann et al., 2019, S. 27	Hybride Vorgehensmodelle	Evolution	Constant evolution, a stepwise company transition toward agile development, an increasing number of software parts in complex systems, and the requirements of day-to-day life («It came up naturally» and «We do what works»)
Kuhrmann et al., 2019, S. 27	Hybride Vorgehensmodelle	Flexibilität	Flexibility of teams and resources, selecting the best-fitting approach, fast product feature evolution, meeting deadlines, and the better handling of volatile requirements
Kuhrmann et al., 2019, S. 27	Hybride Vorgehensmodelle	Einschränkungen	The lack of readiness and understanding of agile development, customer satisfaction, and client-domain requirements
Kuhrmann et al., 2019, S. 27	Hybride Vorgehensmodelle	Einschränkungen	Company size, keeping the existing business running, company history, and company philosophy
Kuhrmann et al., 2019, S. 27	Hybride Vorgehensmodelle	Einschränkungen	The business context and target domain requirements, e.g., management, standards, and different target domains to be addressed
Kunwar, 2019, S. 55	Kunde vor Ort	Verfügbarkeit	It is very difficult to find the real representative of customer business
Kunwar, 2019, S. 55	Kunde vor Ort	Abhängigkeit	Single person (onsite customer) is responsible for making decisions about the business.
Kunwar, 2019, S. 55	Kunde vor Ort	Abhängigkeit	High chances of unclear and defective requirement collected from a single person.

Kunwar, 2019, S. 56	Kunde vor Ort	Verfügbarkeit	Limiting factors of onsite customer - Full time availability. - Inadequate domain knowledge. - Decision making authority on single people.
Kunwar, 2019, S. 56	Kunde vor Ort	Domänenwissen	Limiting factors of onsite customer - Full time availability. - Inadequate domain knowledge. - Decision making authority on single people.
Kunwar, 2019, S. 56	Kunde vor Ort	Abhängigkeit	Limiting factors of onsite customer - Full time availability. - Inadequate domain knowledge. - Decision making authority on single people.
Kunwar, 2019, S. 56	Paarpro-grammierung	Kompetenzen	Limiting factors of Pair Programming - Differences in programming and communication skills. - Antisocial or anti personalities. - Perception of cost and time. - Common schedule and agreement. - Discourage in pairing.
Kunwar, 2019, S. 56	Paarpro-grammierung	Kompetenzen	Limiting factors of Pair Programming - Differences in programming and communication skills. - Antisocial or anti personalities. - Perception of cost and time. - Common schedule and agreement. - Discourage in pairing.

Kunwar, 2019, S. 56	Paarpro-grammierung	Kosten	Limiting factors of Pair Programming - Differences in programming and communication skills. - Antisocial or anti personalities. - Perception of cost and time. - Common schedule and agreement. - Discourage in pairing.
Kunwar, 2019, S. 56	Paarpro-grammierung	Verfügbarkeit	Limiting factors of Pair Programming - Differences in programming and communication skills. - Antisocial or anti personalities. - Perception of cost and time. - Common schedule and agreement. - Discourage in pairing.
Kunwar, 2019, S. 56	Paarpro-grammierung	Wohlbefinden	Limiting factors of Pair Programming - Differences in programming and communication skills. - Antisocial or anti personalities. - Perception of cost and time. - Common schedule and agreement. - Discourage in pairing.
Kunwar, 2019, S. 57	Planungs-spiel	Anforderungs-management	The lack of consideration of non-functional requirements from the standpoint of the business.
Kunwar, 2019, S. 58	Paarpro-grammierung	Produktivität	Two developers working together cannot equal the productivity of the same two developers working in parallel.
Kunwar, 2019, S. 58	Paarpro-grammierung	Kosten	It has been statistically shown that paired programming costs approximately 15% more time than traditional programming

Kunwar, 2019, S. 58	Paarprogrammierung	Kompetenzen	Effective paired programming is difficult to achieve and requires a careful cultivation of personalities within the development team.
Kunwar, 2019, S. 58	Paarprogrammierung	Managementakzeptanz	Managers view programmers as a scarce resource, and are reluctant to «waste» such by doubling the number of people needed to develop a piece of code.
Kunwar, 2019, S. 58	Paarprogrammierung	Managementakzeptanz	It requires an enlightened management that believes that letting two people work on the same task will result in better software than if they worked separately.
Smoczyńska et al.,2019, S. 105	Hybride Vorgehensmodelle	Ergänzung (Scrum/XP)	Both Scrum and Extreme Programming share similarities, such as core values, team roles and responsibilities, phases or scope. Both methodologies work well for small or medium teams (preferably no larger than ten developers) [14]. They are said to work well together – they are quite similar, often adopt each other's practices, and complement each other well [15]. However, there are significant differences between Scrum and Extreme Programming, which allow them to be used simultaneously and successfully in the project.
Smoczyńska et al.,2019, S. 105	XP-Methode	Managementpraktiken	Thus XP focuses on providing the best engineering practices while neglecting management practices.
Smoczyńska et al.,2019, S. 105	Hybride Vorgehensmodelle	Ergänzung (Scrum/XP)	Thus XP focuses on providing the best engineering practices while neglecting management practices. This major difference – with XP focusing on programming practices, and Scrum focusing rather on management practices – is what allows these two methodologies to be successfully used together in a single project.
Vanhala & Kasurinen, 2019, S. 220	Kunde vor Ort	Beteiligung	Based on our results, the customer participation has a significant impact on the quality assurance activities and to the overall success of the agile project.
Vanhala & Kasurinen, 2019, S. 220	Kunde vor Ort	Beteiligung	Additionally, we observed that the customer participation does not require physical presence as documented in some agile practices such as XP, but for a meaningful participation it is sufficient that the customer participates for example via shared digital workspace. In fact, we did not find strong indicators of added benefits from the on-site presence by the customer representatives at the development team.

Vanhala & Kasurinen, 2019, S. 220	Kunde vor Ort	Anwesenheit	Additionally, we observed that the customer participation does not require physical presence as documented in some agile practices such as XP, but for a meaningful participation it is sufficient that the customer participates for example via shared digital workspace. In fact, we did not find strong indicators of added benefits from the on-sitepresence by the customer representatives at the development team.
Alam & Gühl, 2020, S. 140	Hybride Vorgehensmodelle	Evolution	Diese Praktiken haben sich inzwischen weiterentwickelt, finden sich aber auch in anderen agilen Vorgehensmodellen wieder.
Dehn, 2020, S. 36	Hybride Vorgehensmodelle	Adaption	Es lässt sich feststellen, dass XP-spezifische Praktiken, wie z.B. Continuous Integration, fünfmal öfter verwendet werden als die Methode an sich. Das könnte daran liegen,dass Continuous Integration sich gut mit anderen Methoden/Praktiken, z.B. Refactoring kombinieren lässt [10] oder auch, dass einige XP-spezifische Praktiken als Grundbaustein für den Entwicklungsprozess verwendet werden [23].
Donick, 2020, S. 28–29	Metapher	Unklarheit	Ein dazu (nicht nur in der Softwareentwicklung, sondern auch in wissenschaftlichen interdisziplinären Zusammenhängen) häufig verfolgter Weg ist das Anlegen eines Glossars; darin werden benötigte Begriffe so definiert, dass alle Beteiligten der Definition zustimmen und bei Bedarf nachschlagen können. In der Variante Extreme Programming war früher auch die Formulierung von Metaphern aus dem Alltagsbereich vorgesehen, um zwischen unterschiedlichen Verständigungshintergründen zu vermitteln. Sowohl Glossar als auch Metaphern lösen das Problem unterschiedlichen Verstehens nur teilweise – denn sie selbst können (und werden) unterschiedlich verstanden
Dhoodhanath & Quilling, 2020	Paarprogrammierung	Kompetenzen	The biggest challenge was the personality mix in a pair, for example introvert/extrovert pairs, where extroverts can dominate pair collaboration
Ibrahim et al., 2020,S. 164	Testen	Kompetenzen	Testing the software is the duty of tester but in TTD, it is performed by a programmer that needs extra skill to write unit test cases. In addition, it slows down the develop- ment processes [4]
Ibrahim et al., 2020,S. 165	Testen	Aufwand	4. The TTD technique becomes more time consuming if more test cases are often failed

Ibrahim et al., 2020,S. 166	Planungs-spiel	Anforderungs-management	XP does not give good importance to Non-functional requirements, it only focuses on functional and productive work [40]
Ibrahim et al., 2020,S. 166–167	Paarpro-grammierung	Kompetenzen	Coding in pair is one of problematic situation in which both programmers need same skills, mutual understating and good coordination [27],[24]. And double person-monthare required to produce same features.
Ibrahim et al., 2020,S. 166–167	Paarpro-grammierung	Ressourcen	Coding in pair is one of problematic situation in which both programmers need same skills, mutual understating and good coordination [27],[24]. And double person-month are required to produce same features.
Ibrahim et al., 2020,S. 167	XP-Methode	Skalierbarkeit	XP process is hard to implement in a big organization having a large team size, it isonly beneficial to low-risk projects with small team [29],[24].
Ibrahim et al., 2020,S. 167	XP-Methode	Verteilte Teams	XP manifesto does not support geographically distributed projects and team [31],[32].
Ibrahim et al., 2020,S. 167	XP-Methode	Management-praktiken	XP lacks in project management practice and depends upon customer support that may become a risk of project failure
Ibrahim et al., 2020,S. 167	Kunde vor Ort	Abhängigkeit	XP lacks in project management practice and depends upon customer support that may become a risk of project failure
Proba, 2021, S. 281	Kunde vor Ort	Interpretation	Überinterpretation der Aussagen einiger weniger Kunden/Nutzer als repräsentativ für die Gesamtheit der Kunden/Nutzer
Proba, 2021, S. 285	Paarpro-grammierung	Kompetenzen	Ungleiche/Unpassende Zusammenstellung der Programmiererteams (z. B. Unterschiede in der Expertise zu gross, menschliche Differenzen).
Proba, 2021, S. 289	Programmier standards	Dogma	Ineffizientes Handeln und hohe Aufwände bei dogmatischer Verwendung der Programmierrichtlinien möglich;
Proba, 2021, S. 289	Programmier standards	Akzeptanz	Fehlende Akzeptanz der Richtlinien im Team.

Proba, 2021, S. 289	Planungs-spiel	Moderation	[Standup Meeting] Ineffiziente Gesprächsführung durch schlechte Moderation
Proba, 2021, S. 289	Planungs-spiel	Disziplin	[Standup Meeting] Fehlender Informationsaustausch durch unregelmässige/undisziplinierte Durchführung des Standup Meetings oder mangelhafte Institutionalisierung.

10.5 Anhang E: Interviewleitfaden

Beim Interview mit Joseph Pelrine, wurden die Einstiegsfragen nicht gestellt, da seine Expertise bereits belegt ist.

Einstiegsfragen		
Nr.	**Frage**	**Hintergrund der Frage**
1	In welcher Rolle sind Sie aktuell tätig?	Klärung richtige Ansprechperson
2	Wieviel Erfahrung haben Sie in der Beratung in der Schweiz?	Klärung richtige Ansprechperson
3	Wieviel Erfahrung haben Sie in der Beratung ausserhalb der Schweiz?	Ermitteln möglicher Erfahrungen ausserhalb der Schweiz
4	Wieviel Erfahrung haben Sie in der Software Entwicklung?	Klärung richtige Ansprechperson

Schlüsselfragen		
Nr.	**Frage**	**Hintergrund der Frage**
5	Haben Sie selbst schon mal XP als Methode oder Bestandteile daraus eingeführt?	Kontextdefinition
6	Was verstehen Sie darunter XP anzuwenden?	Ermitteln der XP Kenntnisse und dem persönlichen Verständnis von XP
7	Wie wird XP heute in der Praxis angewendet?	Ermitteln der Praktiken und Projektvorgehensweisen in der Schweiz
8	Wieviel Erfahrung haben Sie im Coaching/Betreuen agiler Projektteams?	Ermitteln der Erfahrung spezifisch in agilem Umfeld
9	Wie viele Projekte davon haben nach XP gearbeitet? (schätzungsweise)	Ermitteln der Verbreitung von XP in der Schweiz
10	Was haben Sie für Erfahrungen/Beobachtungen gemacht, wenn XP z.B. in Kombination mit Scrum angewendet wird?	Ermitteln der Erfahrungen mit hybriden Ansätzen (z.B. Scrum/XP)
11	Wie erklären Sie sich, dass XP als Methode kaum noch zum Einsatz kommt, die Praktiken jedoch etabliert sind?	Abfangen möglicher weiterer Erklärungen als jene aus der Literatur
12	Wer definiert bzw. entscheidet welche Methode für ein Projektvorgehen angewendet wird?	Ermitteln, wo die Entscheidungsgewalt für ein Projektvorgehen liegt

13	Welche Schwierigkeiten beobachten Sie in Projektteams, die nach XP arbeiten?	Ermitteln der Schwierigkeiten im Umgang mit XP
14	Wie erlangen die Projektteams die notwendigen Skills,um agil arbeiten zu können?	Ermitteln wie die Skills erworben und Qualifikationen erlangt werden

Abschlussfrage		
Nr.	**Frage**	**Hintergrund der Frage**
15	Was haben Sie dem Thema noch beizufügen?	Abfangen nicht berücksichtigter Punkte und möglicher Indizien für künftige Fragen

10.6 Anhang F: Auswertung Interviews (Kategoriensystem)

Interview	Kategorie	Ausprägung	Anfang	Ende	Segment
C	40-Stunden Woche	Wirtschaftlichkeit	16	16	Sustainable Pace ist ein Problem in der Schweiz, das ist eine Spannung mit dem Management. Weil, wenn Sie eine Managementfirma sind, lohnt es sich eigentlich wirtschaftlich, Überstunden zu schaufeln und diese aus-zuzahlen. Das ist ein vernünftiges Business (), es ist aber suboptimal für die Denkarbeiter, [also] das Entwicklungsteam
B	Einfaches Design	Unternehmens-grösse	28	28	Es gibt zum Beispiel unbequeme Elemente [in XP], die man nicht einfach so in einem Grossunternehmen anwenden kann. Zum Beispiel «Simplicity», also Einfach-heit. Das erlaubt einem jetzt niemand, dass man quasi sagt «he mach doch mal etwas, wo gar keine Datenbank dahinter ist, einfach weil es das Minimum ist, das du bauen könntest». In einem Startup ist das durchaus möglich. In einem Grossunter-nehmen musst du halt alles planen, es heisst dann «du weisst, dass du eine Daten-bank brauchst, also mach eine rein» und die Architektur spricht halt auch noch mit.
A	Gemeinsame Verantwortlichkeit	Mindset	28	28	Collective Code Ownership ist eine Einstellungssache, die kann man nicht frontal ausbilden, das macht man einfach, das lernt man.
C	Gemeinsame Verantwortlichkeit	Qualität	26	26	Ich mache regelmässig auch Code-Reviews bei Kunden und naja, die Qualität des Codes ist selten berauschend.
A	Hybride Vorgehens-modelle	Adaption	18	18	Meistens werden dann nur Teile wie Pair Programming angewendet. Also Pair Programming sehe ich relativ häufig.
A	Hybride Vorgehens-modelle	Adaption	22	22	Wir haben meistens mit Scrum angefangen, was ja mit XP verwandt ist und dann haben wir XP dazu genommen. Ab einem gewissen Punkt sind die Teams genug reif, um eigene Prozesse zu finden. An gewisse Dinge wie Pair Programming – Pair Programming ist eine meiner Lieblingsgeschichten –, Code Reviews, Mob Program-

					ming oder Test Driven Design hält man sich schon fest, aber der Prozess und die ganze Entwicklung wird auf ihre Situation adaptiert.
B	Hybride Vorgehens-modelle	Adaption	14	14	XP anzuwenden, heisst für mich das ganze Set von Empfehlungen versuchen umzu-setzen. Das ist bei allen anderen Methoden tendenziell auch so – zumindest am An-fang beim Starten. Es hat natürlich auch Elemente enthalten, wo man lernt, sich an-passt und Veränderungen gegenüber offen ist.
B	Hybride Vorgehens-modelle	Adaption	14	14	Ich denke was wichtig ist, ist dass man nicht von den technischen Empfehlungen, also die technischen Praktiken, wie man als Entwickler vorgeht, abweichen sollte.
B	Hybride Vorgehens-modelle	Adaption	18	18	Und bei SAFe wird XP lustigerweise erwähnt, aber es ist sehr wenig von dem was XP ursprünglich mal war. Also man übernimmt eigentlich nur die technischen Prakti-ken.
B	Hybride Vorgehens-modelle	Adaption	34	34	Ich denke, am Schluss spielt es nicht so eine Rolle, ob es XP oder Scrum oder so ist. Ich glaube es geht mehr darum, welches Problem man lösen möchte und welche Praktik das Problem lösen kann.
D	Hybride Vorgehens-modelle	Adaption	12	12	Jetzt muss ich kurz überlegen was sonst noch zu XP gehört (–) das ist mittlerweile so ein Durcheinander der Methoden
D	Hybride Vorgehens-modelle	Adaption	12	12	Also eigentlich [habe ich] schon fast XP als Ganzes [eingeführt] aber nicht ganz so extrem. Es war nicht alles Pair Programming, es war nicht immer ganz nahe beim Kunden – aber schon in die Richtung.
D	Hybride Vorgehens-modelle	Adaption	20	20	Man muss immer situativ schauen; ist es nun etwas, was ich in Pair Programming machen soll oder handelt es sich um Fleissarbeit, die ich auch alleine erledigen kann? Oder ist es wirklich etwas wo man das ganze Team braucht? Wo es das Wissen von vielen Leuten braucht? Dann ist esvielleicht schlauer ein Mob-Program-ming anzuwenden. Ich sehe das sehr situativ was man anwenden soll.
D	Hybride Vorgehens-	Adaption	34	34	Ich sage mal die guten Teams haben XP Bestandteile mitgenommen und die Teams,

	modelle				die es noch nicht kannten, die haben einfach Scrum als Hülle genommen und dahinter einfach weitergearbeitet wie sie vorher gearbeitet haben. So entsteht quasi ein Mini-Wasserfall [Vorgehen].
C	Hybride Vorgehensmodelle	Adaption	16	16	Beim grünen [Kreis] würde ich sagen wieder zwei [Praktiken] werden richtig gemacht; Continuous Integration und Coding Standard. Aber Coding Standard nicht immer im Sinn von XP, eher von formelleren Sachen.
C	Hybride Vorgehensmodelle	Adaption	16	16	Das grobe Problem mit XP ist, XP ist technisch orientiert und recht diszipliniert und viele Leute sind geflüchtet in Scrum, was eigentlich auchdiszipliniert ist aber deshalb machen alle Leute Scrum aber «meine Firma hat's einfach angepasst» – was sie angepasst haben ist eigentlich das was für sie störend ist weg zu nehmen. Es ist nicht eine Anpassungim Sinne von «wir haben eine Hypothese gestellt, wenn wir es anders machen, wirds besser», sondern eher «was immer stört das nehmen wir weg», was eigentlich der Tod vom Sinn von XP, Agil oder Lean ist.
C	Hybride Vorgehensmodelle	Adaption	28	28	Sogar in SAFe und so, wenn Sie schauen, alle Techniken aus XP sind in SAFe versteckt, niemand macht sie, aber sie sind sauber dokumentiert und empfohlen und als Best Practices für professionelle SAFe Teams. Aber wenn Sie in den SAFe Teams in der Schweiz schauen, wäre es überraschend, wenn Sie mehr als 1–2 dieser Techniken anwenden.
E	Hybride Vorgehensmodelle	Adaption	8	8	Wenn du ‹agile by the book› machst, dann bist du nicht agil. Du folgst eine Religion, hast einfach eine andere Bibel. Es ist für mich eine Herausforderung, Leuten beizubringen, dass sie sich nicht an den Scrum Guide halten sollen. Es geht darum, dass man auch auf Prozessebene agil ist. Man sollte by the book anfangen, schauen was funktioniert und dann nach «apply, inspect and adapt» arbeiten.
E	Hybride Vorgehensmodelle	Änderungsbereitschaft	17	17	Jetzt sind wir hier [rechts vom mittleren Chasm], bei Firmen, die das nicht wollen. Ihre Bereitschaft sich zu ändern und anzupassen ist relativ gering, wenn überhaupt eine da ist. Also was gibt es für Methoden, damit sie ihren Investoren sagen können

					«wir sind auch agil»? Das Einfachste ist, man macht sich eine Wand frei, holt sich ein paar Post-Its – aber ja nicht zu viele – klebt sie an die Wand, um sagen zu können «wir machen Kanban, wir sind auch agil!».
E	Hybride Vorgehensmodelle	Ergänzung (Scrum/XP)	15	17	Aber ich habe das schon mal gesagt, das Wichtigste was bei XP an Praktiken rausgekommen ist und es in den Mainstream geschafft hat, ist Scrum. [...] Und Scrum richtig gemacht, zeigt dir warum du XP brauchst. XP gibt Scrum schnelleres Feedback im Sinne des Produktes und im Sinne der Akzeptanz.
E	Hybride Vorgehensmodelle	Ergänzung (Scrum/XP)	23	23	Steve Freeman hat mal gesagt, Scrum ist die Rache von XP. Wenn du Scrum richtig machst, merkst du, dass du unbedingt XP brauchst, wenn du an einem Softwareentwicklungsprojekt arbeitest.
A	Hybride Vorgehensmodelle	Ergänzung (Scrum/XP)	12	12	Häufig benutze ich einfach die Software Craftmanship Aspekte. Je mehr es dann aber in Businessprozess-Themen geht, desto mehr kommen die Dinge aus Scrum oder Kanban.
B	Hybride Vorgehensmodelle	Ergänzung (Scrum/XP)	26	26	Scrum sagt natürlich nichts über technische Praktiken. Es sagt weder etwas darüber wie man Pair Programmieren soll noch wie man Tests vor der Programmierung schreibt. Also über testgetriebenes Arbeiten steht da grundsätzlich nichts. [Scrum] kann ja auch angewendet werden auf Nicht-Software-Teams. Es ergänzt sich durchaus gut und darum glaube ich, dass man etwas nimmt, was populärer ist oder mehr Lobby hat und beim Rest sagt man «wir kombinieren es dann».
D	Hybride Vorgehensmodelle	Ergänzung (Scrum/XP)	16	16	Leitender Standard ist heute Scrum. Scrum und XP haben einen relativ grossen Overlap aber die Leute kennen die Practices eher aus Scrum als aus XP.
D	Hybride Vorgehensmodelle	Ergänzung (Scrum/XP)	18	18	Also ich glaube Scrum funktioniert nicht ohne XP. Also ohne einen gewissen Teil aus XP zumindest. Primär [kann Scrum] nicht ohne die Engineering Practices funktionieren. Sonst wäre Scrum einfach eine leere Hülle. Scrum kann für mich eigentlich fast nur mit den XP Practices wirklich gut funktionieren. Denn ansonsten ist es einfach häufig ein «Mini-Wasserfall» [Vorgehen]. Dann ist es einfach so «Ja wir haben einen

					Sprint und pro Sprint machen wir eigentlich ein Wasserfall [Vorgehen]: am Anfang planen, dann coden und wenn wir Glück haben und am Ende noch Zeit bleibt, testen wir noch».
D	hybride Vorgehens-modelle	Ergänzung (Scrum/XP)	34	34	Die Teams, die mit Scrum gearbeitet haben, haben dann gedacht «[Scrum] ist gut und recht, alle zwei Wochen machen wir diese Eventsund so aber wie man coded, darauf gibt Scrum keine Antwort» und Scrum sagt ja auch explizit, dass dazu keine Angaben gemacht werden. Dann haben wir gemerkt, dass wir eigentlich einfach mit XP weitermachen sollten.
C	Hybride Vorgehens-modelle	Ergänzung (Scrum/XP)	16	16	50–80% wird benutzt von den Leuten, die irgendein «Agil» machen, hauptsächlich Scrum.
C	Hybride Vorgehens-modelle	Ergänzung (Scrum/XP)	22	22	Aktuell in der agilen Welt machen Scrum oder Varianten von Scrum wie Scrum/XP und Scrumban rund 70–80% aller Projekte aus. Es wäre töricht zu versuchen XP jetzt durchzudrücken. Ich finde es viel effektiver zu sagen «ok wir nehmen das Scrum Framework» und dieses ist sehr XP affin, da kann man sehr viele Sachen von XP reinbringen.
C	Hybride Vorgehens-modelle	Ergänzung (Scrum/XP)	24	24	Das funktioniert sehr gut. Scrum/XP und Scrum Lean sind affin.
C	Hybride Vorgehens-modelle	Ergänzung (Scrum/XP)	32	32	Was schön ist, die Scrum Leute vom Agilen Manifesto haben sehr viel übernommen. Was passiert ist, wenn man den neusten Scrum Guide anschaut; sie haben viele Sachen weggenommen, so dass Scrum für alles angewendet werden kann. Die Methode, die in Scrum empfohlen worden sind, die XP Ansätze, sind in Scrum integriert.
E	Hybride Vorgehens-modelle	Komfortzone	8	8	Das Problem mit ‹inspect› and ‹adapt› ist, dass viele Unternehmen sich einfach ein Buch geholt haben aber nichts machen was sie aus der Komfortzone bringt, dann müssen sie sich nicht fragen wieso es nicht funktioniert.
E	Hybride Vorgehens-	Mainstream	10	10	Moore hat gesagt, dass es bei disruptiven Technologien diesen Chasm gibt. Nur die,

	modelle				die im Mainstream [rechts vom Chasm] arbeiten sind überlebensfähig. Will Pietri, ein alter agiler Knacker, hat mal gesagt, dass es bei agile auch noch einen [Chasm] hier [in der Mitte] gibt und zwar zwischen Firmen, die agile machen, weil sie es wollen und Firmen, die es machen weil sie müssen. Und wir sind jetzt ganz deutlich hier [rechts vom mittleren Chasm] also bei Firmen, die es tun, weil sie es müssen und es eigentlich gar nicht wollen. Was sie wollen ist – ich nenne es #MeTooAgile –, sie wollen sagen, dass sie es tun, obwohl sie eigentlich möglichst wenig verändern möchten.
E	Hybride Vorgehens-modelle	Mainstream	11	11	Und XP liegt in dieser Kurve bei den Innovators. XP hat es nie aus der Nische geschafft. XP ist schwer, es ist verdammt schwer.
E	Hybride Vorgehens-modelle	Mainstream	11	11	In den Mainstream haben es nur einzelne XP-Praktiken geschafft.
A	Hybride Vorgehens-modelle	Regulationen	26	26	Je nach Kontext funktionieren halt gewisse Dinge nicht. Ich arbeite in einem regulierten Umfeld und das heisst ich muss meine Spezifikationen irgendwann freezen, ablegen und sagen «das ist jetzt das Produkt, jetzt kann ich es auf den Markt geben» und ab dem Punkt fällt XP beispielsweise auseinander. In XP würde man sagen «das gibt's, nicht dass man Spezifikationen fixt, die sind immer anpassbar». Aber der Kontext in der Medizin oder im regulierten Umfeld ist einfach so, dass man irgendwann die Spezifikation fixen muss, damit man das Gerät auf den Markt bringen kann, weil ich muss das irgendwann dem TÜV oder FDA übergeben und die können damit nicht umgehen, wenn ich sage «das ist ein wachsendes Ding, wir schauen schon für gute Qualität».
C	Hybride Vorgehens-modelle	Regulationen	26	26	Der Bund hat die HERMES-Methode, welche er über die Hintertür allen Kantonen verordnet hat. Die HERMES-Methode ist eine Krücke. Ich habe es noch knapp geschafft über HERMES () mit anderen Leuten Scrum einzuführen, aber es ist eine Water-Scrum-Fall [Methode]. Darin können Sie weder mit XP noch mit Scrum arbeiten und das ist die offizielle Methode.

A	Kunde vor Ort	KundInnennähe	16	16	Das funktioniert je direkter desto besser. Also ich unterscheide viel zwischen dem Kunden – also der, der das Produkt bezahlt – und dem Endbenutzer, der das Produkt braucht. Mein Ziel ist es möglichst direkt und häufig an den Endbenutzer zu gelangen. In der Realität ist es so, dass ich im Minimum alle zwei Wochen länger mit meinen Kunden oder sogar den Endbenutzern zusammensitze und wir schauen was wir an der Software gemacht haben.
A	Kunde vor Ort	KundInnennähe	16	16	Mein Ziel ist es eigentlich immer möglichst schnell mit dem Endbenutzer auf Du-Basis zu gelangen, sodass ich ihm beispielsweise schnell ein Mail senden kann und ihn beispielweise mit einem Screenshot fragen kann, ob es das ist was er sich vorgestellt hat oder ich rufe ihn schnell an. Also wirklich so nahe wie möglich.
B	Kunde vor Ort	KundInnennähe	18	18	Dank Scrum und anderen Managementpraktiken sind eigentlich die Manager wieder voll am Steuer und auch der Kunde ist wieder in die Ferne gerückt. Der enge Kundenkontakt zwischen Entwickler und Endkunde, also dem Benutzer sozusagen, und auch dem Sponsor vom Projekt der ist wieder verloren gegangen.
D	Kunde vor Ort	KundInnennähe	22	22	Die Nähe zum Kunden ist auch immer schwierig. Es kommt sehr auf das Organisationssetup an. Je grösser die Firma, desto schlechter geht es. Also desto weiter weg sind die Entwickler oder das Team von dem Benutzer. Bei kleineren Teams ist es häufig noch etwas besser aber das ist – was ich persönlich finde – in der Schweiz grundsätzlich eher schlecht. Das Team ist meistens zu weit weg [vom Kunden] oder es gibt Telefonspiele über – in Scrum typischerweise – den Product Owner oder den Requirements Engineer, so funktioniert das nicht gut.
E	Metapher	Resonanz	41	41	Durch die Metapher wird unsere ‹Geheimsprache› in einer weniger reichhaltigen Sprache formuliert, in einer Sprache, die mehr Resonanz erzeugt für jemanden der nicht so vom IT-Fach ist.
E	Metapher	Unklarheit	41	41	Ich kann dir die Idee der Metapher erklären. Ganz ehrlich, geschichtlichweiss ich jetzt nicht in welcher Ausgabe des Buches von Kent die Metapher beschrieben wird.

					Ich weiss aber was es war und was wir damit wollten und die Probleme, die wir damit hatten. Und ich denke es ist heute nicht mehr so präsent, weil es sehr schwer war zu verstehen.
E	Metapher	Unklarheit	45	45	Das Problem ist, dass die meisten Leute den Begriff Metapher nicht verstehen.
E	Paarprogrammie-rung	Akzeptanz	31	31	In der Einführung von XP war es vor allem der Widerstand der Entwickler, gerade gegen Pair Programming und testgetriebene Entwicklung.
E	Paarprogrammie-rung	Disziplin	11	11	Wir haben uns gesagt «warum machen wir Pair Programming? Weil XP zu schwer ist, dass einer alleine die Disziplin hat das durchzuziehen» deswegen hat Kent [Beck] es auch «a discipline of programming» genannt. Es braucht diese Disziplin und das obwohl wir es so strukturiert haben, dass ‹Normalsterbliche› es machen konnten. XP hat sehr viel verlangt.
E	Paarprogrammie-rung	Infrastruktur	31	31	Also wenn du die früheren Bücher liest, siehst du, dass die Firmen, die es gemacht haben, extra Tische bauen liessen. Und tatsächlich gerade am Anfang zeigt das, dass die ganze Infrastruktur nicht für XP ausgelegt worden ist.
E	Paarprogrammie-rung	Kompetenzen	31	31	Es besteht das Problem des Beobachtet-Werdens; Pair Programming war unter Leuten mit demselben Wissenstand eher noch machbar, aber wenn du mal einen Experten hast und ein Anfänger, naja – aber das muss man tatsächlich unter Wissenstransfer abtun.
C	Paarprogrammie-rung	Kosten	28	28	Pair Programming ist immer noch ein heisser Diskussionspunkt in der Schweiz, aber es sind nur Meinungen, es gibt niemanden der eine empirische Studie gemacht hat, auch weltweit gibt es wenige empirische Studien. Alle empirischen Studien, die gemacht worden sind, sind nicht schlüssig. Sie sagen es scheint etwas zu bringen, worauf die Diskussion Pair Programming kostet mehr, sich schon erledigt hat.
E	Paarprogrammie-rung	Logistik	31	31	In der Weiterführung von XP, gab es eher logistische Schwierigkeiten. Wie beispielsweise der Programmiersprache, der Netzwerklatenzzeit oder der Aufstellung der Tische damit sie für Pair Programming geeignet sind.

D	Paarprogrammie-rung	Mainstream	16	16	Und Dinge wie Pair Programming und Test-Driven Development sind sicher nicht im Mainstream angekommen, aber es gibt doch einige Leute, die es anwenden. Es gibt einige Teams, die es anwenden. Aber es ist nach wie vor immer noch umstritten. Es gibt immer noch Diskussionen ob TDD etwas bringt oder nicht. Aber nur bei den Leuten, die es nie gemacht haben. Bei den Leuten, die es kennen und gemacht haben ist es klar, dass es in der richtigen Situation sehr viel bringt.
D	Paarprogrammie-rung	Management-akzeptanz	24	24	Es gibt immer noch sehr viele Entwickler aber vor allem auch Manager, die sagen «was 10 Leute vor einem PC? Das ist ja Zeitverschwendung!»
E	Paarprogrammie-rung	Paarbildung	51	51	Was ich gelernt habe; Pair Programming mit Testern bringt dir als Entwickler bei, wie man testbaren Code schreibt. Pair Programming mit einem Product Owner oder mit einem Kunden bringt dir bei, wie man lesbaren Code schreibt. Und das steht sonst kaum irgendwo geschrieben. Man muss Pair Programming nicht nur mit anderen Entwicklern machen.
E	Paarprogrammie-rung	Wohlbefinden	31	31	Es besteht das Problem des Beobachtet-Werdens; Pair Programming war unter Leuten mit demselben Wissenstand eher noch machbar, aber wenn du mal einen Experten hast und ein Anfänger, naja – aber das muss man tatsächlich unter Wissenstransfer abtun.
B	Schweiz	Ausbildung	30	30	Wer frisch ab Schule kommt oder einer Fachhochschule, der hat zumindest etwas von Testing gehört. Wer als Quereinsteiger irgendwo über einen 3–6-monatigen Intensivkurs kommt, der lernt einfach zu programmieren und Testing ist genau eine Vorlesung von zwei Stunden und dann heisst es «nun hast du etwas übers Testing gehört».
B	Schweiz	Ausbildung	30	30	Was wir gemacht haben, auch in Unternehmen als ich Entwicklungsleiterwar. Dort haben wir Inhousekurse organisiert und haben die Leute geholt, die das damals noch angeboten haben. Es gibt nach wie vor solche Angebote, man muss sie einfach nutzen. Wir hatten das Gefühl, sonst lernen [die Entwickler] das nicht und der

					Fokus muss daraufgelegt werden. Wir haben einzelne Leute nachgeschult aber in der Regel haben wir die ganze Mannschaft genommen und es alle gemeinsam gelernt.
C	Schweiz	Ausbildung	24	24	ich glaube jetzt machen alle Fachhochschulen in der Schweiz mindestens einen «Agil-Scrum-Abstrich» für alle IT-Studenten. Wie gut und wie tiefder Abstrich ist, [darüber] können wir uns streiten. Ich habe aktuell eine Studentin, die von der [anonymisiert] Fachhochschule kommt und bei uns in [anonymisiert] gelandet ist. Sie hat noch nie Git gesehen und () Git ist Standard seit 15 Jahren, das ist keine Kritik gegen die Studenten, es zeigt, dass der Pfad wie die Ausbildung gemacht wird, sei es Fachhochschule, sei es Lehre oder in der Firma, ist verbesserungsbedürftig, um es höflich zu sagen und wir sind () Jahre hinter den guten Firmen der Welt.
C	Schweiz	Ausbildung	24	24	Nun gibt es die genialen Konstrukte «Lehre-BM-Fachhochschule» oder direkt Fachhochschule, beide sind sehr gut, aber Sie sind selber in der Fachhochschule; Die Variation bei den Dozenten ist faszinierend. Ich kenne Chur nicht, aber ich kenne ein paar andere Fachhochschulen. Es gibt geniale Dozenten und es gibt Dozenten die sind seit 30 Jahren da und erzählen seit 30 Jahren das Gleiche und in einem IT-Bereich wo schon alle 6–10 Jahren etwas Neues kommt, wenn man drei Wellen hinten drin ist, tut's weh.
C	Schweiz	Ausbildung	26	26	Es gibt die, die interessiert sind, () an Messen gehen und so und es selbst bezahlen.
C	Schweiz	Ausbildung	26	26	Dann [mittels] Training ‹on the job›. Und es ist sehr variabel es gibt sehr gute Training ‹on the job›, wo ich sagen würde «perfekte Lösung» und der Rest ist dann eine Alibiübung, um sich die Kosten zu sparen.
C	Schweiz	Ausbildung	26	26	Drittens behaupten sie, dass sie Training haben, aber es ist eher eine Selbstlüge oder eine exquisite Lüge, ich weiss nicht wie man das sagen kann.
C	Schweiz	Ausbildung	32	32	Viele der Entwickler, die jetzt mit agil arbeiten, die sogenannten Late-Adapter, was 60–90% der Entwickler ausmacht, haben sehr oft keine Ahnung von dem und da sie

					sich nicht speziell weiterbilden, kommen sie wenig damit in Berührung. Ich hoffe, dass sich das dreht, aber das muss langfristig geplant werden, dass Fachhochschulen und Uni die Studentenschulen, dass neue [Studienabgänger] das kennen. Aber das ist eine Hoffnung. Fachhochschulen sind so weit aber Uni oder ETH tun sich schwer, sagen wirs so. Nicht alle Uni Bern und teilweise Zürich machen es gut, und andere weniger und ETH (–) naja (–).
E	Schweiz	Ausbildung	33	33	In der Regel ‹on the job›; wir sitzen zusammen und machen das.
C	Schweiz	Auslagerung	26	26	Es gibt zum Beispiel keine grossen IT-Projekte [mehr] in der Schweiz, das ist vorbei. Die letzten grossen waren PostFinance und die Banken, die haben alles ausgelagert. PostFinance entwickelt hauptsächlich in Indien und die grossen Banken entwickeln in Grossbritannien. Das haben nicht alle Leute realisiert. Grosse Projekte in der Schweiz sind nicht mehr da.
C	Schweiz	Entwicklungs-fähigkeit	26	26	Das andere was viele noch nicht realisieren, die Komplexität der IT-Welt steigt auch brutal, die letzten 30 Jahre – zwar nicht jedes Jahr – aber die letzten 30 Jahre hat die Komplexität Liga gewechselt. Und man kann sich wirklich streiten, ob die IT-Entwicklungsfähigkeiten in der Schweiz den Rhythmus halten von der Entwicklung der IT.
C	Schweiz	Entwicklungs-fähigkeit	34	34	Und dann kann man nicht eine Kultur wie XP einführen, da muss man auch ehrlich sein, das ist zwei Generationen weiter.
E	Schweiz	Konservatismus	10	10	IT-technisch sind wir Schweizer so konservativ, wir warten bis etwas rauskommt und erfolgreich ist, und dann werfen wir einfach eine Unmenge Geld raus, um den technologischen Rückstand wieder einzuholen.
C	Schweiz	Reglementierung	32	32	Was schön ist, die Scrum Leute vom Agilen Manifesto haben sehr viel übernommen. Was passiert ist, wenn man den neusten Scrum Guide anschaut; sie haben viele Sachen weggenommen, so dass Scrum für alles angewendet werden kann. Die Methode, die in Scrum empfohlen worden sind, die XP Ansätze, sind in Scrum inte-

					griert. Es wurde auf die Seite geschoben «ja ihr könnt es machen». Und wenn man mit Experten redet, sogar Ken Schwaber und so steht dahinter, «ja machts» aber es steht nicht mehr explizit drin. Worauf wir in unserem Land sehr reglementorientiert sind, dann rutscht es wieder raus. Worauf die Qualität der Entwicklungsteams eher abnimmt, aktuell nicht zu viel, aber es nimmt eher ab.
C	Schweiz	Wahrnehmung	34	34	Die Wahrnehmung in unserer Gesellschaft, in der Industrie und so, dass Software ein Kernelement geworden ist, ist noch nicht da. Wir haben in der Schweiz nicht mal eine IT-Software-Assoziation, es gibt 3 oder 4, die sich streiten und versuchen mal zu fusionieren aber im Prinzip ist es nicht klar, wo IT, Softwareengineering, Software im Kanton oder in Bern angegliedert ist.
C	Schweiz	Wahrnehmung	34	34	Die Wahrnehmung, das instudrielle Gewicht und die Anzahl Leute, die in unserem Fach arbeiten und die Struktur dahinter die passen noch nicht. Anders in den USA, dort gibt es ganze Gremien von IT-Verantwortlichen, auch der Chef von Apple und so, und die treffen sich und diskutieren.
E	Testen	Akzeptanz	31	31	In der Einführung von XP war es vor allem der Widerstand der Entwickler, gerade gegen Pair Programming und testgetriebene Entwicklung.
B	Testen	Aufwand	16	16	Die Leute sind faul beziehungsweise Entwickler codieren gerne mal schnell etwas.
E	Testen	Ausbildung	11	11	Wenn wir ein neues Teammitglied gehabt haben, das Einzige was er/sie die ersten zwei Wochen gemacht hat, war Tests lernen, neue Tests schreiben und schauen, ob sie das System zum Absturz bringen können.
B	Testen	Ausbildung	16	16	In dieser kurzen Zeit, in der [anonymisiert] gelernt hat zu programmieren, hat [anonymisiert] nichts über Testing gelernt. Und so eine essentielle Tätigkeit wurde einfach weggelassen. Das finde ich komisch. Aber in der Industrie ist das nicht wirklich (–) ich glaube bei Fachhochschulen ist es anders aber so in der Regel denke ich, wer das lehrt und auch in vielen Unternehmen wird das nach wie vor stiefmütterlich behandelt. Es wird nicht ermuntert, dass man erst einen Test schreibt und danach den

					Code dazu.
D	Testen	Legacy Code	26	26	Viele Teams haben auch das Problem, dass sie eine bestehende Codebasis haben, bei welcher keine bestehenden oder nur wenige Tests vorhanden sind. Das dann noch testbar machen, dass man überhaupt Tests schreiben kann, ist natürlich das Schwierigste, was es überhaupt gibt.
D	Testen	Legacy Code	26	26	Am idealsten ist es, wenn es etwas gibt, was auf der grünen Wiese startet, dann muss man sich nicht mit Legacy Code auseinandersetzen.
B	Testen	Mainstream	14	14	Also die technischen Praktiken, zum Beispiel testgetrieben zu arbeiten, das ist in der heutigen Zeit nach wie vor nicht gegeben.
D	Testen	Mainstream	16	16	Und Dinge wie Pair Programming und Test-Driven Development sind sicher nicht im Mainstream angekommen, aber es gibt doch einige Leute, die es anwenden. Es gibt einige Teams, die es anwenden. Aber es ist nach wie vor immer noch umstritten. Es gibt immer noch Diskussionen ob TDD etwas bringt oder nicht. Aber nur bei den Leuten, die es nie gemacht haben. Bei den Leuten, die es kennen und gemacht haben ist es klar, dass es in der richtigen Situation sehr viel bringt.
E	Testen	Mindset	11	11	Denk mal warum Leute testgetriebene Entwicklung hassen. Streng genommen heisst testgetriebene Entwicklung, dass du keinen einzigen Buchstaben Code schreibst, wenn du keinen Test hast, der nicht läuft. Also das erste was du machen musst ist einen Test zu schreiben, der dir sagt «du bist ein [zensiert], du kannst das nicht», weil der Test noch nicht läuft. Stell dir vor, was das mit dem Ego eines Softwareentwicklers macht. Das erste was du machst ist einen Test zu schreiben, der nicht läuft und dann schreibst du nur so viel Code bis der Test läuft. Aber die psychologischen Veränderungen, die das mit sich bringt, sind ultrastark; Du weisst jeden Augenblick, ob dein Code läuft oder nicht!
B	Testen	Mindset	16	16	Das Vorgehen liegt auch nicht unbedingt in der Natur eines Entwicklers und wenn man sich das nicht selbst etwas auferlegt und diese «Schule» tatsächlich versucht

					so umzusetzen dann wird es schwierig.
B	Testen	Zeitpunkt	32	32	Viele fangen mit automatisierten Tests zu spät an, machen es zu wenig, machen es zu wenig diszipliniert. Und weil sie es nicht schaffen eine Grundlage aufzubauen, bricht auch der ganze Rest zusammen.
A	XP-Methode	Akzeptanz	18	18	Was ich auch sehe ist, dass bei vielen Leuten Extreme Programming einen schlechten Ruf hat, weil es so nach ‹hacken› und ‹planlos drauf los arbeiten› aussieht.
D	XP-Methode	Akzeptanz	24	24	Die Schwierigkeit ist grundsätzlich die Akzeptanz gewisser Engineering Practices. Die Akzeptanz für TDD oder Pair Programming ist nicht im Mainstream angekommen.
E	XP-Methode	Akzeptanz	29	29	Die hatten geschätzt, dass wir 6–9 Monate für eine Anwendung brauchen. Wir haben aber beide in 4 Monate geliefert. Fully refactored, alles lief sauber durch den Testbetrieb, wie geschmiert mit Olivenöl. Das hat dazu geführt, dass XP als Methode in der Bank verboten wurde. Und wir haben aufgedeckt; In der 40-jährigen Informatikgeschichte der Bank: das erste Projekt, welches on-time, on-budget, on-spec geliefert hat und das war für sie eine Bedrohung.
C	XP-Methode	Anpassungs-fähigkeit	24	24	Und das ist so der XP-Mindset, der genial ist, aber er hat gewisse (), zum Beispiel Pair Programming, das steht ausser Diskussion, wenn du nicht mit den Leuten zusammenarbeiten magst; Dein Problem, pass dich an oder verlasse das Team. […] Und das ist ein kultureller Switch und sehr gute Teams in der Schweiz machen es. Sagen wir, dass der Teamdurchschnitt in der Informatik in der Schweiz nicht berauschend gut ist.
D	XP-Methode	Begleitung	26	26	Oder man zieht jemanden bei, der es kann und für Pair Programming oder Mob Programming mit den Leuten zusammensitzt und das auch gleich mit ihnen macht.
D	XP-Methode	Begleitung	26	26	Und meine Erfahrung ist, dass es etwa ein halbes Jahr Begleitung von einem Team braucht, bis es wirklich «klick» macht und sie es anwenden können. Es ist natürlich nicht so schwarz-weiss «die einen könnens, die anderen nicht» aber es braucht etwa

					so ein halbes Jahr bis man so alle Fälle mal gesehen hat, auch die Schwierigeren, wie es auch in der Praxis funktioniert.
B	XP-Methode	Beliebtheit	30	30	Natürlich kann man jede Menge Kurse machen, aber ich glaube die wenigsten (–) es liegt etwas am Entwickler selbst. Ein guter Entwickler ist immer extrem gerne mit neuen Technologien unterwegs. Womit er aber nicht so gerne unterwegs ist, sind gute Engineering Practices.
C	XP-Methode	Beliebtheit	14	14	Und da können wir nachher draufkommen, warum XP, oder sogar gewisse Praktiken von XP, nicht so beliebt sind in der Schweiz.
C	XP-Methode	Disziplin	16	16	Das grobe Problem mit XP ist, XP ist technisch orientiert und recht diszipliniert
E	XP-Methode	Disziplin	27	27	XP ist zu schwer. XP verlangt Disziplin.
A	XP-Methode	Dokumentation	26	26	XP ist sehr cool, sehr offen, aber es ist nicht klar strukturiert und als unerfahrener Softwareengineer brauche ich unter Umständen eine Schritt-für-Schritt Anleitung und das bietet XP nicht.
D	XP-Methode	Dokumentation	36	36	Es war damals für viele Entwickler, die diesen Methoden einen Namen gegeben haben, so klar, dass sie es nicht beschrieben haben und das ist dann verloren gegangen. Also dass man TDD macht, ist bei allen – also auch bei den Unterzeichnern des Agilen Manifests – eigentlich klar, dasses etwas Gutes ist aber die haben sich wohl gedacht «das macht sowiesojeder, also schreiben wir es nicht auf». Aber der Rest der Welt macht es natürlich nicht und sagt, dass es ja da nicht steht.
E	XP-Methode	Dokumentation	55	55	Und die Tatsache ist, dass die meisten Dinge nirgendswo geschrieben stehen. Die meisten Dinge sind irgendwo spät abends in einer Beiz entstanden.
E	XP-Methode	Dokumentation	55	55	Es gibt Dinge, die wichtig sind, die jedoch nirgendswo geschrieben sind. Das agile Manifest, war der kleinste gemeinsame Nenner von dem was 17 weisse amerikanische Männer mittleren Alters alle, die entweder Angestellte oder Besitzer von Tool- und Prozess-Firmen waren, sich darauf einigen konnten.

A	XP-Methode	Domänenwissen	26	26	Also in XP kann man nicht sagen man macht XP und arbeitet eine Checkliste ab und dann funktioniert es. Man braucht viel Domänenwissen.
A	XP-Methode	Durchhaltever-mögen	32	32	Die grösste Schwierigkeit sehe ich in der mangelnden Zeit für Reflexion und dem mangelnden Durchhaltewillen, damit sich das etablieren könnte. Im Sinne von man probiert es mal, es funktioniert nicht wie erwartet beim ersten Mal und dann sagt man direkt «ah das funktioniert nicht» und wirft es wieder weg.
B	XP-Methode	Durchhaltever-mögen	32	32	Und weil sie es nicht schaffen eine Grundlage aufzubauen, bricht auch der ganze Rest zusammen. Das zeigt, dass die Praktiken alle irgendwie zusammengehören und wenn man sie nicht durchgängig anwendet, dann bricht es zusammen und wenn man keinen Erfolg hat, dann hört man auch mit dem Rest auf. Das zieht einen runter, anstatt dass es aufbauend ist und dort braucht es ein relativ grosses Durchhaltevermögen.
B	XP-Methode	Durchhaltever-mögen	32	32	Ich glaube es braucht halt von der Leitung auch das Durchhaltevermögen, die Standfestigkeit, trotz dem Druck des Kunden, der dann auch manchmal sagt «ich hätte lieber 5 neue Features, anstatt dass ihr jetzt noch Tests schreibt», dass er nicht versteht warum man in erster Linie Qualität braucht. Vor allem wenn er dann scheinbar dadurch «zu wenig» bekommt. Der Kampf und der Druck auszuhalten, das wollen die Wenigsten.
B	XP-Methode	Durchhaltever-mögen	34	34	Man kann sich aus Praktiken von XP bedienen. Ich glaube es existiert keine schablonenartige Methode. Es ist aber schade, wenn man nicht zumindest damit startet mal alles [aus der Methode] anzuwenden und erst nachdem man einen informierten Entscheid treffen kann, weil man die Probleme, die man dann hat, selbst durchleben musste, eine Entscheidung fällt. Viele sagen von Anfang an «das funktioniert nicht», «das dürfen wir nicht machen», «das ist uns nicht erlaubt», ich glaube, wenn man nach diesen Ansätzen fahrt, dann wird es nie was. Wenn man sich etwas zu Herzen nimmt, dann sollte man schauen; Wie interagieren diese verschiedenen Teile untereinander und wie unterstützen sich diese gegenseitig. Man sollte es zuerst so pro-

					bieren, wie es tatsächlich beschrieben wurde. Denn zumindest im grösseren Kontext hat es mal so funktioniert.
B	XP-Methode	Durchhaltever-mögen	34	34	Heute werden einem gerne Schablonen verkauft und SAFe zum Beispielist eine riesige Schablone, wo es schwierig ist, Dinge herauszunehmen. Insofern gefällt mir halt Scrum, Kanban oder auch XP besser, weil sie einen relativ minimalistischen Overhead haben im Vergleich zu grossräumigen Aktionen, wo man dann extrem weit weg vom Kunden ist. Ich glaube etwas, was in XP eigentlich sehr zentral war, ist dass «the customer's always available». Dabei handelt es sich aber um eine idealisierte Welt, aber man muss das Ideal versuchen möglichst anzustreben und die Meisten haben das aufgegeben.
D	XP-Methode	Durchhaltever-mögen	24	24	Das ist mal das Eine und das Andere ist, dass die Engineering Practices auch gelernt werden müssen, die kommen nicht einfach so. Ganz speziell TDD ist etwas, was man lernen muss und am Anfang macht man viele Fehler und ist langsamer und viele Teams, die es probieren sagen dann «nein, das ist ja noch viel mühsamer und jetzt haben wir auch viele Tests, die wir warten müssen» und die [Tests] sind noch nicht gut geschrieben, weil das Team es noch nicht beherrscht und dann springen sie ab, bevor sie es gut genug können, dass sie daraus Gewinn ziehen können.
D	XP-Methode	Durchhaltever-mögen	25	26	I: Und wie erlangen die Produktteams die notwendigen Skills, um agil arbeiten zu können? B: Zum einen braucht es Durchhaltevermögen, bis man es kann – so haben wir es gelernt.
C	XP-Methode	Eigeninitiative	24	24	Ich habe mit Scrum Agil angefangen Anfang 2000 und die ersten 10 Jahre musste ich erklären, warum das keine Schnapsidee ist. Jetzt ist es weiter. Und XP ist noch schlimmer. Scrum ist noch sanft «wir verbesserndas Team» und XP hat die Eigenschaft «die Leute, die in einem XP Team sind sind Profis», weil sie machen selbst ihre Weiterbildungen und es ist klar, dass die Leute, die XP machen, wenn Sie schauen was Kent Beck sagt und so als Vater von XP; Es gibt eine implizite, teil-

					weise explizite Erwartung, dass die Leute fähig sind und wenn es ihnen an etwas fehlt, dass sie es ziemlich zügig lernen. () Es gibt nicht wenig Informatiker oder Leute, die in der Informatik arbeiten, die kein einziges technisches Buch pro Jahr lesen. (–) Das sagt alles.
C	XP-Methode	Eigeninitiative	24	24	XP schiebt die Messlatte echt hoch. Okay, Kent Beck ist ein kleines Genie. Ich habe nicht erwartet, dass alle Leute so gut sind wie er aber er hat erwartet, dass alle Leute sich bewegen. Die Leute, die mit ihm mitgemacht haben wie Martin Fowler, Bob Martin und so, die haben das gleiche Gedankengut und sie haben alle ihre Talente und das entspricht, aus meiner Sicht – das ist nicht objektiv, sondern beurteilungssubjektiv – nicht dem Standardinformatiker in der Schweiz.
C	XP-Methode	Eigeninitiative	24	24	Ein Standardinformatiker erwartet, dass die Firma alle Weiterbildungen bezahlt. Dass die Firma zahlt ist ok aber deswegen keine Weiterbildung machen, dann ist er selbst schuld, entweder macht er sie selbst oder sucht sich eine Firma, die es zahlt.
A	XP-Methode	Entscheidungsgewalt	30	30	Das Team. Ich bringe Vorschläge also quasi «he ich fände das gut» aber schlussendlich liegt die Entscheidung beim Team, ob etwas für sie funktioniert oder nicht.
B	XP-Methode	Entscheidungsgewalt	20	20	Das ist sehr unterschiedlich zwischen kleinen und grösseren Projekten, zwischen Klein- und Grossunternehmen. In Grossunternehmen wird meistens einfach entschieden, also das gilt global. Zumindest zunehmend.
B	XP-Methode	Entscheidungsgewalt	20	20	XP wird eigentlich in dem Sinne gar nicht gross gefördert. Also ich wüsste jetzt gegenwärtig kein Team wo sich noch irgendwie für XP frei entscheiden kann. In der Regel heisst es «ihr macht Scrum, weil wir alle Scrum machen» oder «ihr macht Kanban, weil wir alle Kanban machen». Das individuelle Team wurde sozusagen entmachtet. Bei kleineren Unternehmen ist das noch etwas anders, zum Beispiel bei Start-Ups, die wählen eher [ein Vorgehen] was ihnen am besten gefällt oder woran sie selbst glauben, dass es funktioniert.
D	XP-Methode	Entscheidungs-	28	28	Ich wünschte mir es wäre das Team, aber häufig ist es in einer Firma vorgeschrie-

		gewalt			ben.
C	XP-Methode	Entscheidungs-gewalt	30	30	Offiziell nach dem agilen Ansatz ist es die Entwicklungsabteilung. In der Schweiz ist es das obere Management.
E	XP-Methode	Entscheidungs-gewalt	29	29	Das Management entscheidet. Es werden politische Entscheide getroffen, die darauf abzielen ein möglichst geringes Risiko für das Management zu haben.
B	XP-Methode	Entwicklungs-fokus	14	14	Ich behaupte, dass es eine sehr entwicklerzentrierte Methode ist
B	XP-Methode	Entwicklungs-fokus	14	14	Diese [technischen Empfehlungen] kommen in den meisten anderen Methoden eher zu kurz.
B	XP-Methode	Entwicklungs-fokus	18	18	Bei XP sagen sie «ok, von dort nehmen wir die technischen Praktiken, weil sie ja sonst nirgends vorkommen – oder zu wenig» das ist eigentlich so ein bisschen ein Mauerblümchen-Dasein beziehungsweise man musses halt auch sehen, XP ist sehr entwicklerzentriert und stammt auch aus der Entwicklung.
A	XP-Methode	Erfahrung	26	26	XP braucht relativ erfahrene Leute und diese erfahrenen Leute habe ich nicht immer. Leute, die frisch vom Studium kommen ohne oder mit wenig Berufserfahrung, die tun sich schwer mit XP, weil man eine gewisse Erfahrung mit sich bringen muss.
A	XP-Methode	externer Druck	32	32	das andere, was ich sehe, ist, dass es halt aufgrund von Druck ausserhalb vom Team, also Businessdruck, Termindruck, Deadlines und so weiter, gewisse Dinge nicht mehr in dem Ausmass gemacht werden können. Dann ist es typisch, dass die Qualität darunter leidet. Also «jetzt müssen wir voll schaffen, Vollgas geben usw.» – was man ja nicht machen sollte, aber das gibt es halt einfach und dann werden plötzlich Dinge, wie das konstante Refactoring mit Aussagen wie «nein, ich muss das jetzt einfach rein machen» abgetan und man hat die Illusion, dass man es dann einfach später nachholt.
E	XP-Methode	Extremität	55	55	Bei XP wird der Regler einfach mal auf 11 gestellt. Man arbeitet quasiunter «High

					Voltage».
A	XP-Methode	Kompetenzen	28	28	Die meisten Leute haben diese Skills, um agil arbeiten zu können. Was ich als Coach liefere ist meistens der Kontext und das Feedback, um das verbessern zu können. Die Skills selbst sind nicht wahnsinnig schwierig, um es zu erlernen, das kann jeder. Auch diese XP Skills sind im Grossen und Ganzen extrem einfach zum Erlernen. Eine der wichtigsten Skills ist Kommunikation und an sich können alle kommunizieren, insbesondere wenn Entwickler unter sich sind, dann funktioniert es mit der Kommunikation relativ gut.
C	XP-Methode	Kompetenzen	24	24	Und XP oder Lean oder ähnliche Craftmanship, verlangen ein gewisses Level. Ich meine wie Lean sagt; Sie können nur eine Führungsposition haben, die den Job von den Leuten unter sich besser machen können. Ich meine nicht vielleicht besser aber die Standardführungskraft in einer IT-Firma in der Schweiz ist HS-diplomiert und wenn sie Excel bedienen kann, ist sie schon ein kleines Mirakel. Und ich sage immer meinen Kunden, es ist kein Zufall, dass die Firmen mit grossem Erfolg in den Zeitungen sind: Apple, Facebook, Twitter, Microsoft und sogar Amazon und Google; die Leute oben sind IT-affin
C	XP-Methode	Kultur	16	16	Aber kulturell ist das ein Problem. Europaweit, aber ich bewege mich [mit der Aussage] so in der Schweiz, Süddeutschland und etwas Frankreich – die Probleme sind alle ähnlich.
C	XP-Methode	Kultur	24	24	Und das ist so der XP-Mindset, der genial ist, aber er hat gewisse (), zum Beispiel Pair Programming, das steht ausser Diskussion, wenn du nicht mit den Leuten zusammenarbeiten magst; Dein Problem, pass dich an oder verlasse das Team. […] Und das ist ein kultureller Switch und sehr gute Teams in der Schweiz machen es. Sagen wir, dass der Teamdurchschnitt in der Informatik in der Schweiz nicht berauschend gut ist.
C	XP-Methode	Kultur	26	26	XP ist schon sehr fokussiert und eine Meritokratie; wenn Sie gut sind, dürfen Sie

					neue Ideen einbringen, Änderung einbringen – so auch bei Scrum, aber bei XP ist es noch stärker – und diese Meritokratie ist im Spannungsfeld mit der Kultur der Deutschschweizer. Wenn Sie in der Deutschschweiz jemanden in einem Meeting kritisieren, dann haben Sie einen lebenslangen Feind im schlimmsten Fall. Bei Meritokratie dürft ihr sagen «den Code den ich da gesehen der ist nicht umwerfend» man darf nicht die Leute kritisieren aber die Resultate schon. Der Druck auf Meritokratie zu gehen, als hinterschwelligen Wert von XP, ist in unserer Kultur nicht speziell ausgeprägt. Aber das ist wieder eine subjektive Interpretation warum XP sich nicht durchgesetzt hat und Scrum sich eher durchsetzt.
E	XP-Methode	Management-praktiken	4	4	Und dann kam noch dazu, dass Kent [Beck] gesagt hat, «du das sind alles technische Praktiken, was wir brauchen ist ‹the simplest thing that could possibly work›, um diese Projekte zu managen, Joseph, was gibt's da?» und dann habe ich mich an ein Paper von OOPSLA '95 erinnert von einem Typ namens [Ken] Schwaber über etwas namens Scrum. Ich habe Kent [Beck] das Paper gegeben und er hat gesagt «gut, nehmen wir was davon». Also haben wir Teile daraus genommen. Deswegen gibt es einige Praktiken in XP, die ein bisschen nach Scrum «stinken».
C	XP-Methode	Marketing	22	22	Und Scrum aus Sicht Marketing oder wenn wir die Einführungskurve anschauen, () hat einfach abgesahnt. Das muss man anerkennen.
A	XP-Methode	Marketing	26	26	Es hat etwas Negatives an sich, wenn man sagt «wir machen Extreme Programming» und ich als Dienstleister muss ja meine Arbeit verkaufen können und wenn der Kunde dann das Gefühl hat «Oh, Extreme Programming klingt nach kopflosem Hacken» und denkt ich verkaufe ihm etwas Unseriöses, was es zwar nicht ist, aber das ist halt das Label, das es trägt.
D	XP-Methode	Marketing	34	34	Über XP hat Kent Beck ein Buch geschrieben und ja, es hat ein Buch gegeben [aber mehr nicht]. Und Scrum ist natürlich marketingtechnisch wesentlich stärker gewesen. XP hat eher Entwickler angesprochen. Scrum hat eher Manager angesprochen oder Projektleiter, Produktmanager, die sagten sich dann «ja jetzt habe ich endlich etwas,

					wie ich alles koordinieren kann» und das ist natürlich wie eine Bombe eingeschlagen. Es klingt einfach, ist verständlich und es gab auch das Versprechen von Jeff Sutherland: «das Doppelte in der Hälfte der Zeit» und so etwas hören Manager natürlich wahnsinnig gern.
D	XP-Methode	Marketing	36	36	Es ist für all die Consultants, die sich verkaufen müssen, um Geld zu machen, einfacher zu sagen «ich bin ein Scrum-Consultant» anstatt «ich bin ein XP-Consultant» auch wenn sie es noch wären. Das ganze Marketing spielt halt dabei auch eine grosse Rolle.
C	XP-Methode	Marketing	32	32	Die Ideen waren gut, beim Marketing hat Scrum klar gewonnen.
B	XP-Methode	Motivation	30	30	Man sagt so «der gute Softwareengineer entsteht von alleine», man musses sich selbstgetrieben aneignen. Da muss man selbst Motivation zeigen.
B	XP-Methode	Priorisierung	34	34	Dann kann man noch so gute Engineering Practices haben, wenn man am Ende an den falschen Features arbeitet, die es gar nicht braucht – was ich schon so oft erlebt habe, dass man in den ersten paar Iterationen das Unwichtigste macht – dann muss ich sagen, dass wir als Software Engineer auch etwas selbst schuld sind. Dem Management, Projektmanager, Product Owner sagen wir oft nicht unsere Meinung und stehen nicht dafür ein, dass man mal wieder am wirklich Wichtigen arbeiten darf, sondern einfach das was einem gerade vorgelegt wird, weil es auf der Liste steht. Aber ist es wirklich das was den Kunden weiterbringt? Auch ein Software Engineer dürfte sich das mal von Zeit zu Zeit fragen. Und die besten Software Engineers, die ich kenne, die machen das. Die fragen dann auch den Kunden «wofür brauchst du das genau? Was für ein Problem willst du damit lösen? Was bringt dir das?» und wenn man das nicht macht, dann kümmern sie sich um den technischen Schnickschnack und technische Probleme und man merkt dann plötzlich, dass man das ganze technische Konstrukt gar nicht braucht und dann ist es schon zu spät denn dann haben sie sich schon verloren in den technischen Problemen und dem Endbenutzer wird viel zu wenig echter Wert geschaffen. XP war eigentlich so ausgelegt,

					das zu ändern, aber ich habe das Gefühl wir sind nicht viel weiter als damals vor 20 Jahren. Leider.
A	XP-Methode	Reflexion	32	32	Die grösste Schwierigkeit sehe ich in der mangelnden Zeit für Reflexion und dem mangelnden Durchhaltewillen, damit sich das etablieren könnte. Im Sinne von man probiert es mal, es funktioniert nicht wie erwartet beim ersten Mal und dann sagt man direkt «ah das funktioniert nicht» und wirft es wieder weg.
A	XP-Methode	Reflexion	32	32	Schön wäre, dass man tiefer reflektiert wieso es nicht funktioniert und vielleicht auch sagt, «ok wir probieren es jetzt mal drei Wochen».
A	XP-Methode	Reflexion	32	32	Vielleicht auch mal eine Reflexion in der grösseren Gruppe machen; «Sind wir vorwärtsgekommen? War das gut? Hat es sich gelohnt?» und dann ergibt sich das und man sieht plötzlich den Weg. Es kann aber auch sein, dass man zum Entschluss kommt; «Nein es hat keinen Wert, uns fehlt der Kontext damit es einen Wert ergibt, also müssen wir zuerst den Kontext schaffen» und das sehe ich häufig, dass es gemacht wird und nach einer Woche höre ich «das ist schwierig, anstrengend, bringt nicht» und dann lässt man es wieder sein
B	XP-Methode	technische Orientierung	24	24	Ich glaube XP als Methode ist, würde ich mal behaupten – bis auf ein paar kleine «Orte» – tot beziehungsweise es haben nur noch die technischen Praktiken überlebt. Die Methode als Ganzes mit all den Dingen, die nicht direkt mit der Programmierung zu tun haben, überlebt im Geist aber nicht als Methode.
B	XP-Methode	technische Orientierung	20	20	Weil sich sonst niemand um die Technik kümmert. Die technischen Praktiken kommen an vielen anderen Orten zu kurz.
C	XP-Methode	technische Orientierung	16	16	Das grobe Problem mit XP ist, XP ist technisch orientiert und recht diszipliniert
E	XP-Methode	technische Orientierung	8	8	XP ist eine Sammlung von Techniken, um Software in hoher Qualität zu programmieren. Es ist der technische Teil eines gesamt agilen Ansatzes und XP war die erste offizielle agile Methode. Also es geht darum zu sagen, was wir machen, warum wir

					es machen, wann wir es machen und wie wir es machen. Das Wie ist XP, das Wann ist Scrum. Und das Was und Warum sollten eigentlich gute Projektmanagement- und Marketingpraktiken sein, welche oft einfach aus dem Fenster geworfen werden.
C	XP-Methode	Unkenntnis	24	24	Hauptprobleme bei vielen Teams sind ihre Unkenntnisse und die verschiedenen Ansätze. Ich würde sagen 3/4 von allen Teams in der Schweiz machen Scrum nicht wie es im Scrum Guide steht; Also auf 5 Seiten und Scrum hat 3 Rollen, 3 Artefakten und 5 Events. Ich meine sogar Männer sollten es schaffen 11 Sachen zu memorisieren aber das funktioniert nicht.
E	XP-Methode	Unternehmens-grösse	39	39	Das Problem bei grossen Firmen ist, dass die Entfernung deiner Tätigkeit und dem Geld, das du verdienst, das Geld, das die Firma verdient, zu gross ist, das stimmt nicht so.
B	XP-Methode	Unternehmens-grösse	28	28	Und XP hat dort natürlich schon gesagt «möglichst früh releasen, sofort die Kunden [die Software] nutzen lassen» und dort sind wir schon wieder etwas im Startup-Modus. Ich glaube in der Grossunternehmenswelt hat XP keine Chance und das macht halt auch einen grossen Anteil der Entwicklungsjobs, die wir so haben aus.
C	XP-Methode	Zeitpunkt	32	32	XP die Idee ist cool, die Methode hat den Momento verpasst und wird historisch verschwinden wie Crystal oder DSDM.
C	XP-Methode	Zeitpunkt	34	34	XP war zu früh für die meisten Leute.
B	XP-Methode	Zertifizierung	28	28	Zweitens gibt es keine Zertifizierung dafür. Das heisst es gibt keine Lobby dafür. Etwas, wo in Scrum oder zumindest die Urväter von Scrum auf eine Art gut gemacht haben, aber sie wussten damals schon – ich durfte mich auch mit Ken Schwaber unterhalten, dem Erfinder von Scrum – damals waren wir ganz wenige. Ich bin der [x-te] Scrum Master in der Schweiz, und eben damals meinte der Ken «Zertifizierungen sind unser Fluch und Segen» – und recht hatte er. Wo es eine Zertifizierung gibt, dann holt man sich das Papier und kann sagen «ich bin zertifizierter Scrum Master». Es gibt keinen zertifizierten XP Programmer. Und der Vorteil, den

					das hat, ist, dass grössere Unternehmen sagen, dass sie einen Karrierepfad für ihre Entwickler oder Manager ebnen und Scrum hat dort Dinge angeboten.
B	XP-Methode	Zertifizierung	28	28	Das hat sich XP alles nicht angetan. Das ist so ein ‹grassroots-approach› geblieben, also von unten nach oben von den Entwicklern. Und weil es halt keine Lobby und keine Zertifizierungen hat ist es hart das wirklich durchzuziehen.
D	XP-Methode	Zertifizierung	34	34	Die ganze Zertifiziererei hat Scrum einen riesen Boost gegeben, nun konnte man sagen, dass man zertifizierter Scrum Master ist. Das scheint in der Branche einfach unglaublich gut zu funktionieren und so hat Scrum eigentlich XP total verdrängt.

10.7 Anhang G: Zusammenfassung aller Gründe und Zuweisung für Diskussion

Kategorie	Erhebung	Ausprägung	Einordnung		
			Being agile	Doing agile	extern/ Umwelt
Planungsspiel	Literatur	Anforderungsmanagement		x	
		Disziplin	x		
		Moderation		x	
		Organisation		x	
Total Planungsspiel			1	3	0
Prozent gerundet			**25%**	**75%**	**0%**
Kurze Releasezyklen	Literatur	Kompetenzen (fachlich)		x	
Total Releasezyklen			0	1	0
Prozent gerundet			**0%**	**100%**	**0%**
Metapher	Literatur	Erfahrung		x	
		Kultur	x		
		Unklarheit			x
	Interviews	Unklarheit			x
		Resonanz			x
Total Metapher			1	1	3
Prozent gerundet			**20%**	**20%**	**60%**
Einfaches Design	Literatur	Anleitungen		x	
		Dokumentationen		x	
		Priorisierung		x	
		Zeit		x	
	Interviews	Unternehmensgrösse			x
Total Einfaches Design			0	4	1
Prozent gerundet			**0%**	**80%**	**20%**
Testen	Literatur	Aufwand		x	
		Kompetenzen (fachlich)		x	
		Kosten			x

	Interviews	Akzeptanz	x		
		Aufwand		x	
		Ausbildung			x
		Legacy Code			x
		Mainstream			x
		Mindset	x		
		Zeitpunkt		x	
Total Testen			2	4	4
Prozent gerundet			**20%**	**40%**	**40%**
Refactoring	Literatur	Abhängigkeit (Architekturdesign)		x	
		Abhängigkeit (Unit-Tests)		x	
		Akzeptanz	x		
		Anleitungen		x	
		Durchhaltevermögen	x		
		Kompetenzen (fachlich)		x	
		Legacy Code			x
Total Refactoring			2	4	1
Prozent gerundet			**29%**	**57%**	**14%**
Paarprogrammierung	Literatur	Infrastruktur			x
		Kompetenzen (sozial und fachlich)	x	x	
		Kosten			x
		Kultur	x		
		Managementakzeptanz	x		
		Produktivität		x	
		Ressourcen			x
		Unterbrechung			x
		Verfügbarkeit			x
		Verteilte Teams		x	
		Wohlbefinden	x		

	Interviews	Akzeptanz	x		
		Disziplin	x		
		Infrastruktur			x
		Kompetenzen		x	
		Kosten			x
		Logistik			x
		Mainstream			x
		Managementakzeptanz	x		
		Paarbildung		x	
		Wohlbefinden	x		
Total Paarprogrammierung			8	5	9
Prozent gerundet			**38%**	**24%**	**43%**
Gemeinsame Verantwortlichkeit	Literatur	Mindset	x		
		Verständnis		x	
	Interviews	Mindset	x		
		Qualität	x		
Total Gemeinsame Verantwortlichkeit			3	1	0
Prozent gerundet			**75%**	**25%**	**0%**
Fortlaufende Integration	Literatur	Aufwand		x	
Total Fortlaufende Integration			0	1	0
Prozent gerundet			**0%**	**100%**	**0%**
40-Stunden Woche	Literatur	Organisation		x	
		Projektart und -grösse			x
	Interviews	Wirtschaftlichkeit			x
Total 40-Stunden Woche			0	1	2
Prozent gerundet			**0%**	**33%**	**67%**
Kunde vor Ort	Literatur	Abhängigkeit		x	
		Anwesenheit		x	
		Beteiligung		x	
		Domänenwissen		x	
		Interpretation	x		

		Verantwortung	x		
		Verfügbarkeit		x	
	Interviews	KundInnennähe		x	
Total Kunde vor Ort			2	6	0
Prozent gerundet			**25%**	**75%**	**0%**
Programmier-standards	Literatur	Akzeptanz	x		
		Dogma	x		
		Erfahrung		x	
		Organisation		x	
Total Programmierstandards			2	2	0
Prozent gerundet			**50%**	**50%**	**0%**

Kategorie	**Erhebung**	**Ausprägung**	**Einordnung**		
			Being agile	**Doing agile**	**Extern / Umwelt**
XP-Methode	Literatur	Abhängigkeit		x	
		Akzeptanz	x		
		Dokumentation		x	
		Durchhaltevermögen	x		
		Entscheidungsgewalt			x
		Implizites Wissen	x		
		Interpretation	x		
		Kommunikation	x		
		Kompetenzen (sozial und fachlich)	x	x	
		Managementpraktiken		x	
		Mindset	x		
		Motivation	x		
		Produktivität		x	
		Projektkontrolle		x	
		Skalierbarkeit		x	
		Transparenz	x		

		Verteilte Teams		x	
		Wirtschaftlichkeit		x	
	Interviews	Akzeptanz	x		
		Anpassungsfähigkeit	x		
		Begleitung		x	
		Beliebtheit	x		
		Disziplin	x		
		Dokumentation		x	
		Domänenwissen		x	
		Durchhaltevermögen	x		
		Eigeninitiative	x		
		Entscheidungsgewalt			x
		Entwicklungsfokus		x	
		Erfahrung	x		
		Externer Druck			x
		Extremität	x		
		Kompetenzen	x		
		Kultur	x		
		Managementpraktiken		x	
		Marketing			x
		Motivation	x		
		Priorisierung		x	
		Reflexion	x		
		Technische Orientierung		x	
		Unkenntnis		x	
		Unternehmensgrösse			x
		Zeitpunkt		x	
		Zertifizierung			x
Total			21	18	6
Prozent gerundet			**48%**	**41%**	**14%**

Hybride Vorgehensmodelle	Literatur	Adaption		x	
		Einschränkungen			x
		Ergänzung (Scrum / XP)		x	
		Evolution		x	
		Flexibilität	x		
		Mainstream			x
		Neophobie	x		
	Interviews	Adaption		x	
		Änderungsbereitschaft	x		
		Ergänzung (Scrum/XP)		x	
		Komfortzone	x		
		Mainstream			x
		Regulationen			x
Total			4	5	4
Prozent gerundet			**31%**	**38%**	**31%**
Schweiz	Interviews	Ausbildung	x		x
		Auslagerung		x	
		Entwicklungsfähigkeit	x		
		Konservatismus			x
		Reglementierung			x
		Wahrnehmung	x		
Total			3	1	3
Prozent gerundet			**50%**	**17%**	**50%**

Bisher erschienene Schriften

Ergebnisse von Forschungsprojekten erscheinen jeweils in Form von Arbeitsberichten in Reihen. Sonstige Publikationen erscheinen in Form von alleinstehenden Schriften.

Derzeit gibt es in den Churer Schriften zur Informationswissenschaft folgende Reihen:
Reihe Berufsmarktforschung

Weitere Publikationen

Churer Schriften zur Informationswissenschaft – Schrift 123
Herausgegeben von Wolfgang Semar
Susanne Grieder
Archive: Infrastruktur- und Bestandesnutzung durch Menschen mit Sehbehinderung oder Blindheit
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 124
Herausgegeben von Wolfgang Semar
Sophia Zimmerer
Digital Nudging im Pre-Purchase Kontext
Einfluss des Social Norm Nudge im Social Media
Chur, 2020
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 125
Herausgegeben von Wolfgang Semar
Nadine Christinger
Medienpädagogik in Schulbibliotheken
Zukünftige Rolle von Schul- und Gemeindebibliotheken im Bereich der Medienpädagogik am Beispiel des Kantons St. Gallen
Chur, 2020
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 126
Herausgegeben von Wolfgang Semar
Mirjam Nydegger
Unterrichtskonzept eines Forschungsdatenmanagement-Kurses für Mediziner im Masterstudium an der Universität Bern
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 127
Herausgegeben von Wolfgang Semar
Meret Stocker
Erlesene Räume
Eine Analyse zur Nutzungsauslastung von Lesesälen wissenschaftlicher Bibliotheken von 1990 bis heute
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 128
Herausgegeben von Wolfgang Semar
Ramona Blum
Das Medien-Image der "Grossen Vier" (Google, Apple, Facebook und Amazon - GAFA)
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 129
Herausgegeben von Wolfgang Semar
Linus Niederhauser
Digital Nudging im Pre-Purchase-Kontext der Customer Journey unter Berücksichtigung des Umweltbewusstseins
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 130
Herausgegeben von Wolfgang Semar
Colin Bolli
Impact of Digital Payment Methods on Traditional Payment Transactions
An Analysis of the Effects on the Swiss Financial Market
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 131
Herausgegeben von Wolfgang Semar
Patrik Jurkovic
Erfolgsgarant Lean-Startup Approach?
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 132
Herausgegeben von Wolfgang Semar
Sandra Rumiz
Firmenarchive in Wikimedia-Projekten
Wie Bestände von Schweizer Textilunternehmen über Wikipedia und
Wikidata auffindbar werden
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 133
Herausgegeben von Wolfgang Semar
Vanessa Seyffert
Chatbots und Semantic-Web – ein "Dream-Team"?
Einsatz semantischer Technologien in der Chatbot-Entwicklung und
Anwendung im Bibliotheksbereich
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 134
Herausgegeben von Wolfgang Semar
Mircea Obreja
Loss Aversion im E-Commerce
Moderierende Faktoren bezüglich des digitalen Loss Aversion Nudges
in der Purchase-Stage
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 135
Herausgegeben von Wolfgang Semar
Vanessa Brogli
Messinstrumente für die Untersuchung der Lesekompetenz
Wie sich Effekte auf das Lesen bei Leseförderung von Bibliotheken untersuchen lassen
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 136
Herausgegeben von Wolfgang Semar
Nichola Schwendimann
Cloud Readiness von Schweizer IT-KMU
Untersucht anhand von zwei Mikrounternehmen
Chur, 2021
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 137
Herausgegeben von Wolfgang Semar
Stefanie Moser
Homeoffice für Bibliotheksmitarbeitende von öffentlichen und wissenschaftlichen Bibliotheken
in der Schweiz während der COVID-19-Pandemie
Chur, 2021
ISSN 1660-945X

Über die Informationswissenschaft der Fachhochschule Graubünden

Die Informationswissenschaft ist in der Schweiz noch ein relativ junger Lehr- und Forschungsbereich. International weist diese Disziplin aber vor allem im anglo-amerikanischen Bereich eine jahrzehntelange Tradition auf. Die klassischen Bezeichnungen dort sind Information Science, Library Science oder Information Studies. Die Grundfragestellung der Informationswissenschaft liegt in der Betrachtung der Rolle und des Umgangs mit Information in allen ihren Ausprägungen und Medien sowohl in Wirtschaft und Gesellschaft. Die Informationswissenschaft wird in Chur integriert betrachtet.

Diese Sicht umfasst nicht nur die Teildisziplinen Bibliothekswissenschaft, Archivwissenschaft und Dokumentationswissenschaft. Auch neue Entwicklungen im Bereich Medienwirtschaft, Informations- und Wissensmanagement und Big Data werden gezielt aufgegriffen und im Lehr- und Forschungsprogramm berücksichtigt.

Der Studiengang Informationswissenschaft wird seit 1998 als Vollzeitstudiengang in Chur angeboten und seit 2002 als Teilzeit-Studiengang in Zürich. Seit 2010 rundet der Master of Science in Business Administration das Lehrangebot ab.

Der Arbeitsbereich Informationswissenschaft vereinigt Cluster von Forschungs-, Entwicklungs- und Dienstleistungspotenzialen in unterschiedlichen Kompetenzzentren:

- Information Management & Competitive Intelligence
- Collaborative Knowledge Management
- Information and Data Management
- Records Management
- Library Consulting
- Information Laboratory
- Digital Education

Diese Kompetenzzentren werden im Swiss Institute for Information Science (SII) zusammengefasst.

Impressum

Impressum

FHGR - Fachhochschule
Graubünden
Information Science
Pulvermühlestrasse 57
CH-7000 Chur

www.informationscience.ch
www.fhgr.ch
ISSN 1660-945X

Institutsleitung

Prof. Dr. Ingo Barkow
Telefon: +41 81 286 24 61
Email: ingo.barkow@fhgr.ch

Sekretariat

Telefon: +41 81 286 24 24
Fax: +41 81 286 24 00
Email: clarita.decurtins@fhgr.ch