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

Durchführung einer Umfrage-Studie zur Nutzung von Code-Reviews in der Praxis

Masterarbeit

im Studiengang Elektrotechnik und Informationstechnik

von

Hendrik Leßmann

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Prof. Dr.-Ing. Peter Werle
Betreuer: Tobias Baum
Zweitbetreuer: Prof. Dr.-Ing. Ernst Gockenbach

Hannover, 10.04.2017

Erklärung der Selbstständigkeit

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

Hannover, 10.04.17

Hendrik Leßmann

Zusammenfassung

Die Umfrage kann als Erfolg betrachtet werden. Es beteiligten sich 318 Personen aus 19 Ländern.
Bei der Verbreitung von Code-Reviews ist zu beobachten, dass von 240 Teilnehmern 186 Teilnehmer zur Zeit der Beantwortung Code-Reviews nutzen. Somit ist davon auszugehen, dass die Vorteile von Reviews bekannt sind und sich in den Teams Erfolge durch Code-Reviews eingestellt haben. 30 Teilnehmer der Umfrage unternahmen noch keine Gedanken zu Code-Reviews, obwohl viele eine hohen Nutzen vermuten. Bei 15 Teilnehmern ist der Reviewprozess eingeschlafen und ein Teilnehmer beendete bewusst Reviews.
Bei der Überprüfung der Einflüsse der Kontextfaktoren auf die Review-Effekte konnten einige Zusammenhänge durch den exakten Test nach Fisher bestätigt werden. Die statistische Auswertung ergab, dass häufige Reviews und das Verbessern der Code-Qualität im engen Zusammenhang stehen. Ebenso wurde für den Kontextfaktor, der besagt, dass Folgen ernste Konsequenzen haben und dem Review-Effekt „Finden von Defekten“, eine statistisch signifikante Beziehung nachgewiesen. Eine große und komplexe Codebase bildet mit dem Effekt des Lernens des Autors ebenso einen Zusammenhang. Viele weitere Ergebnisse können nicht als statistisch signifikant eingestuft werden, jedoch ist eine Tendenz erkennbar.
Es lässt sich festhalten, dass von 15 Entwicklerteams die Code-Reviewprozesse eingeschlafen sind. Ein Entwicklerteam entschied sich für ein bewusstes Beenden von Code-Reviews. Häufig lässt sich beobachten, dass Teams mit beendetem Reviewprozess keine festen Prozessregeln und -konventionen verwendeten. Es besteht die Vermutung, dass gesetzliche und vertragliche Regelungen sich positiv auf die Erhaltung des Reviewprozesses auswirken.

Abstract

The survey can be considered as a success. In total, there were 318 participants from 19 different countries.
Regarding the use of code reviews, it can be observed that out of 240 participants, 186 stated that they currently conduct code reviews. Therefore, it can be assumed that the benefits of conducting reviews are known and that the teams have experienced increased success rates when using code reviews. 30 participants in the survey have not yet thought about code reviews, although many suspect a high benefit. 15 participants have slowly stopped using code reviews over time and one participant terminated code reviews consciously.
When examining the influences of the context factors on the review effects, some connections could be confirmed by Fisher's exact test. The conducted statistical evaluation showed that frequent reviews and an improvement of the code quality are closely correlated. Similarly, for the context factor that states that consequences have serious consequences and the review effect 'finding defects', a statistically significant relationship was detected. A large and complex codebase is also correlated with the author's learning effects. Many other results are not statistically significant, but a tendency has been discovered.
It can be concluded, that 15 developers teams have stopped using code reviews over time. One developer team decided to deliberately stop code reviews. It can be observed that teams which have stopped using the code review process also do not make use of fixed process rules and conventions most of the time. It can be suspected that legal and contractual regulations have a positive effect on the maintenance of the review process.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Software nimmt im heutigen Leben einen hohen Stellenwert ein. Die Herausforderungen, um Software zu entwickeln und zu pflegen, steigen ständig. Auf der einen Seite wird Software immer komplexer, auf der anderen Seite steigen die Anforderungen an die Qualität. Technologie ist immer mehr Menschen zugänglich und somit wachsen auch die Absatzzahlen von Software. Ein Defekt der Software wiegt somit schwer.
Eine vollständige Vermeidung von Defekten ist ausgeschlossen, da es sich bei der Software-Entwicklung um eine menschliche Tätigkeit handelt, die somit für Defekte anfällig ist.[1] Bei einem Defekt handelt es sich um jegliche Abweichungen von der vereinbarten Anforderung an die Software vgl.[2]
Da immer mehr Software-Unternehmen auf den Markt drängen, ist es für die Unternehmen wichtiger geworden, sich mit qualitativ hochwertiger Software von der Konkurrenz abzuheben. Um die Software-Qualität zu steigern, gibt es eine Reihe von verschiedenen Verfahren. Code-Reviews eignen sich hierzu hervorragend. Bei Code-Reviews werden nicht nur Defekte beseitigt, sondern auch viele positive Effekte erzielt, etwa indem die Kompetenz der Mitarbeiter gefördert wird. Viele Entwicklerteams nutzen bereits Code-Reviews, jedoch besteht Bedarf, den Review-Prozess mit seinen Kontextfaktoren und den Review-Effekten zu erforschen. Dadurch können Defizite bei der Verwendung erkannt und Hilfestellungen für Entwickler und Forschergruppen abgeleitet werden. Durch neue Erkenntnisse zur Prozessbeendigung können Teams vermeiden, die selben Fehler zu begehen.

Nicht nur Software-Entwickler nutzen Reviews zum Erhöhen der Qualität. Ebenso in den Ingenieursdisziplinen wie Elektrotechnik werden Design-Reviews verwendet, um kontinuierlich die Einhaltung der Anforderungen zu überprüfen. Des Weiteren wird kurz im Kapitel 6 auf den Zusammenhang zwischen Reviews in der Software-Entwicklung und der Elektrotechnik eingegangen.

Diese Arbeit untersucht, wie Teams, die kommerziell Software entwickeln, Code-Reviews anwenden. Hierzu werden die verschiedenen Facetten des Review-Prozesses, sowie deren Kontextfaktoren analysiert. Ein weiteres Thema sind die Review-Effekte. Zum einen geht es hierbei um die potentiellen Ziele und zum anderen um die unerwünschten Nebeneffekte von Code-Reviews. Die Verbreitung ist auch von Interesse. Zuletzt sollen kurz die Gründe, die zur Nichtverwendung von Code-Reviews in Teams führen, untersucht werden.

[1] Laitenberger, Vegas und Ciolkowski 2002.
[2] Ebd.

1.2 Ziel der Arbeit

Es wurden bereits in einer Primärforschungsarbeit[3] 22 Interviews geführt und beschrieben, in denen 22 verschiedene Fälle von Code-Review-Gebrauch in 19 Firmen vorkommen. Aus den Daten wurde ein Klassifikationsschema zu industriellen Code-Reviews angefertigt, welches auf dem Change-based Ansatz beruht. Der Ansatz wird näher in Abschnitt 2.2 erläutert. Des Weiteren wurde anhand der Daten ein Diagramm erstellt, welches die Kontextfaktoren-, sowie die Effekte des Code-Reviews abbildet.[4]
Durch den gewonnenen Überblick über die verschiedenen Prozessvarianten, Kontextfaktoren und Review-Effekte wurden Zusammenhänge ersichtlich, die es mittels einer Sekundäranalyse zu prüfen gilt. Es werden daher in dieser Arbeit die zuvor gewonnenen Daten aus der Primärforschung genutzt, um einen Fragebogen zu entwickeln. Die neu gewonnenen Daten aus der quantitativen Befragung sollen die, in dieser Arbeit aufgestellten Forschungsfragen 3.1 beantworten und eine Hypothese aus der Primärforschung[5] prüfen.

Der Schwerpunkt der Arbeit liegt auf die Beziehungen zwischen den Kontextfaktoren und den Review-Effekten. Durch das Wissen, wie der Einfluss der Kontextfaktoren auf die Review-Effekte ist, kann der Entwicklungsprozess verbessert werden. Das Verständnis der Zusammenhänge kann auch dazu dienen, Fehler beim Code-Review zu vermeiden. Dieses Wissen ist für die Industrie und Forschergruppen von Relevanz, um beispielsweise Empfehlungen auszusprechen. Es wird versucht, den Code-Review-Prozess in einem großen Umfang, über die Forschungsfragen hinaus, zu erfassen. Außerdem wird der Frage nachgegangen, warum Entwicklungsteams die Nutzung von Code-Reviews beenden. Möglicherweise ist bereits in der Review-planung ein Mangel zu erkennen. Ist dieser bekannt, so könnte dies den Entwicklerteams helfen, diesen Mangel zu vermeiden. Es soll ebenso untersucht werden, wie verbreitet aktuell Code-Reviews sind.

1.3 Aufbau der Arbeit

Im zweiten Kapitel werden die Grundlagen zu Code-Reviews und zum Planen der Umfrage erläutert. Im dritten Kapitel wird erläutert, wie die Erstellung und die Durchführung der Umfrage verliefen. Die Ergebnisse sowie die Validierung der Ergebnisse werden im vierten Kapitel präsentiert. Auch die Beantwortung der Forschungsfragen erfolgt an dieser Stelle. Anschließend werden im fünften Kapitel in aller Kürze verwandte Ausarbeitungen diskutiert. Im sechsten Kapitel erfolgt eine Diskussion, bei der die Einschränkungen der vorliegenden Untersuchung genannt werden. Im Anhang findet sich der vollständige Fragebogen in Deutsch und Englisch.

[3] T. Baum u. a. 2016.
[4] Tobias Baum u. a. 2016.
[5] Ebd.

2 Grundlagen

2.1 Einführung in Code-Reviews

Code-Reviews sind ein Teil des Software-Entwicklungsprozesses. Sie können nicht einer einzigen Abteilung in der Entwicklung zugeordnet werden, sondern finden sich in mehreren Bereichen des Entwicklungsprozesses. Auch die Art, Code-Review zu verwenden, ist vielseitig und verschiedene Entwickler verstehen Unterschiedliches unter dem Begriff „Code-Reviews“.
Code-Reviews wurden bereits häufig untersucht. Viele dieser Artikel handeln von Inspektionen. Dieses Verfahren wurde 1976 von Michael Fagan vorgeschlagen.[6] Der grobe Ablauf hierbei lautet: Ein Team für das Review wird zusammengestellt, es erfolgt ein erstes Treffen, um Ziele zu besprechen, Gutachter lesen sich das Produkt erstmalig durch. Während das Team nach Mängeln sucht, werden die Ergebnisse protokolliert und ggf. wird der Autor befragt. Zuletzt korrigiert der Autor die gefundenen Mängel.
Ebenso haben Gilb und Graham[7] sowie Wiegers[8] Forschungen zu Inspektionen betrieben. Der zweite große Bereich, an dem geforscht wurde und wird, nennt sich Peer-Reviews. Die Akteure sind zwei Entwickler. Nachdem ein Entwickler einen Code fertig gestellt hat, sendet er diesen Code an einen zweiten Entwickler. Der zweite Entwickler prüft den Code, macht Anmerkungen zu möglichen Fehlern und sendet diese Anmerkungen an den ersten Entwickler zurück. Untersucht wurden zum Beispiel die positiven Auswirkungen von leichtgewichtigen Peer-Code-Reviews[9], Peer-Reviews in der Open-Source-Entwicklung[10] oder auch die Implikationen von Peer-Code-Reviews für die Wissensschöpfung in Informationssystemen in Entwicklungsteams[11]. Auch die IEEE-Organisation hat Reviews der klassischen Art definiert.[12]

Die Definitionen zu Code-Reviews entstammen älteren Papern. Der Prozess hat sich jedoch in den letzten Dekaden in der Industrie wesentlich geändert. Häufig werden agile Entwicklungsmethoden und Entwicklungsmethoden aus dem Open-Source-Bereich verwendet. Der Prozess ist häufig leichtgewichtig, kontinuierlich und asynchron geformt.[13]

[6] Fagan 1976.
[7] Gilb und Graham 1993.
[8] „Peer reviews in software: a practical guide“ 2002.
[9] Ratcliffe 2009.
[10] P. C. Rigby 2012.
[11] K. Spohrer 2013.
[12] IEEE 2008.
[13] Rigby und Bird 2013.

Es wird für die nachfolgenden Kapitel eine Definition verwendet, die auf aktuellen empirischen Daten einer Arbeit beruht[14]:

Definition: Code Review ist eine Software-Qualitätssicherungsaktivität mit folgenden Eigenschaften:

- Die Hauptprüfung wird von einem oder mehreren Menschen durchgeführt.
- Mindestens einer dieser Menschen ist nicht der Autor des Codes.
- Die Prüfung erfolgt hauptsächlich durch Betrachten und Lesen des Codes.
- Die Prüfung wird nach der Implementierung oder als Unterbrechung der Implementierung durchgeführt.

Die Menschen, die die Prüfung durchführen, außer der Autor, werden folgend als „Reviewer“ bezeichnet.

In den nächsten Abschnitten werden der Code-Review-Prozess, die Kontextfaktoren von Reviews, sowie die Review-Effekte dargelegt. In der Abbildung 2.1 ist zu sehen, dass die „Kontextfaktoren“ die „Beabsichtigten/Akzeptablen Niveaus der Review-Effekte“ beeinflussen. Diese wiederum beeinflussen den „Code-Review-Prozess“. Es ist zu sehen, dass die „Kontextfaktoren“ und die „Code-Review-Prozessfacetten“ sich gegenseitig beeinflussen. Ebenso beeinflussen die benutzten Review-Tools und Rollenmodelle den Prozess. Dieser Zusammenhang trägt nicht zur Beantwortung der Forschungsfragen bei und wird deshalb nicht weiter betrachtet.

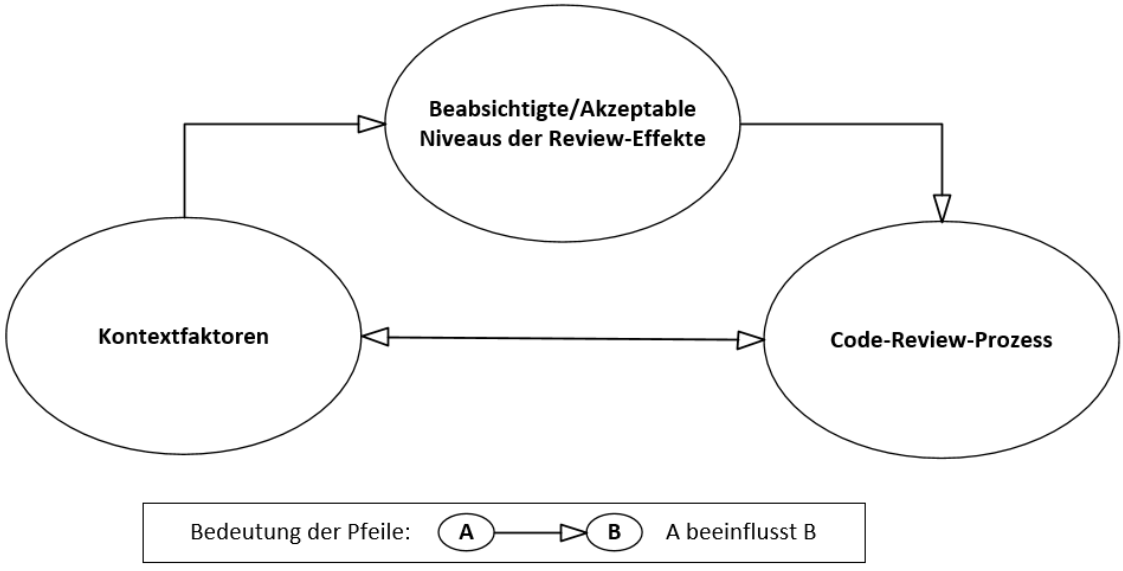

Abbildung 2.1: Beziehungsdarstellung des Code-Review-Prozesses, Kontextfaktoren und Review-Effekte.

[14] T. Baum u. a. 2016.

2.2 Die Code-Review-Prozessvarianten

Es gibt nicht nur verschiedene Definitionen zu Code-Reviews. Auch die einzelnen Aspekte des Review-Prozesses sind unterschiedlich. Diese Arbeit hält sich an eine aktuelle Studie zum Thema „Facetten-basiertes Klassifikationsschema des Code-Review-Prozesses“, da darin zum einen eine übersichtliche Klassifikation erstellt wird und zum anderen die Studie zum jetzigen Zeitpunkt aktuell ist.[15] Dieses Klassifikationsschema behandelt den Change-based-Code-Review-Prozess. Der Change-based-Reviewansatz ist wie folgt definiert: Regelmäßige, änderungsbasiertes Code-Review ist eine Art der Code-Überprüfung, die im Entwicklungsprozess des Teams oder der Organisation auf folgende Weise festgelegt ist: Jedes Mal, wenn eine „Arbeitseinheit“ als „bereit für das Review“ angesehen wird, werden alle Änderungen, die im Laufe ihrer Implementierung durchgeführt wurden, als Reviewkandidaten betrachtet. Jeder Kandidat wird dann beurteilt: Für welche Teile der Änderung ist eine Überprüfung erforderlich? Wenn ein Review benötigt wird, wartet der Reviewkandidat auf die Reviewer, die das Review zu starten.
In der Tabelle 2.1 findet sich das Klassifikationsschema, welches fünf Gruppen enthält: Prozesseinbettung, Reviewer, Überprüfung, Rückmeldung, übergreifende Facetten. Jede dieser Hauptgruppen umfasst mehrere Facetten, welche mögliche Werte enthalten können. Hierbei wurden zu dem gegenwärtigen Zustand des Code-Review-Prozesses 19 Unternehmen interviewt. Des Weiteren wurden diese Ergebnisse mit Daten von 11 Unternehmen durch eine semi-systematische Literaturrecherche erweitert. Es ist zu beobachten, dass viele Gemeinsamkeiten im Code-Review-Prozess dieser Unternehmen sowie viele Variationen der Einzelheiten auftreten.

[15] T. Baum u. a. 2016.

Tabelle 2.1: Industrielles Change-based Code-Review, Quelle: Eigene Darstellung in Anlehnung an[16]

Industrielles Change-based Code-Review	
Prozesseinbettung(A)	Arbeitseinheit(A1): VERÖFFENTLICHUNG oder STORY/ANFORDERUNG oder AUFGABE oder PUSH/PULL/KOMBINIERTES COMMIT oder EINZELNES COMMIT
	Werkzeugunterstützung / Erzwingung der Auslösung (A2): WERKZEUG oder VEREINBARUNGEN
	Öffentlichstellung des überprüften Codes (A3): POST-COMMIT-REVIEW oder PRE-COMMIT-REVIEW
	Mittel, um unberücksichtigte Änderungen von Veröffentlichungen an den Kunden zu vermeiden(A4): ORGANISATORISCH oder PRE-COMMIT-REVIEW oder VERÖFFENTLICHUNGSZWEIG
	Mittel, um sicher zu stellen, dass das Review schnell abgeschlossen wird(A5): PRIORITÄT und/oder WIP-BEGRENZUNG und/oder ZEITFENSTER und/oder ZUSTÄNDIGKEIT DES AUTORS
	Blockierung des Prozesses(A6): VOLLSTÄNDIGE WIEDERHOLUNG oder AUF DAS REVIEW WARTEN oder KEINE BLOCKIERUNG
Reviewteilnehmer(B)	Gewöhnliche Anzahl an Reviewteilnehmern(B1): 1 oder 2 oder 1 + AUTOR
	Regeln für die Anzahl an Reviewer / das Aussetzen von Reviews(B2): KOMPONENTE oder ERFAHRUNG DES AUTORS und/oder LEBENSZYKLUSPHASE und/oder ÄNDERUNGSGRÖßE und/oder PAIR-PROGRAMMING und/oder ENTSCHEIDUNG DES REVIEWERS oder ENTSCHEIDUNG DES AUTORS
	Reviewer Besetzung(B3): JEDER oder ELITE oder ALLE
	Zuweisung der Reviewer zu den Reviews(B4): PULL oder PUSH oder UNVERÄNDERLICH
	Werkzeugunterstützung für die Reviewerzuweisung(B5): KEINE UNTERSTÜTZUNG oder REVIEWER EMPFEHLUNGEN
Überprüfung(C)	Interaktion während der Überprüfung(C1): AUF NACHFRAGE oder ASYNCHRONE DISKUSSION oder TREFFEN MIT DEM AUTOR oder TREFFEN OHNE DEN AUTOR
	Zeitliche Anordnung der Reviewer(C2): PARALLEL oder SEQUENTIELL
	Spezialisierte Rollen(C3): ROLLEN oder KEINE ROLLEN
	Entdeckungshilfen(C4): CHECKLISTEN und/oder STATISCHE-CODE-ANALYSE und/oder TESTING
	Abänderung durch den Reviewer(C5): NIEMALS oder MANCHMAL
Rückmeldung(D)	Kommunikation von Fragen(D1): GESCHRIEBEN oder AUSSCHLIEßLICH MÜNDLICH oder MÜNDLICH UND ABGESPEICHERT
	Optionen zur Behandlung von Problemen(D2): BESEITIGEN und/oder ZURÜCKWEISEN und/oder VERTAGEN und/oder IGNORIEREN
Übergreifende Facetten(E)	Gebrauch von Metriken(E1): Gebrauch von Metriken ODER Kein Gebrauch von Metriken
	Spezialisierte Werkzeuge(E2): GENERELLER ZWECK oder SPEZIALISIERT

Nachfolgend werden kurz die einzelnen Facetten der Gruppen erläutert. Die Facetten stellen alle beobachteten Fälle dar, die in der Praxis vorkommen. Die Ausprägung der Facetten ist in den einzelnen Fällen höchst unterschiedlich.

2.2.1 Prozesseinbettung(A)

Die erste Gruppe an Facetten beschreibt auf welche Art und Weise der Code-Review-Prozess in den Entwicklungsprozess eingebettet ist.

Arbeitseinheit(A1)
Veröffentlichung: Reviews werden durch Änderungen, die für die Produktion oder für die Veröffentlichung bereit sind, ausgelöst.
Story/Anforderung: Reviews werden durch eine User Story /Anforderung ausgelöst, welche wohl überlegt wurde.
Aufgabe: Viele Teams teilen User-Stories in separate Implementierungsaufgaben. Wenn der gewählte Wert für die Arbeitseinheit „Aufgabe“ lautet, umfasst ein Review die Änderungen, die in solch einer Implementierungsaufgabe enthalten sind.
Push/Pull/kombiniertes Commit: Ein Review wird für jedes Source Code Management (SCM) „pull request“ oder kombinierte Typen von commit durchgeführt.
Einzelnes Commit: Reviews werden durch jedes kleinere SCM „commit“ ausgelöst.

Werkzeugunterstützung / Erzwingung der Auslösung(A2)
Werkzeug: Ein Werkzeug stellt sicher, dass für jede Arbeitseinheit ein Review-Kandidat erstellt wird.
Vereinbarungen: Wenn kein Werkzeug verwendet wird, werden Vereinbarungen und der Druck der Gruppe genutzt, um die Prozess-Einhaltung zu gewähren.

Veröffentlichung des überprüften Codes(A3)
Post-Commit-Review: Nachdem die Veränderungen an dem Code für andere Entwickler sichtbar sind, erfolgt ein Code-Review.
Pre-Commit-Review: Es erfolgt ein Review, bevor die Änderungen am Code den Hauptentwicklungs-Trunk erreicht haben.

Mittel, um unberücksichtigte Änderungen von Veröffentlichungen an den Kunden zu vermeiden(A4)
Organisatorisch: Es wird manuell geprüft, ob noch Reviews offen sind, wenn ein Release näher rückt. Dies geschieht in Kombination mit organisatorischen Mitteln, um schnell offene Reviews abzuschließen.
Pre-Commit-Review: Es werden entweder allgemein oder nur für eine gewisse Zeit vor der Veröffentlichung Pre-Commit-Reviews verwendet.
Veröffentlichungszweig: Es gibt eine ständige technische Teilung zwischen dem Entwicklungszweig / -strom und dem Veröffentlichungszweig / -strom.

Mittel, um sicher zu stellen, dass das Review schnell abgeschlossen wird(A5)
Priorität: Reviews haben eine höhere Priorität als andere Aufgaben.
WIP-Begrenzung: Das Team hat eine „Work in Progress“-Begrenzung. Diese schränkt die Anzahl an Aufgaben ein, die „bereit zum Review“ sind.
Zeitfenster: Die Reviewer reservieren sich eine bestimmte Zeit für das Code-Review.
Zuständigkeit des Autors: Der Autor sucht aktiv nach Reviewern, möglichweise nehmen Autor und Reviewer auch zusammen das Review vor.

Blockierung des Prozesses(A6)
Vollständige Wiederholung: Die Schritte des Code-Reviews folgen in einer Einheit des Arbeitszyklus.
Auf das Review warten: Die Arbeitseinheit ist blockiert, bis alle Reviewer mit dem Review fertig sind. Die Festlegung basiert auf Vertrauen und es wird nicht explizit gewartet.
Keine Blockierung: Die Arbeitseinheit kann weiterhin bearbeitet werden.

2.2.2 Reviewteilnehmer(B)

Die Facetten in dieser Gruppe beschreiben Unterschiede bezüglich der Auswahl der Reviewer.

Gewöhnliche Anzahl an Reviewteilnehmern(B1)
Gibt an, wie hoch die Anzahl der Reviewer in den meisten Fällen ist.

Regeln für die Anzahl an Reviewern / das Aussetzen von Reviews(B2)
Komponente: Das Review ist nur für bestimmte Komponenten obligatorisch, oder bestimmte Komponenten werden mit einer erhöhten Anzahl überprüft.
Erfahrung des Autors: Ausschließlich Abänderungen von unerfahrenen Entwicklern müssen gereviewt werden.
Lebenszyklusphase: In manchen Teams sind Reviews nur nahe dem Veröffentlichungszyklus obligatorisch.
Änderungsgröße: Änderungen, die kleiner als ein bestimmter Grenzwert sind, werden nicht überprüft.
Pair-Programming: Änderungen, die während einer Pair-Programmierung vorgenommen werden, müssen nicht oder mit weniger Reviewern geprüft werden.
Entscheidung des Reviewers: Die Reviewer entscheiden, ob ein Review, oder ein zweiter Reviewer nötig ist.
Entscheidung des Autors: Der Autor entscheidet, ob zusätzliche Reviewer erforderlich sind.

Reviewer-Besetzung(B3)
Jeder: Jedes Teammitglied soll für ein Review verfügbar sein. Meistens sind Ausnahmen erlaubt.
Elite: Ausschließlich erfahrene Entwickler sollen Reviews durchführen.
Alle unveränderlich: Alle Reviews werden von den selben Reviewern durchgeführt.

Zuweisung der Reviewer zu den Reviews(B4)
Pull: Bei dieser Art der Reviewer-Zuweisung bestimmt der Reviewer, wer den ausstehenden Review durchführen soll.
Push: Das Gegenstück zu „Pull“. Der Autor bestimmt, wer das Review durchführen soll.
Mix: Eine Mischung aus „Pull“ und „Push“. Der Autor lädt eine bevorzugte Teilmenge von Reviewern ein. Diese Reviewer entscheiden dann, ob sie an dem Review teilnehmen möchten.
Unveränderlich: In manchen Fällen führen die gleichen Reviewer alle Reviews für ein bestimmtes Team oder Modul durch.

Werkzeugunterstützung für die Reviewerzuweisung(B5)
Keine Unterstützung: Es gibt keine Unterstützung für die Reviewerzuweisung.
Reviewer-Empfehlungen: Es gibt eine Computerunterstützung beim Finden von passenden Reviewern.

2.2.3 Überprüfung(C)

Interaktion während der Überprüfung(C1)
Auf Nachfrage: Der Reviewteilnehmer interagiert nur auf Nachfrage.
Asynchrone Diskussion: Zusätzlich zur Interaktion „Auf Nachfrage“, Review-Anmerkungen werden sofort kommuniziert und asynchron von den Review-Teilnehmern diskutiert.
Treffen mit dem Autor: Alle Review-Teilnehmer treffen sich, möglicherweise mit dem Autor, zur Überprüfung. Häufig führt der Autor den Reviewer aktiv und erklärt den Code. Von manchen wird dies „Pair-Reviews“ genannt.
Treffen ohne den Autor: Die Reviewer treffen sich zur Diskussion über den Code. Auch diese Variante wird von einigen „Pair-Review“ genannt.

Zeitliche Anordnung der Reviewer(C2)
Parallel: Alle Reviewer arbeiten parallel. Die Nacharbeit startet, wenn alle Reviewer die Arbeit abgeschlossen haben.
Sequentiell: Nur ein Reviewer reviewt einen Code zu einem Zeitpunkt. Die Nacharbeit wird nach jedem Review erledigt.

Spezialisierte Rollen(C3)
Rollen: Die einzelnen Reviewer haben verschiedene Rollen.
Keine Rollen: Die einzelnen Reviewer nehmen die gleichen Rollen ein.

Entdeckungshilfen(C4)
Checklisten: Es wird während des Reviews eine Checkliste verwendet.
Statische Code-Analyse: Es wird während des Reviews die statische Code-Analyse verwendet, um den Reviewer darauf hinzuweisen, welche Stellen im Code weitere Kontrollen benötigen.
Testing: Es werden während des Reviews Tests ausgeführt.

Abänderung durch den Reviewer(C5)

Niemals: Der Reviewer darf oder kann aus technischen Gründen während der Kontrolle keine Änderungen am Code vornehmen.
Manchmal: Der Reviewer soll Änderungen am Code während der Kontrolle vornehmen.

2.2.4 Rückmeldung(D)

Kommunikation von Fragen(D1)
Geschrieben: Häufig werden die Fragen zum Code in geschriebener Form kommuniziert.
Ausschließlich mündlich: Die Fragen werden ausschließlich mündlich mit dem Autor diskutiert.
Mündlich und abgespeichert: Die Fragen werden mündlich kommuniziert und ebenso schriftlich notiert.

Optionen zur Behandlung von Problemen(D2)
Beseitigen: Der Code wird laut den Anmerkungen geändert.
Zurückweisen: Um Klarheit über die angemerkten Stellen zu gewinnen, wird der Autor gefragt.
Vertagen: Es wird ein Task zum Durchführen der Änderungen angelegt. Dieser Task wird priorisiert, jedoch im Einklang mit dem Entwicklungsprozess.
Ignorieren: Es wird nichts unternommen.

2.2.5 Übergreifende Facetten(E)

Gebrauch von Metriken(E1)
Gebrauch von Metriken: Es werden Metriken für Code-Reviews sammelnd und systematisch verwendet.
Kein Gebrauch von Metriken: Metriken für Code-Reviews sind weder verfügbar noch systematisch im Gebrauch.

Spezialisierte Werkzeuge(E2)
Genereller Zweck: Es werden keine spezialisierten Werkzeuge verwendet.
Spezialisiert: Es wird ein spezialisiertes Werkzeug für Code-Reviews verwendet, dass alle relevanten Funktionen kombiniert.

2.3 Die Review-Effekte

Um auszuwählen, welche Prozessfacetten in welcher Ausprägung verwendet werden, spielen häufig Review-Effekte eine wichtige Rolle[17]. Die Kontextfaktoren wiederum beeinflussen die Review-Effekte. Generell betrachten viele Entwickler Qualitätsverbesserungen als wichtig.[18] Viele Entwickler sehen die Verminderung von Mängeln am Produkt als Hauptmotivator von Reviews.[19] Effekte ohne direkte Auswirkungen werden von den Entwicklern als wenig wichtig angesehen. So werden beispielsweise Prozessverbesserungen oder Wissensteilung als irrelevant betrachtet.[20]
Aus dem Paper von Baum et al.[21] stammen die folgenden Review-Effekte. Hierbei handelt es sich um Effekte, die für Teams und Organisationen relevant sind. Relevante Effekte für einzelne Entwickler werden nicht betrachtet.
Eine Übersicht über die beobachteten Effekte finden sich in der Tabelle 2.2. Darunter erfolgt eine kurze Beschreibung der Effekte.

Tabelle 2.2: Review-Effekte, Quelle: Eigene Darstellung in Anlehnung an[22]

Review-Effekte	
Gewünschte Effekte	Verbesserung der Code-Qualität Auffinden von Defekten Wissens- /Verständniszuwachs des Reviewers Wissens- /Verständniszuwachs des Code-Autors Stärkung des gemeinsamen Verantwortungsgefühls für die Codebasis Auffinden von besseren Lösungen
Ungewünschte Effekte	Erhöhter Personalaufwand Erhöhte Durchlaufzeit im Entwicklungsprozess Kritikfähigkeit bei den Entwicklern gefordert

Die gewünschten und unerwünschten Effekte werden kurz beschrieben:

Verbesserung der Code-Qualität (gewünscht)
Es ist davon auszugehen, dass Reviews zu einer besseren Code-Qualität führen und die Wartbarkeit erhöhen. Diese Effekte resultieren meistens direkt aus dem Prüfen und Beheben von Problemen.

Auffinden von Defekten (gewünscht)
Es wird von Reviews erwartet, dass Defekte gefunden werden. Dies wird als besonders wichtig angesehen, wenn die Fehler schwer mit Tests zu ermitteln sind.

[17] Tobias Baum u. a. 2016.
[18] M. Ciolkowski und Biffl 2003.
[19] L. Harjumaa und Huttunen 2005.
[20] Ebd.
[21] Tobias Baum u. a. 2016.

Wissens- /Verständniszuwachs des Reviewers (gewünscht)
Es ist davon auszugehen, dass Reviews das Lernen des Reviewers anregen. Reviewer erlangen Wissen über die spezifische Änderung des betroffenen Moduls, aber auch mehr generelles Wissen über den Programmierstil des Autors und möglicherweise über neue Wege zum Lösen von Problemen. Auf diese Weise soll ein regulärer Review zu einer Balance von Fähigkeiten und Werten im Team führen.

Wissens- /Verständniszuwachs des Code-Autors (gewünscht)
Es ist davon auszugehen, dass Reviews das Lernen des Autors anregen. Der Autor lernt seine eigenen Schwächen kennen. Außerdem erkennt er weitere Möglichkeiten, das Problem zu lösen. Auf diese Weise soll ein regulärer Review zu einer Balance von Fähigkeiten und Werten im Team führen.

Stärkung des gemeinsamen Verantwortungsgefühls für die Codebasis (gewünscht)
Es wird von Reviews erwartet, dass sie zu einer stärkeren Wahrnehmung bezüglich des Collective-Code-Ownerships und zu steigender Solidarität führen.

Auffinden von besseren Lösungen (gewünscht)
Es wird von Reviews erwartet, dass sie neue Ideen generieren und für neue und bessere Lösungen sorgen. Auch Ideen die den vorliegenden spezifischen Code übersteigen können so entstehen.

Erfüllen von Qualitätssicherungsrichtlinien (gewünscht)
Es kann sein dass es externe Qualitätsrichtlinien gibt, welche Code-Reviews oder bestimmte Stile von Code-Reviews verlangen. Solche Richtlinien können sicherheitsrelevante Richtlinien oder Standards sein oder auf Kundenwunsch erfolgen.

Erhöhter Personalaufwand (ungewünscht)
Das Durchführen von Code-Reviews verlangt einen Investitionsaufwand, welcher für andere Aufgaben verwendet werden könnte.

Erhöhte Durchlaufzeit im Entwicklungsprozess (ungewünscht)
Das Durchführen von Reviews führt zu einem Anstieg der Zeit, bis eine Funktion implementiert ist. Diese Zeit kann aufgeteilt werden in die Zeit bis das Review durchgeführt wird, die Zeit für das Review selbst und zuletzt die Zeit, bis die gefundenen Probleme korrigiert sind.

Kritikfähigkeit bei den Entwicklern gefordert (ungewünscht)
Der Autor kann sich beleidigt (oder entmutigt) fühlen, wenn sein Code gereviewt wird und Fehler gefunden werden.

2.4 Die Kontextfaktoren

Neben den Review-Effekten formen auch die Kontextfaktoren den Prozess. Bei der Einführung oder Abänderung des Review-Prozesses muss der Prozess in den Kontext des Teams passen.[23][24] Des Weiteren müssen die Kontextgruppen „Kultur“, „Produkt“, „Entwicklungsprozess“, und „Werkzeug-Kontext“ dem Review-Prozess entsprechen. Diese Beziehungen sind in der Abbildung 2.2 zu sehen. Die Bezeichnung „Beabsichtigtes/ Akzeptables Niveau der Review-Effekte“ gibt an, dass das Team eine Lösung sucht, die „gut genug“ ist und „nicht zu schlecht“ bzgl. der unerwünschten Effekte. Einflüsse zwischen den Kontextfaktoren sind nicht enthalten. Der Abbildung folgen kurze Erklärungen zu den Elementen der Kontextgruppen.

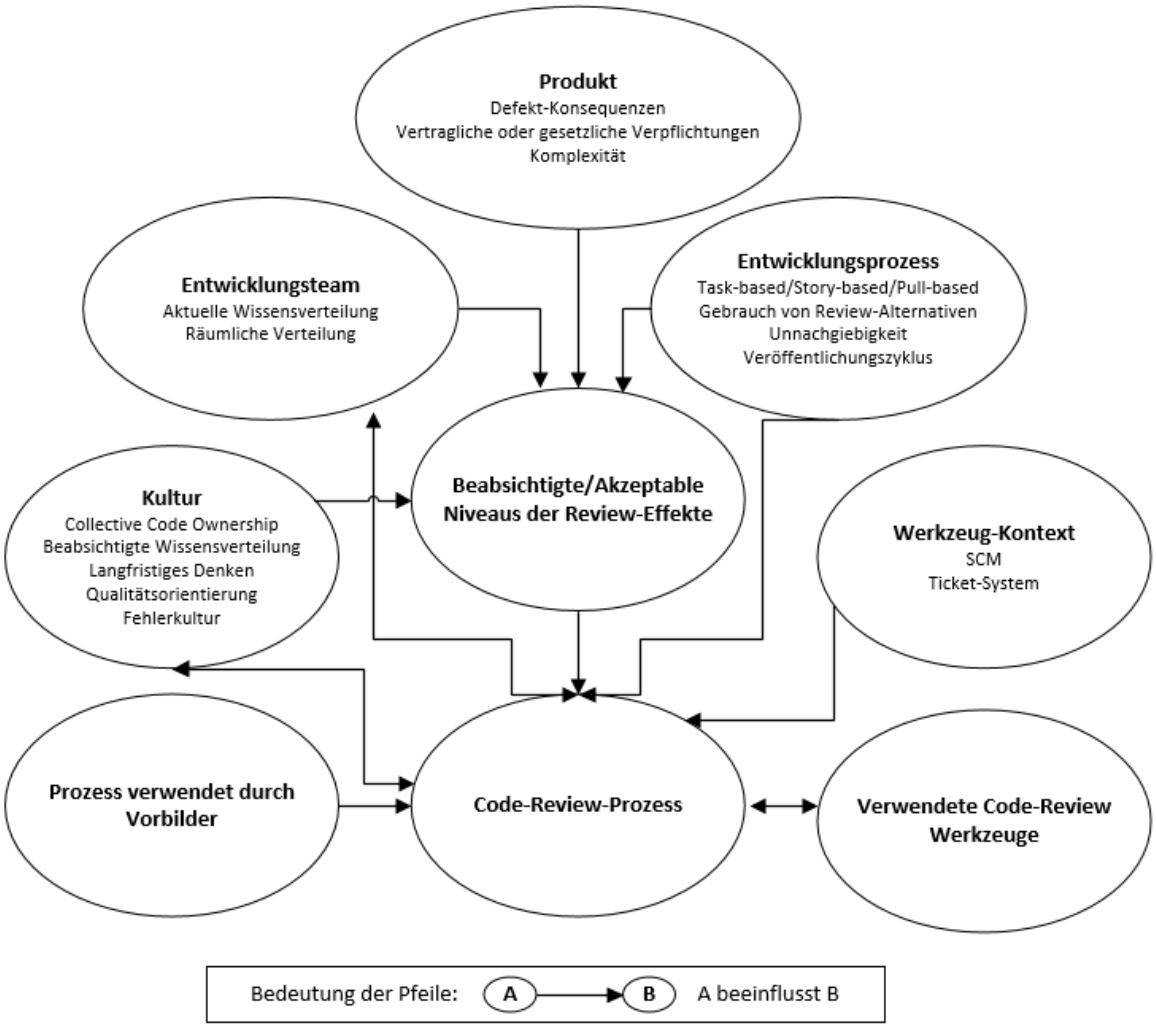

Abbildung 2.2: Die Kontextfaktoren formen den Review-Prozess, Quelle: Eigene Darstellung in Anlehnung an[a]

[a] T. Baum u. a. 2016.

[23] Mustonen-Ollila und Lyytinen 2003.
[24] L. Harjumaa und Huttunen 2005.

2.4.1 Kultur

Die Kategorie „Kultur“ fasst Handelsbräuche, Werte und Vorstellungen des Teams oder des Unternehmens zusammen.

Collective Code Ownership
In Unternehmen mit voller Collective-Code-Ownership kann und soll jeder Entwickler an jedem Teil des Codes arbeiten. An dem anderen Ende des Spektrums sind Unternehmen, bei denen ein ausgewählter Modulbesitzer bestimmte Teile am Code ändern darf.

Beabsichtigte Wissensverteilung
Die beabsichtigte Wissensverteilung ist verwandt mit dem Collective-Code-Ownership. Wenn es jedem Entwickler möglich sein soll, an jedem Teil des Codes zu arbeiten, dann muss das Wissen breit gestreut sein. Als eine Maßnahme zur Wissensverteilung müssen dann mehr Reviews durchgeführt werden.

Long-term-Denken
Eine Orientierung entgegen dem langfristigen Erfolges des Unternehmens oder des Produkts steigert die Wichtigkeit der Code-Qualität und Wissensverteilung und beeinflusst somit die Review-Prozess-Entscheidungen indirekt.

Qualitätsorientierung
Die Balance zwischen Qualität, Aufwand und time-to-market unterscheidet sich zwischen Teams und Unternehmen. Wenn Qualität als zweitwichtigste Sache betrachtet wird, sind der Aufwand und die Zeit für Reviews ein wichtiger Faktor und umgekehrt.

Fehlerkultur
Wenn Fehler als persönliche Verfehlung des Autors gesehen werden, steigert dies das Risiko, dass der Autor sich durch die Code-Anmerkungen beleidigt fühlt.

2.4.2 Entwicklungsteam

Die Kategorie „Entwicklungsteam“ fasst Faktoren zusammen, die das Entwicklungsteam charakterisieren.

Aktuelle Wissensverteilung
Die Kompetenz des Reviewers wird als ein wichtiger Faktor für das effektive Finden von Defekten angesehen. Außerdem wählen einige Teams eine Begrenzung der Reviewer-Population bei erfahrenen Teammitgliedern. Weniger erfahrene Teammitglieder bringen häufig mehr Fehler ein und profitieren stärker von dem Wissenstransfer durch Code-Reviews.

Standortverteilung
Manche Teams arbeiten am selben Standort, andere arbeiten verteilt. In verteilten Teams sind persönliche Gespräche kaum umzusetzen.

2.4.3 Produkt

Die Kategorie „Produkt“ beinhaltet Faktoren, die die Anforderungen an den Entwicklungsprozess darstellen. Diese Faktoren haben indirekt Einfluss auf den Review-Prozess.

Konsequenzen von Defekten
Wenn Defekte ernste Konsequenzen haben, ist das Finden von Defekten von hoher Wichtigkeit.

Vertragliche und gesetzgebundene Pflichten
Wenn Code-Reviews auf Grund von Verträgen oder Gesetzen angeordnet werden, muss der Review-Prozess so gestaltet sein, dass er die daraus entstehen Anforderungen befriedigt.

Komplexität
Wenn der entwickelte Code nicht sehr komplex ist, kann die beabsichtigte Code-Qualität möglicherweise ohne reguläre Code-Reviews erreicht werden.

2.4.4 Entwicklungsprozess

Der Code-Review-Prozess stellt einen Unterprozess des generellen Entwicklungsprozesses dar und wird durch andere Teile dieses Prozesses beinflusst.

Task/Story/Pull-basiert
Manche Teams teilen User-Stories in Entwicklungsaufgaben. Andere Teams verwenden User-Stories oder nur Pull-Requests basierend auf commits. Ein Team kann nur dann zwischen Tasks und Stories als Arbeitseinheit zum Reviewen entscheiden, wenn beide im Entwicklungsprozess vorhanden sind.

Gebrauch von Review-Alternativen
Es gibt alternative Techniken zum Erreichen der Ziele von Code-Reviews. Eine häufig verwendete Alternative für das Aufspüren von Defekten ist das Testing. Viele Teams verwenden die statische Code-Analyse auf einem regelmäßigen Einbindungsserver, um Wartbarkeitsprobleme zu finden. Zum Finden von besseren Lösungen diskutieren viele Teams die Anforderungen und Design-Alternativen. Pair-Programming ist eine weitere Alternative, um ähnliche Erfolge zu liefern.

Starrheit
Wenn der Entwicklungsprozess wenig starr ist, hat der einzelne Entwickler viele Freiheiten. Techniken wie Pull-Requests helfen, dass Reviews durchgeführt werden.

Veröffentlichungszyklus
Wenn Veröffentlichungen sehr häufig vorkommen, dann werden mehr Reviews weniger akzeptiert. Häufige oder auch kontinuierliche Veröffentlichungen verlangen Techniken, die verhindern, dass Code der noch nicht gereviewt wurde an den Kunden ausgeliefert wird. Dies bedeutet häufig Pre-Commit-Reviews/Pull-Requests.

2.4.5 Werkzeugzusammenhang

Werkzeuge die im Entwicklungsteam verwendet werden können die mögliche Auswahl an Prozessvarianten reduzieren

SCM
Um einen Pull-basierten Review-Prozess zu verwenden, bedarf es eines dezentralen Source-Code-Management-Systems (SCM-Systems).

Ticket-System
Der Review-Prozess kann alleine durch den Arbeitsfluss des Ticket-Systems festgelegt werden, wenn ein Ticket-System in Gebrauch ist, das einen anpassbaren Ticket-Arbeitsfluss unterstützt.

2.5 Beginn und Abbruch von Code-Reviews

Manche Teams stellen die Verwendung von Code-Reviews wieder ein. Hierzu gibt es bereits einige beobachtete Fälle bei denen es zwischen dem bewussten Beenden durch eine explizite Maßnahme und dem Einschlafen des Prozesses zu unterscheiden gilt.
Ein Grund für eine bewusste Beendigung von Code-Reviews ist eine negative Bewertung bei der Gegenüberstellung von Kosten und Nutzen.[25]
Bei der Untersuchung des unbewussten Beendens durch Entwicklerteams muss auch überlegt werden, ob dies mit Mängeln während der Einführung von Reviews zusammenhängt.
Es ist denkbar, dass bei der Einführung Fehler gemacht werden, die später zu einem Abbruch führen. Die bereits aufgeführte Tabelle 2.2 benennt unerwünschte Review-Effekte. Hier werden die Effekte „Erhöhter Personalaufwand“ und „Erhöhte Durchlaufzeit im Entwicklungsprozess“ erwähnt. Das diese Effekte auftreten lässt sich nicht verhindern, jedoch können ihre Auswirkung minimiert werden. Es ist denkbar, dass der Reviewumfang bzw. der Zeitumfang bei der Einführung falsch geplant wurde und daraufhin die Entwickler frustriert sind. Die Durchlaufzeit im Entwicklungsprozess hängt auch davon ab, wer mit welcher Priorität Reviews durchführt. Durch eine falsche Planung kann ein Nadelöhr entstehen, das zu einem unnötigen Stau im Prozess führt.
Als dritter Effekt wird „Kritikfähigkeit bei den Entwicklern gefordert“ genannt. Möglicherweise werden die Mitarbeiter nicht für die Kritikausübung in einem Review sensibilisiert, was zu Spannungen und zum unbewussten Beenden von Reviews beitragen kann.
Ein weiterer Grund für das unbewusste Beenden sind einschlafende Prozesse. Dies wird als häufigster Grund angesehen, besonders wenn Entwickler eigenständig entscheiden müssen, ob ein Review angesetzt wird.[26].

[25] Jalote und Haragopal o.D.
[26] Tobias Baum u. a. 2016.

2.6 Planung der Umfrage

Bei dem Erstellen einer Umfrage gilt es, die Methoden der empirischen Sozialforschung anzuwenden. Das generelle Vorgehen bei der Umfragenentwicklung ist dabei die Formulierung eines Forschungsproblems mit konkreter Forschungsfrage sowie die Festlegung des Forschungsdesigns, welches die Grundgesamtheit und die Stichprobe behandelt. Es muss ausgewählt werden, welche Daten vom Fragebogen erfasst und wie die entsprechenden Fragen formuliert werden sollen. Der formulierte Fragebogen muss mit Checklisten abgeglichen werden und mehrfachen Tests standhalten. Anschließend müssen potentielle Teilnehmer eingeladen werden, denen das Erhebungsinstrument bereit gestellt wird. Nach dem Erheben der Daten müssen diese aufbereitet und ausgewertet werden. Die Ergebnisse werden anschließend präsentiert.

2.6.1 Literaturrecherche und Präzisieren des Forschungsschwerpunktes

Am Anfang der Forschung sollten bereits eine Fülle wissenschaftlicher Ausarbeitungen und deren Ergebnisse bekannt sein. Dies verhindert bereits geklärte Fragestellungen erneut zu bearbeiten. Durch die Internationalisierung von Forschungsvorhaben stehen heutzutage fachspezifische Informationsdatenbanken zur Verfügung, die eine professionelle Suche ermöglichen.[27]. So lassen sich Forschungslücken aufdecken, die an bereits vorhandene Forschungen anknüpfen. Als relevant erachtete Lücken können als Forschungsfrage formuliert und untersucht werden. Ebenso ist es möglich, eigene Hypothesen aufzustellen und zu behandeln.

2.6.2 Festlegung des Forschungsdesigns

Bei der Umfragenerstellung wird zwischen qualitativer und quantitativer Forschung unterschieden. Qualitative Forschung wird bei wenig bekannten Themen eingesetzt, bei denen ein Befragter möglichst seine eigene Meinung nennt. Bei der quantitativen Forschung wird die Sicht des Forschers dargelegt, was zu viel zielgerichteteren Fragen führt. Für die quantitative Forschung wird häufig ein Fragebogen als Messinstrument verwendet.
Es ist sinnvoll, quantitative Arbeiten mit vorherigen Ausarbeitungen in qualitativer Form miteinander zu kombinieren.
Der Forscher muss festlegen, welche Art der Untersuchung er anwenden möchte. Zum einen können Hypothesen bzw. Theorien überprüft werden. Daneben werden explorative Untersuchungen, deskriptive Untersuchungen, Prognosen und Evaluationsstudien unterschieden. Explorative Studien werden durchgeführt, wenn der Erkenntnisstand in einem Untersuchungsbereich gering ist. Durch deskriptive Untersuchungen lassen sich zu einem definierten Untersuchungsgebiet umfassende Erkenntnisse gewinnen. Sind Aussagen zu zukünftigen Entwicklungen von Interesse, werden Prognosen aufgestellt. Um neu eingeführte Maßnahmen zu prüfen, finden Evaluationsstudien Anwendung.[28]
Die zu untersuchenden Themen, Forschungsfragen und aufgestellten Modelle müssen operationalisiert werden. Der für Forscher verständliche Sinn der Fragen muss den Befragten zugänglich gemacht werden. Hierbei müssen mehrdimensionale Begriffe in einzelne Dimensionen heruntergebrochen werden, um klare Fragen und Antworten zu generieren.

[27] Baur und Blasius 2014a.
[28] Baur und Blasius 2014b.

Es muss nicht nur entschieden werden, wann das Erhebungsinstrument als fertig erachtet und den Befragten zum Ausfüllen gegeben wird. Auch muss überlegt werden, nach welcher Zeit oder nach welcher Befragtenanzahl die Erhebung beendet ist.
Bei der Art der Erhebung ist zwischen telefonischer, schriftlich-postalischer und Onlinebefragung zu unterscheiden. Telefonische und schriftlich-postalische Befragungen bedeuten einen immensen Arbeitsaufwand, da die einzelnen Teilnehmer direkt kontaktiert werden. Außerdem fallen möglicherweise Telefon- oder Portokosten an. Auch die Kontaktaufnahme ist schwierig, da die Adressen bzw. die Telefonnummern verfügbar sein müssen. Bei einer Onlinebefragung hingegen muss kein direkter Kontakt zu dem Befragten erfolgen, sondern es lassen sich große Gruppen ansprechen, was Zeit und Geld spart. Die Fragen müssen nicht wie am Telefon vorgelesen werden, sondern der Teilnehmer liest und antwortet auf die Fragen selber. Auch ist die Umfrage ständig verfügbar. Die Kontaktaufnahme ist einfach, da potentielle Teilnehmer durch verschiedene Medienarten wie Online-Foren, Mailinglisten oder die direkte Weitergabe durch E-Mails angesprochen werden können.
Es stellt sich auch die Frage, wie häufig die Erhebung erfolgen soll. Eine Mehrerhebung lädt Teilnehmer mehrfach ein, um zeitliche Veränderungen zu beobachten. Dies kann saubere Daten liefern, erfordert jedoch einen größeren Aufwand, da die Antworten zu den verschiedenen Erhebungszeitpunkten denselben Personen zugeordnet werden müssen. Um die Befragten zu den weiteren Messungen einzuladen, müssen Kontaktmöglichkeiten vorhanden sein, die jedoch manche Teilnehmer nicht zur Verfügung stellen möchten.
Ein weiterer Punkt sind die Grundgesamtheit und die Stichprobe. Zuerst wird überlegt, wer an der Umfrage teilnehmen soll. In einer Vollerhebung werden alle Teilnehmer befragt, welche die Grundgesamtheit bilden. Bei einer Zufallsstichprobe wird ein Teil aller potentiellen Teilnehmeruntersucht, auch bekannt als Stichprobe. Um anhand der Stichprobe Aussagen über die Grundgesamtheit treffen zu können, müssen übereinstimmende Merkmale aus dieser Umfrage und einer Erhebung mit größerem Datensatz verglichen werden. Es wird versucht, aus der Umfrage mit dem großen Datensatz den Prozentwert eines Merkmals zu erfassen. Es kann nun aus der Stichprobe eine Teilmenge mit dem gleichen Prozentsatz des Merkmals entnommen werden. Diese Stichprobenziehung nennt sich Quota-Verfahren und hat sich für die statistische Auswertung bewährt.[29]
Aber auch eine Totalerhebung ist denkbar. Ob diese sinnvoll ist, hängt von der Größe der Teilnehmeranzahl ab. Viele Teilnehmer bedeuten einen größeren Aufwand.

2.6.3 Fragebogenentwurf

Eine große Fehlerquelle bei einer Umfrage ist der Fragebogen. Zuallererst sollte sich der Fragebogenersteller sicher sein, welche Fragen in den Fragebogen gehören und welche nicht. Die Fragen und Antworten müssen für den Teilnehmer verständlich und auch für den Umfragebetreiber gut ausgearbeitet sein, damit er die spätere Datenanalyse vollziehen kann. In den Fragebogen gehören nur Fragen, die für das Forschungsvorhaben relevant sind. Überflüssige Fragen beschäftigen nicht nur den Teilnehmer unnötig, sondern machen die Auswertung umfangreicher.

[29] Schnell, Hill und Esser 1999.

Das Formulieren von Fragen ist deshalb schwierig, da Fragen, die der Umfragebetreiber formuliert hat, für andere schwer verständlich sein können. Um die Verständlichkeit und Logik der Fragen zu prüfen, gibt es Checklisten. Hier wird jede einzelne Frage auf Kriterien wie „Lenkung zu einer bestimmten Antwort“ oder „Mehrdimensionalität der Frage“ untersucht. Dadurch lässt sich aber nicht ermitteln, ob die Probanden die Frage inhaltlich verstehen. Dazu bieten sich Vorabtests, sogenannte Pretests an. Bei der räumlichen Aufteilung der Fragen auf dem Fragebogen ist zu beachten, dass sich nicht zu viele Fragen auf einer Seite befinden sollten und die Fragen zu Themenbereichen gruppiert werden.
Die Titelseite sollte in aller Kürze folgende wichtige Punkte enthalten: Name der Institution, von der die Erhebung ausgeht, um Seriosität zu vermitteln, sowie der Grund für die Umfrage. Hierbei ist ein Deklarieren als Forschungsarbeit vorteilhaft, da es die Bedeutung hervorhebt. Ebenso sollte der Nutzen für den Befragten deutlich werden, um das Interesse zu steigern. Die Dauer der Umfrage sollte benannt werden, damit der Teilnehmer sich auf die voraussichtliche Dauer der Befragung einstellen kann. Des Weiteren soll die Zielgruppe bereits im Anschreiben genannt werden, denn wenn die potentiellen Teilnehmer wissen, dass die Befragung für sie lohnenswert ist, sind sie motiviert, den Link zu öffnen. Die Adresse, unter der die Umfrage zu erreichen ist, wird genannt und ein Dank für die Teilnahme wird formuliert. Wichtig ist bei der Weitergabe des Links auch, dass der Link nicht bei Teilnehmern landet, die nicht zur Zielgruppe gehören. Entweder wird dies bereits in dem Anschreiben formuliert oder auf der Titelseite der Umfrage. Nach dem Bedanken muss der Name des Umfragebetreibers mit E-Mail Adresse genannt werden, falls Fragen oder Probleme auftauchen oder die Teilnehmenden Anmerkungen machen möchten. Mögliche Betreuer können ebenso benannt werden.
Auch das Werben weiterer Teilnehmer bietet sich an. Mehrsprachige Fragebögen können nicht nur beim Verständnis der Fragen helfen, sondern erhöhen die Teilnehmeranzahl.
Das Wichtigste bei einem Fragebogen ist neben der Verständlichkeit die Motivation der Teilnehmer. Häufig sind die Teilnehmer beruflich stark beschäftigt und füllen den Fragebogen während ihrer Arbeitszeit aus. Bei einer geringen Teilnehmerzahl entstehen wenig Daten, wodurch sich keine ausdrucksstarken Schlüsse ziehen lassen. Der Befragte sollte sich nicht durch Pflichtfragen bedrängt fühlen oder durch weggelassene Antwortmöglichkeiten zu Falschaussagen gezwungen werden. Die Information über den aktuellen Fortschritt während der Teilnahme motiviert ebenso, die Umfrage zu Ende zu führen. Die Teilnahmedauer muss kurz gehalten werden. Als Richtwert werden häufig 10-15 Minuten angegeben.[30] Eine zu lange Dauer ist demotivierend und sorgt für hohe Abbruchquoten. Eine zeitliche Einschätzung ist schwierig, da sie von der Komplexität der Fragen und der individuellen Erinnerungsleistung der Befragten abhängt. Pretests helfen hier bei der Abschätzung. Bei der Gestaltung ist die Reihenfolge der Fragen von hoher Bedeutung. Sozial schwierige Fragen sollten zum Schluss gestellt werden, falls auf Grund der Frage ein Abbruch folgen könnte. Auch das Mischen von Fragetypen ist vorteilhaft, um Abwechslung beim Lesen zu erwirken.
Der Umfragebetreiber sollte sich Gedanken über die Standardisierung machen. Es bieten sich offene und geschlossene Fragen an. Geschlossene Fragen haben den Vorteil, dass nur begrenzte Antwortkategorien bereit stehen, was die spätere Auswertung vereinfacht. Offene Fragen

[30] Kuckartz u. a. 2009.

hingegen haben den Vorteil, dass der Teilnehmer ausgiebig auf die Frage antworten kann.
Um eine spätere Auswertung durchführen zu können, sollte sich der Umfragebetreiber vorher Gedanken zu den Antworttypen machen. Nur bestimmte Skalentypen sind mit bestimmten statistischen Methoden vereinbar. Kategorien werden durch Nominalskalen abgebildet und es können Aussagen zur Häufigkeit getroffen werden. Ordinalskalen bieten eine Rangreihe, durch die jedoch keine Aussage zu den Abständen der Werte möglich ist. Auch diese Skala bietet stark begrenzte Auswertungsmöglichkeiten. Eine Intervallskala besitzt absolute Abstände und die Korrelation oder die multiple Regression lassen sich bestimmen.
Nachdem alle Vorbereitungen zum Ausformulieren der Fragen getroffen sind, können der Literatur die ausgewählten Daten entnommen und die Fragen formuliert werden. Bei dem Entnehmen der Daten aus der Literatur ist darauf zu achten, dass komplexe Informationen eindimensional abgebildet werden müssen, so dass möglicherweise für die gewünschte Information mehrere Fragen formuliert werden. Es ist bei einem mehrsprachigen Umfragebogen vorteilhaft, den Bogen zuerst in einer Sprache zu optimieren.

2.6.4 Checklistenentwurf und Umfrageprüfung

Bei dem Entwurf von Checklisten kann auf bereits formulierte Checklisten zur Fragen- und Antwortprüfung zurückgegriffen werden, um diese dann zu kombinieren. Es ist auch erforderlich, den Fragebogen als Ganzes zu betrachten, um die Stimmigkeit der Fragen untereinander zu prüfen. Hierbei ist es wichtig, positive und negative Aussagen zu mischen, weil positive Antwortalternativen häufig negativen Antwortalternativen vorgezogen werden. Für eine gelungene Spannungskurve sollten die wichtigen Fragen im zweiten Drittel auftauchen. Mittels Filterfragen können überflüssige Fragen vermieden werden. Dies verringert die Bearbeitungszeit, weil so weniger Antworten entstehen und erhöht die Motivation des Befragten.

2.6.5 Pretesten des Fragebogens

Bei dem Pretest sollten sich der Umfragenersteller und ein Proband aus der Zielgruppe gemeinsam an dem Umfragebogen setzen. Der Proband füllt den Fragebogen aus und macht Anmerkungen, wenn Fragen unklar oder verbesserungswürdig sind. Der Umfragenersteller notiert während des Pretests Auffälligkeiten und erstellt Zeitstempel zu den einzelnen Fragen, um zeitintensive Fragen zu erfassen. In einem Gespräch nach dem Pretest werden Ungereimtheiten besprochen. Häufig sind vielfache Pretests erforderlich, um ein zufriedenstellendes Ergebnis zu erreichen. Falls ein mehrsprachiger Fragebogen geplant ist, bietet es sich an, den ausgiebig getesteten Fragebogen zu übersetzen. Nach dem Übersetzen sollten wieder Pretests durchgeführt werden, weil auch hier von Fehlern und Verständnisschwierigkeiten nach der Übersetzung auszugehen ist.

2.6.6 Onlinestellung und Einladung der Teilnehmer

Nachdem die Umfrageversionen fertig sind, kann die Umfrage online gestellt werden und Teilnehmer können eingeladen werden. Hierzu sollte sich im Vorfeld überlegt werden, wer eingeladen werden soll. Je nach Zielgruppe der Umfrage sollte verhindert werden bestimmte

Menschengruppen zu kontaktieren, um einen statistischen Fehler zu verhindern. Die Einladung sollte kurz, prägnant und höflich formuliert sein. Der Text sollte ähnlich der Titelseite, wie sie in Kapitel 2.6.3 vorgeschlagen wurde, formuliert werden.
Der Zeitpunkt der Einladung hat wenig Bedeutung bei einer internationalen Umfrage, weil das Einstellen einer Umfrage in einem internationalen Forum keiner Uhrzeit zugeordnet werden kann. Aber auch bei dem Anschreiben an eine regionale Gruppe sind keine signifikanten Unterschiede zu bemerken, ob der Startzeitpunkt in der Mitte der Woche oder am Wochenende liegt. Dies wurde in einer Studie zum Bietverhalten von Käufern beobachtet.[31]
Bei einer Onlineumfrage, bei der eine Adresse in digitaler Form vorliegt, ist es sinnvoll, ebenso digitale Verteilungswege zu nutzen, da eine mündliche Nennung einer Adresse als Fehlerquelle dient. Hierbei kann die Adresse auf verschiedenen Wegen im Internet verteilt werden, da eine zeitversetzte Teilnahme möglich ist. Als Möglichkeiten mit einer großen Erreichbarkeit an potentiellen Teilnehmern bieten sich Fachforen, Mailinglisten, oder die direkte Kontaktaufnahme via E-Mail oder Karriereportalen an.

2.6.7 Auswertung der Daten und Ergebnispräsentation

Mit Hilfe von Datenanalyse-Software können die erhobenen Daten ausgewertet werden. Statistische Verfahren dienen der Auswertung der Forschungsfragen oder Hypothesen. Neben der Hypothese gibt es die Nullhypothese. Diese besagt, dass kein bestimmter Zusammenhang besteht und ist somit eine Gegenhypothese. Es lassen sich beispielsweise Zusammenhänge zwischen zwei Variablen mit dem Pearson Chi-Quadrat-Test untersuchen. Bei dem Test werden die beobachteten Häufigkeiten mit theoretisch erwarteten Häufigkeiten verglichen. In Umfragen entstehen durch fehlende Teilnehmerantworten häufig Lücken in den Daten. Dadurch sind manchmal von zwei Variablen nicht genügend Antworten vorhanden, um diesen Test durchzuführen.
Deshalb wird häufig der exakte Test nach Fisher als Signifikanztest auf Unabhängigkeit verwendet. Er ermöglicht es, Variablen mit weniger als 5 Zellhäufigkeiten zu untersuchen. Hierzu wird, wie in der Abbildung 2.3 zu sehen, eine Kontingenztafel mit den gezählten Variablen eingetragen.

Tabelle 2.3: Kontingenztafel mit den beobachteten Häufigkeiten in der Stichprobe, Quelle: Eigene Darstellung in Anlehnung an[32]

	Prozess wird verwendet	Prozess wird nicht verwendet
Kontextfaktor vorhanden	a	b
Kontextfaktor nicht vorhanden	c	d

Es lässt sich nun eine Gesamtzahl für die Häufigkeiten bestimmen:

$$n = a + b + c + d \tag{1}$$

Und anschließend der p-Wert bestimmen:

[31] Bauer, Große-Leege und Rösger 2012.

$$p = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{a!b!c!d!n!} \tag{2}$$

Wenn der p-Wert kleiner als 0,05 ist, kann die Nullhypothese zurück gewiesen werden. Für sehr viele Validierungen wird ein Signifikanzniveau von 5% angenommen, welches 0,05 entspricht.[33]

Hieraus können schließlich Erkenntnisse gewonnen werden, die die Forschungsfrage beantworten. Neben den Ergebnissen soll auch auf die Rücklaufquote und die Stichprobe eingegangen werden. Eine Validierung der Ergebnisse ist erforderlich, um deren Gültigkeit zu beweisen.

[33] Liddell 1976.

3 Durchführung der Umfrage

Im Abschnitt 2.6 wurden die Planungspunkte einer Umfrage genannt. An dieser Stelle soll nun erläutert werden, wie die konkrete Realisierung aussieht. Der Ablauf der Umfrage ist in Abbildung 3.1 zu sehen. Die Auswertung und Ergebnispräsentation folgt im Kapitel 4.

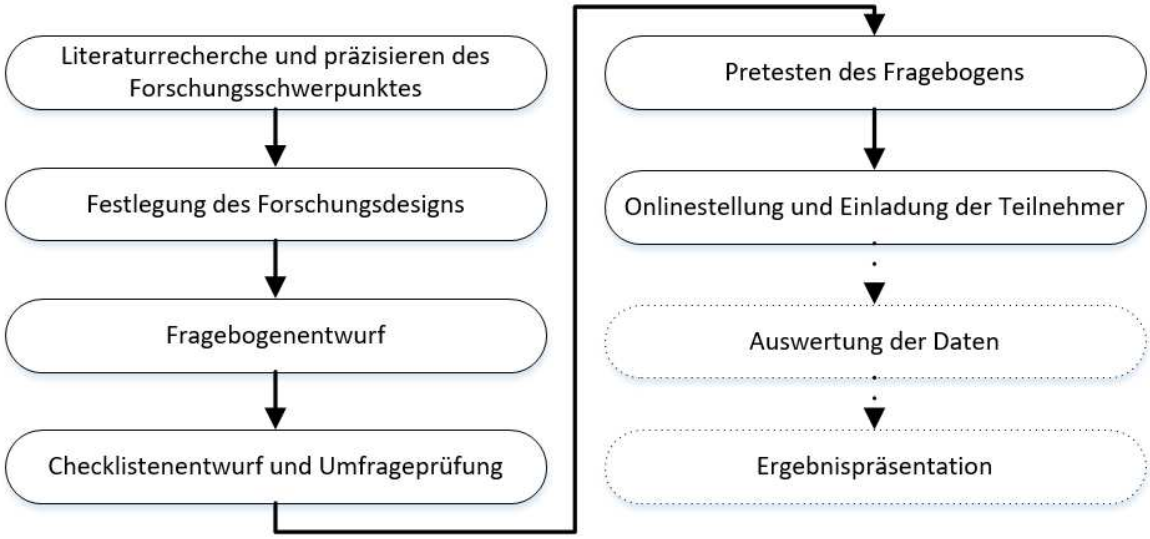

Abbildung 3.1: Ablaufdiagramm der Umfragenplanung

3.1 Literaturrecherche und Präzisieren des Forschungsschwerpunktes

Bei dem Sichten der Literatur wurden hauptsächlich Ausarbeitungen zu dem Code-Review-Prozess, den Kontextfaktoren und den Zielen von Code-Reviews berücksichtigt.
Bisherige Untersuchungen zu dem Review-Prozess lieferten Erkenntnisse, wie der Prozess aussieht und wie er zu klassifizieren ist[34], welche Optimierungsmaßnahmen erfolgen[35], ob den Beteiligten der Sinn der einzelnen Code-Review-Maßnahmen bekannt ist[36] und wie versucht wird, den Prozess zu kontrollieren[37].
Zu den Kontextfaktoren wurden folgende Literatur gesichtet: Es ist den kulturellen und sozialen Faktoren bei der Review-Einführung ein hoher Stellenwert anzuerkennen[38], die Einteilung der Kontextfaktoren in fünf Gruppen erweist sich als praktikabel[39], nichttechnische Faktoren haben bedeutende Auswirkungen auf das Reviewergebnis[40], es ist eine Unterscheidung zwischen verteilten und ortsgleichen Teams zu erkennen[41], zwischen den Review-Teilnehmern herrscht ein großes Vertrauen, Code-Review dient auch zum Wissensaustausch[42] und die Reife der Firma hat keinen Einfluss auf die Verwendung von Code-Reviews[43].

[34] Tobias Baum u. a. 2016.
[35] M. Ciolkowski und Biffl 2003.
[36] Laitenberger, Vegas und Ciolkowski 2002.
[37] Ebd.
[38] Tobias Baum u. a. 2016.
[39] Ebd.
[40] Baysal u. a. 2013.
[41] Bosu, Carver u. a. 2016.
[42] Bosu und Carver 2013.
[43] Laitenberger, Vegas und Ciolkowski 2002.

Es zeigt sich, dass schon viele Erkenntnisse zu dem Code-Review-Prozess und den Kontextfaktoren gewonnen wurden. Diese Erkenntnisse können mit einer quantitativen Befragung vertieft werden.
Zu den Zielen von Code-Reviews sind folgende Studienergebnisse bekannt: Es lassen sich sieben gewünschte und drei unerwünschte Effekte von Reviews benennen[44], Entwickler fassen Reviews als wichtig auf und geben an, dass Reviews neben dem Finden von Defekten viele weitere Vorteile bieten[45], als Hauptziel von Reviews haben Befragte die Qualitätsverbesserung angegeben[46].
Wie in Abschnitt 2.4 beschrieben, ist davon auszugehen, dass zwischen den Review-Effekten und den Kontextfaktoren eine Beziehung besteht. Eine genaue Untersuchung dieser Beziehung ist interessant, da möglicherweise Kontextfaktoren Reviewmaßnahmen begünstigen oder verhindern.
Über das Beenden von Reviews ist in der Literatur wenig bekannt. Eine recherchierte Studie besagt, dass ein großer Teil der Firmen keine Reviews verwenden.[47] Auch die Reife der Firma hat keinen Einfluss auf die Verwendung von Code-Reviews.[48] Es wird häufig keine Differenzierung von Entwicklern, die Reviews beenden und nicht einführen, getroffen.
In einer Hypothese zum nicht Verwenden von Reviews heisst es: Wenn Code-Reviews nicht verwendet werden, geschieht dies aus kulturellen und sozialen Gründen.[49] Das nicht Verwenden von Reviews bezieht sich jedoch auf Fälle, in denen Reviews nicht eingeführt werden. Als zweiten Grund werden der aufzubringende Aufwand und die Zeit genannt.[50]

Auf Basis der Recherche werden Forschungslücken erkannt und folgende Forschungsfragen formuliert:

Tabelle 3.4: Die Forschungsfragen

Nr.	Forschungsfrage
1	Wie verbreitet sind Code-Reviews?
2	Wie ist der Einfluss von den Kontextfaktoren auf die Review-Effekte?
3	Warum beenden Teams die Nutzung von Code-Reviews?

Es ist davon auszugehen, dass die empirische Untersuchung einen relevanten Beitrag zur Beantwortung dieser Fragen liefert.

Mit der ersten Forschungsfrage wird die weltweite Verbreitung von Code-Reviews untersucht und wie der Zusammenhang mit den Variablen der Entwicklung ist. Des Weiteren steht die Forschungsfrage auch im Zusammenhang mit den Forschungsfragen 2 und 3, da sie versucht, die Relevanz der Forschungsfragen zu bestätigen.

[44] Tobias Baum u. a. 2016.
[45] Bosu, Carver u. a. 2016.
[46] M. Ciolkowski und Biffl 2003.
[47] Laitenberger, Vegas und Ciolkowski 2002.
[48] Ebd.
[49] Tobias Baum u. a. 2016.
[50] Ebd.

Die Forschungsfrage 2 versucht, Zusammenhänge zwischen den möglichen Variablen der Kontextfaktoren sowie der Review-Effekte herauszufinden. Die Erkenntnisse sind sowohl für Teams, die Code-Reviews einführen wollen, als auch für Teams, die Code-Reviews verwenden interessant, um einen geeigneten Prozess zu finden oder den vorhandenen zu optimieren. Auch Teams, die sich noch keine Gedanken zu Code-Reviews gemacht haben, können sich darüber informieren, welche Kontextfaktoren sich mit welchen Effekten vereinbaren lassen. Teams, die die Nutzung von Code-Reviews abgebrochen haben, können mit einem zweiten Blick die damals getroffenen Entscheidungen überdenken und so mögliche Fehlentscheidungen ausfindig machen. Zur Beantwortung der Forschungsfrage 2 werden die Facetten der Kontextfaktoren und die Review-Effekte aus dem Kapitel 2 verwendet.

Die Forschungsfrage 3 wird explorativ gestellt, obwohl bereits eine Hypothese zum Nichtgebrauch formuliert. Diese lautet: "Wenn Code-Reviews nicht verwendet werden, geschieht dies aus kulturellen und sozialen Gründen"[51]. Das nicht Verwenden von Reviews bezieht sich dabei jedoch auf Fälle, in denen Reviews nicht eingeführt werden. Die Forschungsfrage drei zielt auf Teams, die Reviews verwendet und anschließend beendet haben. Hierbei ist zwischen dem bewussten und unbewussten Beenden von Reviews zu unterscheiden.
Es ist das Ziel, neue Erkenntnisse zum Abbrechen von Reviews zu gewinnen, um beispielsweise Teams, die vor der Einführung von Code-Reviews stehen, eine Hilfestellung zur Fehlervermeidung anzubieten.

Die Forschungsfragen 2 und 3 wurden so oder in ähnlicher Form bisher in keiner Ausarbeitung zu Code-Reviews gesichtet.

3.2 Festlegung des Forschungsdesigns

Zu den Themengebieten Review-Prozess, Kontextfaktoren und Review-Effekte liegen bereits Ausarbeitungen vor. Mit dem Wissen zu den einzelnen Begriffen in den Schemen werden Fragen und Antworten vereinfacht formuliert. Somit findet die Umfrage in quantitativer Form statt, bei der als Messinstrument ein Fragebogen verwendet wird. Die Umfrage startete am 22.02.2017. Aufgrund der ausreichend großen Menge an Teilnehmern wurde diese am 20.03.2017 beendet. Es wird eine Hypothese überprüft und versucht die aufgestellten Forschungsfragen zu beantworten. Der Fragebogen wurde online zur Verfügung gestellt, was eine einfache Weitergabe des Bogens ermöglichte. Es wurde sich für die anonyme Einmalbefragung entschieden. Ein Erinnerungsschreiben erfolgte somit nicht. Die Befragung wurde mit einer Online-Umfrage vorgenommen. Hierbei standen die Sprachen Deutsch und Englisch zur Auswahl.
Eine Umfrage stellt die Befragung einer Teilmenge der Grundgesamtheit dar. Zwischen der Teilmenge und der Grundgesamtheit muss versucht werden, einen Zusammenhang herzustellen. Durch den Zusammenhang lässt sich dann aus den Umfrageergebnisse die Grundgesamtheit abbilden. Hierzu ist zu überlegen, ob sich eine Umfrage aus dem Jahre 2016 mit ei-

[51] Tobias Baum u. a. 2016.

ner Teilnehmerzahl von 56.033 eignet, um als Grundgesamtheit angesehen zu werden.[52] Als Zielgruppe wurden Entwickler eines Online-Forums befragt. Hierbei könnten die Merkmale Teamgröße oder die Organisationsgröße dazu dienen, einen Zusammenhang der Studien herzustellen, da sie in beiden Umfragen vorkommen. Um die formulierten Forschungsfragen unter wissenschaftlichen Aspekten beantworten zu können, müssten alle oder eine repräsentative Auswahl an Entwicklern befragt werden. Dies kann und soll aus mehreren Gründen nicht geschehen. Erstens standen weder finanzielle, personelle noch zeitliche Ressourcen zur Verfügung. Aus den genannten Gründen wurde lediglich eine geringere Anzahl an Teilnehmern untersucht. Es wurden ausschließlich kostenfreie Mittel beim Werben von Teilnehmern verwendet. Hauptsächlich wurden potentielle Teilnehmer von mir und dem Betreuer kontaktiert. Zweitens sind mir keine Daten darüber bekannt, wie viele Entwickler, die exakt der Zielgruppe entsprechen, weltweit in bestimmten räumlichen Grenzen vertreten sind, um eine Aussage darüber treffen zu können, in welchem Maße die Stichprobe repräsentativ ist.
Selbst wenn die in den Grundlagen genannte Umfrage des Online-Forums verwendet wird, treten Probleme auf. Es würden dann verschiedene Zielgruppen gleichgestellt und ein großer Teil der Umfrageergebnisse würde auf Grund des Quota-Verfahrens wegfallen. Somit wären wenige Ergebnisse verfügbar und es könnten die Forschungsfragen unzureichend beantwortet werden.
Es wurden Entwickler-Teams befragt, die kommerziell Software entwickeln. Diesbezüglich wurde durch Filterfragen abgefragt, ob die Entwickler in Teams arbeiten und ob sie kommerziell Software entwickeln. Ebenso wurde aus statistischen Gründen zu Beginn und zum Schluss darauf hingewiesen, die Umfrage nicht an Teamkollegen weiterzuleiten. Denn sonst würden Stimmen für das jeweilige Team doppelt gezählt.
Außerdem besteht die Gefahr, dass durch eine geringe Anzahl die Aussagekraft verloren geht. Aus den genannten Gründen wurden sämtliche Daten aus der Erhebung entnommen.

3.3 Fragebogenentwurf

Es wurden überwiegend geschlossene Fragen verwendet, um die Auswertung zu vereinfachen. Einige Fragen haben jedoch ein Item für die Antwort „Sonstige“, wenn es für möglich gehalten wird, dass es weitere Antwortmöglichkeiten gibt. Wie bereits erwähnt, soll die Teilnahmedauer an einer Umfrage maximal 15 Minuten betragen. Es wurde bereits festgelegt, wie die Forschungsfragen lauten. Die hierzu benötigen Daten stammen aus dem Kapitel 2 und die daraus formulierten Fragen stellen eine große Menge dar. Somit fiel die Entscheidung, eine Teilung der Fragen in einen Hauptteil und einen Zusatzteil vorzunehmen. Es ist möglich, dass der Zusatzteil von wenigen Teilnehmern beantwortet wird. Deshalb müssen alle Forschungsfragen aleine durch den Hauptteil beantwortbar sein. Wie im Abschnitt 3.5 ausführlicher beschrieben, erfolgten mehrfach Pretests zum Abschätzen der Teilnahmedauer. Der Fortschritt während der Teilnahme wurde im oberen Teil der Umfrage eingeblendet und bereits von Pretest-Teilnehmern positiv bewertet.
Am Anfang des Fragebogens finden sich demografische Fragen. Fragen zu der Teilnehmermeinung zu Code-Reviews und zu negativen Review-Effekten werden am Ende des Hauptteils

[52] *Developer Survey Results 2016* o.D.

gestellt. Fragen zum Umgang mit Fehlern im Team werden als sozialkritisch eingestuft und finden sich hier ebenso. Es wurde versucht, die Fragen so zu mischen, dass der Aufbau auf die Probanden nicht trist wirkt. Es wurde eine Checkliste formuliert, die in Kapitel 3.4 zu finden ist. Der Fragebogen wurde mit den Themengruppen zu Demographie, Kontextfaktoren, dem Reviewprozess, den Review-Effekten sowie dem Nutzungsverhalten anregend formuliert. Auf jeder Seite findet sich eine ähnliche Anzahl von Fragen, diese sind jedoch nicht für den Befragten identisch, da der Fragebogen variabel erstellt ist. Filterfragen sorgen dafür, dass nur zu dem jeweiligen Teilnehmer passende Fragen erzeugt werden. So werden überflüssige Fragen vermieden und die Bearbeitungszeit verringert sich, was wiederum die Motivation des Befragten erhöht.

Der Fragebogen

Obwohl bereits im Anschreiben an den Teilnehmer die wichtigsten Informationen schon aufzuführen sind, werden diese auf der Titelseite noch einmal genannt. Es ist davon auszugehen, dass bei der Weitergabe der Umfrage nur die Adresse zu der Umfrage weitergegeben wird, ohne die Teilnahmebedingungen. Auch die Regeln zur Weitergabe werden hier genannt. Es werden, wie im Kapitel 2.6.3 aufgeführt, die Punkte für die Titel- und Abschlussseite genannt.

Der Fragebogen umfasst jeweils 89 Fragen auf Deutsch und auf Englisch. Der komplette Fragebogen findet sich im Anhang. Für die Teilung des Fragebogens in Haupt- und Zusatzteil wurde eine Filterfrage eingebaut. So kann der Teilnehmer selber entscheiden, ob er noch gewillt ist, den Fragebogen fortzuführen. Weitere Filterfragen sorgen dafür, dass jeder Teilnehmer nur für ihn passende Fragen erhält, um den Fragebogen möglichst kurz zu halten. Hierdurch ist es möglich, dass im Hauptteil maximal 33 Fragen abgefragt werden.

Die erste Frage „F1“ im Fragebogen stellt eine verpflichtende Filterfrage dar, die zur Prüfung der Zielgruppe eingesetzt wird. Der Fragebogen teilt sich in vier Themenblöcke, zu sehen unter 3.5.

Tabelle 3.5: Themenblöcke des Fragebogens

Themenblock	Thema
A	Demographische Daten
B	Kontextfaktoren
C	Prozessvarianten
D	Nichtverwendung/Verwendung
E	Review-Effekte
F	Filterfragen: F1-Zielgruppe, F2-Reviewnutzung, F3-Zusatzteil
G	Abschlussfragen

Wie in der Tabelle 3.6 zu sehen, gibt es eine Unterteilung der Teilnehmer in fünf Personengruppen. Die Unterteilung der Personengruppen erfolgt in der Umfrage durch eine verpflichtende Filterfrage „F2“.

Tabelle 3.6: Personengruppen

Personen-gruppe	Code-Review-Nutzung
G1	Es werden zur Zeit Code-Reviews verwendet.
G2	Es wurden in der Vergangenheit Code-Reviews verwendet, jedoch ist das Vorgehen inzwischen eingeschlafen.
G3	Es wurden in der Vergangenheit Code-Reviews verwendet und bewusst beendet.
G4	Es wurde sich gegen die Nutzung von Code-Reviews entschieden und deshalb noch nie verwendet.
G5	Es wurden noch keine Überlegungen zu Code-Reviews unternommen.
G6	Teilnahme aller Gruppen (G1,G2,G3,G4,G5)

Der Aufbau des Fragebogens ist in der Abbildung 3.2 zu sehen. Die Kästchen mit Füllung stellen Fragen oder Fragengruppen mit Angabe zur Personengruppe und Themengruppe im Format „Personengruppe:Themenblock“ dar. Jedes gestrichelte Kästchen stellt eine Seite im Fragebogen dar. Im linken oberen Bereich des gestrichelten Kästchens findet sich für den Hauptteil die Bezeichnung „Haupt{Seitenanzahl}“ und für den Zusatzteil die Bezeichnung „Zusatz{Seitenanzahl}“.

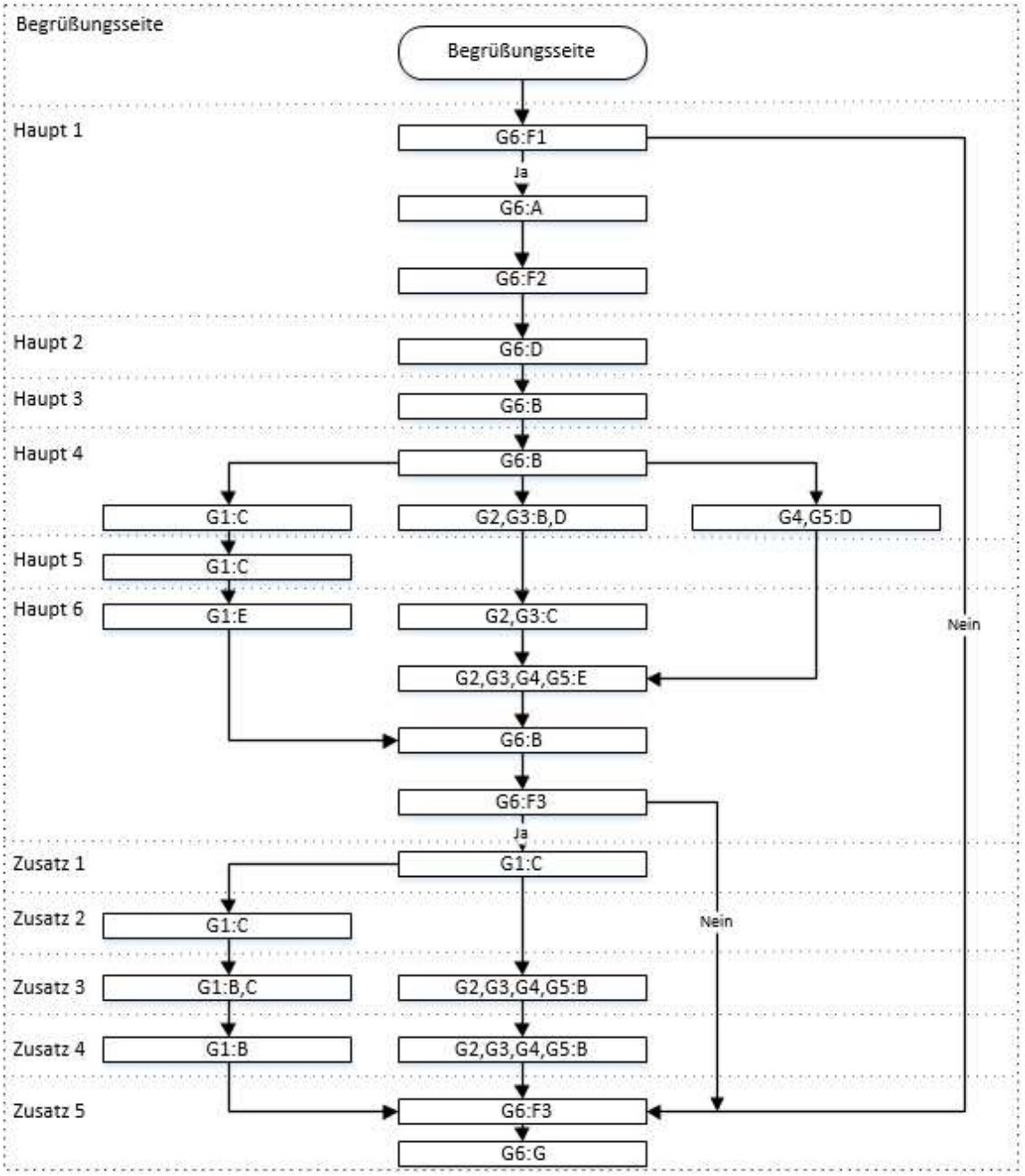

Abbildung 3.2: Schematischer Aufbau des Fragebogens, Bedeutung der Knoten: Gruppe: Themenblock

Als Plattform für die Onlineumfrage diente das Umfragenwerkzeug „Limesurvey“. Hier wurden bei Fragen mit Rangordnung eine zufällige Anordnung gewählt, um einen Biasfehler zu vermeiden.
Nachfolgend wird nur der deutsche Hauptteil des Fragebogens mit der Personengruppe G1 abgebildet. Diese Kombination stellt den Hauptteil des Forschungsvorhabens dar.

In der Umfrage finden sich unter jeder Frage Hinweise darauf, ob es sich es sich um eine Einfach- oder Mehrfachankreuzfrage handelt. Aus Platzgründen werden umfragetypisch ein Kreis „◯“ für eine Einfachauswahl, ein Rechteck „□“ für eine Mehrfachauswahl und ein Stern „ * “ für eine Pflichtangabe verwendet.

F1*: Arbeiten Sie in einem Team, das kommerzielle Software entwickelt?
◯ Ja ◯ Nein

Team: „Team“ meint in dieser Umfrage eine Gruppe von Mitarbeitern eines Unternehmens mit einem gemeinsamen Verantwortungsbereich, die unter einer gemeinsamen Bezeichnung auftritt.

A1: Wie viele Software-Entwickler arbeiten in Ihrem Team (Sie inbegriffen)?
_____

Software-Entwickler: Im Kontext dieser Frage zählt jedes Team-Mitglied, das Code verfasst, als Software-Entwickler.

A2: In welchem Bereich entwickelt Ihr Team vorwiegend Software?
◯ Closed-Source-Entwicklung ◯ Open-Source-Entwicklung

A3: In welchem Land befindet sich Ihr Arbeitsplatz?
_____

F2*: Verwenden oder verwendeten Sie Code-Reviews?
◯ Es werden zur Zeit Code-Reviews verwendet.
◯ Es wurden in der Vergangenheit Code-Reviews verwendet, jedoch ist das Vorgehen inzwischen eingeschlafen.
◯ Es wurden in der Vergangenheit Code-Reviews verwendet und bewusst beendet.
◯ Es wurde sich gegen die Nutzung von Code-Reviews entschieden und deshalb noch nie verwendet.
◯ Es wurden noch keine Überlegungen zu Code-Reviews unternommen.

Code-Review ist eine Technik zur Qualitätssicherung von Software, für die folgendes gilt:
- Die Hauptprüfung wird von einem oder mehreren Menschen durchgeführt.
- Mindestens einer dieser Menschen ist nicht der Autor des Codes.
- Die Prüfung erfolgt hauptsächlich durch Betrachten und Lesen des Codes.
- Die Prüfung wird nach der Implementierung oder als Unterbrechung der Implementierung durchgeführt.

Die Menschen die die Prüfung durchführen, außer der Autor, werden folgend als „Reviewer“ bezeichnet.

A4: Wie groß ist Ihre Organisation?

	1 - 10	11 - 25	26 - 50	51 - 100	101 - 250	251 - 500	501 - 1000	1001 - 10000	10001 oder mehr
Mitarbeiter	◯	◯	◯	◯	◯	◯	◯	◯	◯

A5: Was ist das Hauptgeschäftsfeld Ihres Teams?
◯ Standard-Software
◯ In-house-IT
◯ Contractor/Auftragsentwicklung
◯ Software as a Service (SaaS)
◯ Sonstiges: _____

A6: Welche Rolle wird Ihnen am häufigsten zugewiesen?
◯ Projekt-Manager
◯ Software-Architekt
◯ Entwickler
◯ Tester
◯ Administrator/ IT-Betrieb
◯ Qualitätsbeauftragter
◯ Sonstiges: _____

A7: Wie erfolgt in Ihrem Team der Softwareentwicklungsprozess?
◯ Agile Softwareentwicklung (Softwareentwicklung erfolgt mit wenigen Regeln, meist iterativ und mit geringem bürokratischem Aufwand).
◯ Ad hoc (Softwareentwicklung als improvisierte Handlung, die speziell für einen Zweck entworfen wird).
◯ Klassische Entwicklung (Softwareentwicklung erfolgt durch Anlehnung an ein Vorgehensmodell mit größerem bürokratischen Aufwand).
◯ Sonstiges _____

B1: Soll jeder Entwickler in Ihrem Team an jedem Teil des Codes arbeiten können?
◯ Ja, jeder Entwickler soll überwiegend an jedem Teil des Codes arbeiten können. (Collective-Code-Ownership)
◯ Nein, jeder Entwickler soll überwiegend nur in seinem eingeschränkten Zuständigkeitsbereich arbeiten.

B2: Welches Ziel hat Ihr Team in Bezug auf die Wissensverteilung? Die Teammitglieder sollen überwiegend als [..]
◯ ..Generalisten arbeiten.
◯ ..Spezialisten arbeiten.

B3: Strebt Ihr Team eher einen kurzfristigen oder einen langfristigen Produkterfolg an?

kurzfristiger	kurzfristiger bis mittelfristiger	mittelfristiger	mittelfristiger bis langfristiger	langfristiger
◯	◯	◯	◯	◯

B4: Arbeitet die Mehrheit der Entwickler Ihres Teams an verteilten Standorten (z.B. arbeiten Ihre Teammitglieder in verschiedenen Städten)?

◯Ja ◯Nein

B5: Wie lange dauert der Veröffentlichungszyklus in Ihrem Team im Durchschnitt? Das Intervall zwischen zwei Veröffentlichungen unserer Software ist [..]

◯ .. kontinuierlich

◯ .. regelmäßig, angegeben im rechten Feld mit folgender Anzahl an Wochen:

◯ .. regelmäßig, angegeben im rechten Feld mit folgender Anzahl an Monaten:

◯ .. unregelmäßig

B6: Welches Source Code Management System (SCM) nutzen Sie?

☐ Microsoft Team Foundation Server
☐ Git
☐ Subversion
☐ Rational ClearCase
☐ Helix
☐ Mercurial
☐ Es wird kein SCM verwendet
☐ Sonstiges: _____

B7: Welchen Folgen kann ein Softwarefehler in Ihrem Team haben? Ein Softwarefehler kann: [..]

☐ ..die menschliche Gesundheit oder das Leben bedrohen.
☐ ..zu einem Systemausfall führen.
☐ ..sich auf das Geschäft Ihres Unternehmens auswirken.
☐ ..zu finanziellen Verlusten führen.
☐ ..die Umwelt beeinflussen.
☐ ..rechtliche Konsequenzen haben.
☐ ..zu einer Funktionsbeeinträchtigung des Systems (Service) führen.
☐ ..sich auf den Ruf des Unternehmens auswirken.
☐ Sonstiges: _____

C1: Wie wird festgelegt, welcher Code einem Review unterzogen wird?
◯ Das Review erfolgt auf Basis von Änderungen am Source-Code (z.B. für Pull Request, User-Story, Task oder Requirement)
◯ Das Review erfolgt für ein gesamtes Modul/Paket/Klasse
◯ Sonstiges: _____

C2: Nachdem Änderungen am Code vorgenommen wurden [..]
◯.. ,werden diese für unbeteiligte Entwickler sichtbar gemacht. Anschließend erfolgt das Code-Review. (Post-Commit-Review).
◯.. erfolgt ein Code-Review. Erst anschließend werden die Code-Änderungen für unbeteiligte Entwickler sichtbar gemacht. (Pre-Commit-Review, z.B Pull-Request).

C3: Werden in Ihrem Team wenige große Code-Reviews (mit großem Umfang) oder viele kleine Code-Reviews (mit kleinem Umfang) durchgeführt?
◯ Sehr großer Umfang (z.B. ein Review für alle Änderungen eines Releases)
◯ Großer Umfang (z.B. ein Review für jede User-Story)
◯ Mittelgroßer Umfang (z.B. ein Review für jede Entwicklungsaufgabe)
◯ Kleiner Umfang (z.B. ein Review für jeden Pull/Push)
◯ Sehr kleiner Umfang (z.B. ein Review für jedes Commit)

C4: Wie viele Personen (ohne den Verfasser des Codes) nehmen meistens am Review teil?
[ ]

C5: Wie lange führt Ihr Team bereits Code-Review durch?

	Weniger als 1 Monat	1-3 Monate	4-6 Monate	7-12 Monate	1-2 Jahre	2-5 Jahre	über 5 Jahre
Verwendungsdauer von Code-Review:	◯	◯	◯	◯	◯	◯	◯

C6: Wie entscheidet sich in den meisten Fällen, ob ein Code-Review angesetzt werden soll?
◯ Der Reviewer entscheidet von Fall zu Fall, ob ein Review stattfinden soll.
◯ Der Autor entscheidet von Fall zu Fall, ob ein Review stattfinden soll.
◯ Ein Manager entscheidet von Fall zu Fall, ob ein Review stattfinden soll.
◯ Es gibt feste Regeln/Konventionen, ob ein Review stattfinden soll.
◯ Sonstiges: _____

C9: Wie geschieht in Ihrem Team die Zuweisung von Reviewern zu Code-Reviews?
◯ Die gleichen Reviewer führen alle Reviews für ein bestimmtes Team oder Modul durch.
◯ Der Autor lädt eine Gruppe von Reviewern ein. Diese Reviewer wählen dann aus den für sie ausstehenden Reviews.
◯ Der Autor bestimmt, wer das Review durchführt.
◯ Der Reviewer wählt zwischen allen ausstehenden Reviews aus.
◯ Sonstiges: _____

C10: Welche Kommunikationsformen erfolgen in Ihrem Team während des Reviews am häufigsten?
◯ Die Kommunikation zwischen den Beteiligten erfolgt nur bei Bedarf, z.B. bei Rückfragen.
◯ Review-Anmerkungen werden sofort kommuniziert und asynchron diskutiert, z.B. per E-Mail oder in einem Review-Tool.
◯ Die Reviewer treffen sich für das Review mit dem Autor.
◯ Die Reviewer treffen sich für das Review, allerdings ohne den Autor.

C11: Wird in Ihrem Team der Code überwiegend parallel oder nacheinander von den Reviewern geprüft, wenn mehrere Reviewer an einem Review beteiligt sind?
◯ Die Prüfung durch die einzelnen Reviewer erfolgt parallel. Korrigiert wird, wenn alle Reviewer fertig sind.
◯ Die Prüfung durch die einzelnen Reviewer erfolgt nacheinander. Korrigiert wird nach jedem Reviewer.
◯ Es gibt bei uns niemals mehr als einen Reviewer.
◯ Sonstiges: _____

C12: Ändern Reviewer in Ihrem Team beim Entdecken von Fehlern den Code eigenständig?
◯ Reviewer dürfen den Code ändern und tun dies häufig.
◯ Reviewer dürfen den Code ändern und tun dies gelegentlich.
◯ Reviewer dürfen den Code ändern, aber tun dies nicht.
◯ Reviewer dürfen den Code nicht ändern.
◯ Reviewer können den Code technisch nicht ändern.
◯ Sonstiges: _____

C13: Wie teilt der Reviewer dem Autor die gefundenen Fehler mit?
◯ Hauptsächlich schriftlich.
◯ Hauptsächlich mündlich.
◯ Die Probleme werden hauptsächlich mündlich kommuniziert, aber zusätzlich schriftlich festgehalten.

C14: Welche Tools verwenden Sie für Code-Reviews?
- ☐ Gerrit
- ☐ Atlassian Crucible
- ☐ Atlassian Bitbucket Server (früher: Stash)
- ☐ JetBrains Upsource
- ☐ SmartBear Collaborator
- ☐ ReviewClipse
- ☐ Microsoft CodeFlow
- ☐ GitHub Pull Requests
- ☐ Phabricator
- ☐ Klocwork Cahoots
- ☐ Allgemeine Softwareentwicklungstools (IDE, SCM, Ticket-System/bug tracker)
- ☐ Sonstiges: _____

E1: Welche der folgenden potentiellen Ziele des Codes-Reviews haben für Ihr Team die höchste Priorität?
Bitte ordnen Sie die Ziele mittels „drag and drop“ oder Doppel-Klick in „Ihre Rangfolge“ ein. Beginnen Sie oben in der Rangfolge mit dem am stärksten angestrebten Ziel.

	Ihre Rangfolge:
Verbesserung der Code-Qualität	
Finden von Defekten	
Wissens- / Verständniszuwachs des Reviewers	
Wissens- / Verständniszuwachs des Code-Autors	
Stärkung des gemeinsamen Verantwortungsgefühls für die Codebasis	
Finden von besseren Lösungen	
Erfüllung von Qualitätsmanagement-Vorgaben	

E2: Code-Reviews können manchmal unerwünschte Nebeneffekte haben.
Bitte ordnen Sie die Effekte mittels „drag and drop“ oder Doppel-Klick in „Ihre Rangfolge“ danach, wie problematisch sie in Ihrem Team sind. Beginnen Sie oben in der Rangfolge mit dem problematischsten Nebeneffekt.

	Ihre Rangfolge:
Erhöhter Personalaufwand	
Erhöhte Durchlaufzeit im Entwicklungsprozess	
Kritikfähigkeit bei den Entwicklern wird gefordert	

B11: Kann Ihr Team den Code-Review-Prozess eigenständig anpassen?

	trifft nicht zu	trifft eher nicht zu	teils-teils	trifft eher zu	trifft zu
Zustimmung:	◯	◯	◯	◯	◯

Definition Eigenständigkeit: Es können z.B. ohne Absprache mit einer höheren Personalebene Entscheidungen wie „Regelung zur Revieweranzahl“ getroffen werden.

B12: Wie ist der Umgang mit Fehlern (z.B. fehlerhaftes Handeln, Planen, Denken) in Ihrem Team? Welche der folgenden Situationen kommen häufiger vor?

☐ Es werden gemachte Fehler im Team offen angesprochen.
☐ Fehler werden gemeinsam im Team analysiert, um daraus zu lernen.
☐ Wenn durch Fehler Probleme entstehen, stehen alle im Team für einander ein.
☐ Fehler werden verschwiegen.
☐ Vorgesetzte schieben Schuld für eigene Fehler auf andere Mitarbeiter.
☐ Bei einem Fehler wird nicht nach der eigentlichen Ursache gesucht, sondern nach einem Schuldigen.
☐ Bestimmte Aufgaben werden einer Person nicht mehr zugeteilt, weil die Person in der Vergangenheit Fehler gemacht hat.

F3*: Vielen Dank für Ihre Mühen! Sie haben das Ende des Hauptteils erreicht. Haben Sie noch ca. 6-9 Minuten Zeit, weitere Fragen zu beantworten?

◯ Ja ◯ Nein

3.4 Checklistenentwurf und Umfrageprüfung

Es wurden zwei Checklisten erstellt, mit denen die Fragen der Umfrage auf die Kriterien Korrektheit, Verständlichkeit usw. geprüft wurden. Nur wenn alle Fragen aus den Checklisten für jede Frage bzw. Antwort mit ja beantwortet werden konnten, wurde die Frage in die Umfrage aufgenommen.

Die Prüffragen für die Fragen und Antworten stammen aus einem Kurzleitfaden mit einer Sammlung an Fachbüchern[53][54][55][56], sowie zwei weiteren Büchern[57][58]:

[53] Diekmann 2008.
[54] Mummendey und Grau 2008.
[55] Porst 2006.
[56] Schumann 2012.
[57] Jacob, Heinz und Décieux 2013.
[58] Höpflinger 2003.

Tabelle 3.7: Checkliste für die Fragenformulierung

Frage	trifft zu ✓
Wird die Frage zum beantworten der Forschungsfrage benötigt?	
Beinhaltet die Frage alle wichtigen Aspekte und ist die Frage ein vollständig formulierter Satz?	
Ist die Frage einfach, kurz, verständlich und präzise formuliert?	
Wurden so wenig wie möglich Fachbegriffe verwendet?	
Wurde nach Möglichkeit auf wertbesetzte Begriffe verzichtet, die einen Beigeschmack haben?	
Ist die Frage eindimensional, d.h. wird mit ihr nur ein Aspekt abgefragt?	
Ist die Frage nicht suggestiv formuliert, so dass der Befragte nicht in eine bestimmte Richtung gelenkt wird?	
Falls es sich bei der Frage um einen Teil einer Fragebatterie handelt und sie sehr lang ist, wurden die Items in unterschiedlichen Richtungen gepolt, um Antworttendenzen zu vermeiden?	
Wurde nach Möglichkeit auf hypotetische Fragen verzichtet?	
Passt die Formulierung der Frage zu den Antworten?	
Ist die Frage frei von doppelter Verneinung?	
Enthält die Frage keine Kausalkonstruktion?	
Ist die Frage frei von mehrdeutigen Begriffen, so dass Missverständnisse unwahrscheinlich sind?	
Sind alle möglichen Antworten vorhanden, so dass die Frage beantwortet werden kann?	
Ist die Frage, falls möglich, geschlossen angelegt?	
Ist die Frage frei von Widersprüchen?	
Wurde die Komplexität tief gehalten, um den Befragten nicht zu überfordern, wurde beispielsweise eine komplizierte Frage auf mehrere Fragen aufgeteilt?	
Ist die Frage in einer zielgruppenadäquaten Sprache formuliert?	
Hat die Zielgruppe wahrscheinlich das notwendige Wissen, um die Frage beantworten zu können?	
Erfordert die Frage eine zumutbare Erinnerungsleistung des Befragten?	
Hat die Frage mit der Lebenswirklichkeit der Befragten zu tun?	
Ist die Frage unter dem sozialen Gesichtspunkt zumutbar?	
Ist die Frage neutral formuliert, so dass keine Reizwörter vorkommen?	

Tabelle 3.8: Checkliste für die Antwortformulierung

Frage	trifft zu ✓
Sind die Antwortkategorien trennscharf, vollständig und präzise?	
Ist die Anzahl der Antwortkategorien überschaubar?	
Sind die Antwortkategorien auf Alternativfragen eindimensional?	
Falls eine quantifizierende Skala anwendbar ist, wurde diese z.B. einer Likert Skala vorgezogen?	
Falls nur eine Antwort vorgesehen ist: Schließen sich die Antworten gegenseitig aus?	
Ist bei zusammengefassten Antwortvorgaben die Zusammenfassung sinnvoll?	
Ist die Anzahl an Antworten ungerade, damit die Frage auch neutral beantwortet werden kann?	
Wurden bei Multiple-Choice-Fragen die Antwortvorgaben wortwörtlich von der Frage wiederholen?	
Ist bei Fragen mit zeitlicher Größe die Zeiteinheit in der Antwort vorgegeben?	
Bei langen und komplizierten Antwortvorgaben: Wurde dem starken Einfluss der Reihenfolge der Antworten Beachtung geschenkt?	
Enthalten die vorgegebenen Antwortmöglichkeiten alle relevanten Antwortmöglichkeiten und wurde, falls es erforderlich ist, die Antwortkategorie „Sonstige“ integriert?	
Sind die Skalen der Antwortmöglichkeiten symmetrisch verteilt, sprich, ist die Anzahl von negativen und positiven Antworten identisch, um die Antwort nicht in eine Richtung zu lenken?	

Übergeordnete Punkte bei der Gestaltung

Es ist auch erforderlich, den Fragebogen als Ganzes zu betrachten, um eine Stimmigkeit innerhalb der Fragen zu prüfen. Es wurden positive und negative Aussagen gemischt, weil positive Antwortalternativen häufig negativen Antwortalternativen vorgezogen werden. Für eine gelungene Spannungskurve wurden die wichtigen Fragen im zweiten Drittel gestellt werden.

3.5 Pretests des Fragebogens

Wenn der Fragebogen verteilt wird, ist es nicht mehr möglich, Änderungen vorzunehmen. Eine Änderung, während die Teilnehmer Zugriff auf die Umfrage haben, würde bedeuten, dass die Teilnehmer unterschiedliche Fragen und Antworten bekämen. Die Ergebnisse ließen sich nicht auswerten, bei unterschiedlichen Fragen. Es ist auch möglich, dass eine fehlende Antwort, die weiterführende Pfade stellt, zu einem Komplettausfall eines Pfades führt.
Um diese Probleme zu umgehen, bietet es sich an, mehrfach Pretests mit verschiedenen Teilnehmern durchzuführen. Für die Umfrage wurden 9 Pretests mit 6 Personen durchgeführt. Eine Auflistung findet sich in Abbildung 3.9. Bei den Treffen mit den Pretestern wurden Protokolle während der Sitzung angefertigt und teilweise mit dem Betreuer besprochen. Komplizierte Fragen konnten so identifiziert und bei der Struktur eine Vereinfachung vorgenommen werden.
Nach vielfachen Verbesserungen des Fragebogens wurde dieser als fertig angesehen. Eine Übersetzung ins Englische folgte, um eine größere Masse an Teilnehmern zu gewinnen. Es wurden hierbei 2 Pretests durchgeführt. Gefundene Fehler mussten diesmal in Englisch und anschließend in Deutsch abgeändert werden.

Tabelle 3.9: Übersicht der Pretester

Reihen-folge	Sprache: Beruf des Pretesters	Anzahl Pretests
1	Deutsch: Projektingenieur für Automatisierungstechnik EMSR	1
2	Deutsch: Bildverarbeitungs-Ingenieur	2
3	Deutsch: Softwareentwicklerin	1
4	Deutsch: PhD Student	1
5	Deutsch: PhD-Student / Software-Entwickler	2
6	Englisch: People and Culture Projects Analyst/United Kingdom	1
7	Englisch: PhD Student / Software-Entwickler	1

3.6 Onlinestellung und Einladung der Teilnehmer

Am 22.02.2017 wurde die Umfrage für die Öffentlichkeit bereit gestellt und zielgruppengerecht zur Teilnahme eingeladen. Für den ersten Tag wurden 40 Flyer mit QR-Code vorbereitet, um den beschriebenen Nachteil der analogen Adresseingabe klein zu halten. Diese wurden an Entwickler auf der Tagung Software Engineering in Hannover verteilt. 31 Entwickler und 21 Personen zum Weiterreichen der Umfrage wurden über das Karriereportal Xing und mittels E-Mails eingeladen. In 8 Fachforen, einer Mailingliste, zwei Mailverteilern, fünf Facebookfachgruppen und einer Twittergruppe wurde auf die Umfrage aufmerksam gemacht.

4 Ergebnisse

Die Ergebnisse sind lediglich für diese Studie mit eingeschränkter Zielgruppe verifizierbar. Eine Diskussion über die Übertragbarkeit auf alle Entwickler findet im Diskussionsteil im Kapitel 6 statt.

4.1 Rücklaufquote und Stichprobenstruktur

Die Umfrage endete am 20.03.2017 und war somit 27 Tage erreichbar. Die Umfrage verlief ruhig, näheres hierzu im Kapitel 6. An der Umfrage haben 318 Teilnehmer aus 19 Ländern teilgenommen. Hiervon sind 208 Teilnehmerantworten vollständig und 110 unvollständig. Unvollständig bedeutet, dass die letzte Seite nicht erreicht wurde und „absenden“ nicht gedrückt wurde.
Hierbei haben 79 Teilnehmer ausschließlich den Hauptteil beantwortet und 130 Teilnehmer den Hauptteil mit dem Zusatzteil. Die übrigen 109 Teilnehmer haben im Hauptteil den Fragebogen abgebrochen.
In der Abbildung 4.1 ist die Teamgröße der Umfrageteilnehmer zusammen mit der Auftretenshäufigkeit zu sehen. Die durchschnittliche Anzahl der Teammitglieder beträgt 13,4 Teilnehmer.

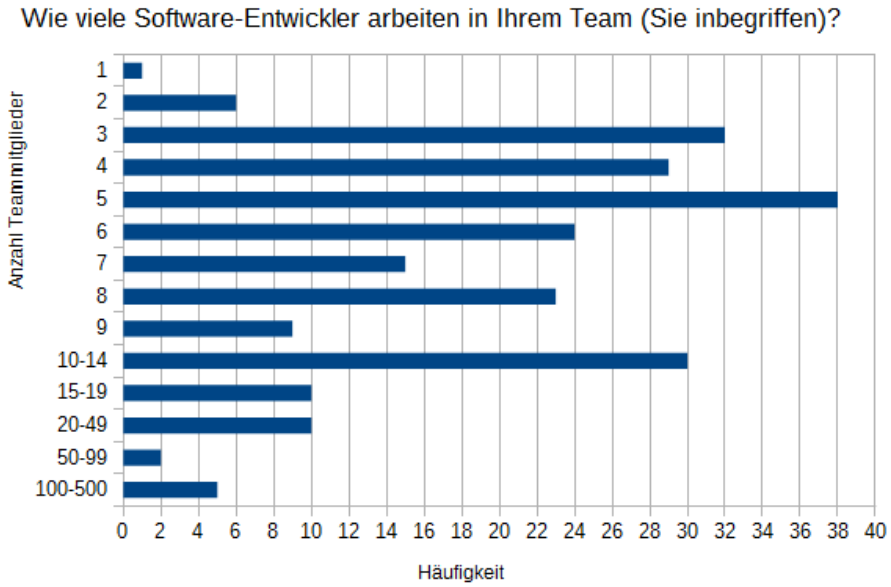

Abbildung 4.1: Teamgröße der Umfrageteilnehmer

In der Abbildung 4.2 ist die Rollenverteilung der einzelnen Teilnehmer zu sehen. Hierbei stellen die Entwickler mit 153 Teilnehmern die größte Gruppe, gefolgt von 50 Software-Architekten. Kleine Teilnehmergruppen bilden Projekt-Manager mit 10, Qualitätsbeauftragte mit 4, 1 Tester, sowie sonstige Rollen mit 8.

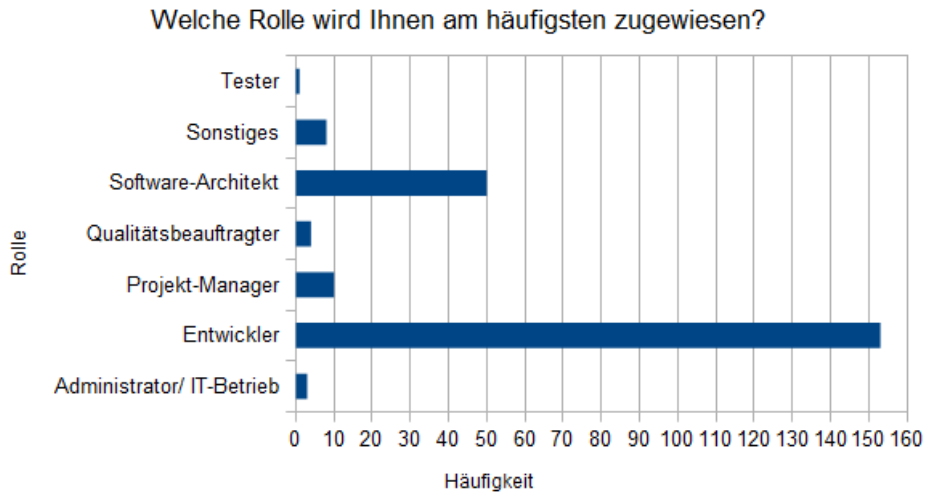

Abbildung 4.2: Rollenverteilung der Umfrageteilnehmer

4.2 RQ1: Wie verbreitet sind Code-Reviews?

Die erste Forschungsfrage lautet: „Wie verbreitet sind Code-Reviews?“. In dem Fragebogen lautet eine Frage hierzu „Verwenden oder verwendeten Sie Code-Reviews?“. 240 Teilnehmer antworteten auf diese Frage, hiervon sagten 186 Teilnehmer: „Es werden zur Zeit Code-Reviews verwendet.“. Dies stellt einen Anteil von 78 % dar. Die zweitgrößte Gruppe mit 30 sind Teilnehmer, die noch keine Überlegungen zu Code-Reviews unternommen haben. Es vermuten 15 von Ihnen, dass das Verhältnis aus Nutzen und aufzubringenden Anstrengungen von Code-Review lohnenswert ist. 4 Teilnehmer vermuten dies nicht. Von den 15 Teilnehmern, die vermuten, dass es hilfreich wäre Code-Reviews zu verwenden, gaben 8 an, dass keine Zeit für Code-Reviews ist.
Bei 15 Teilnehmern ist der Prozess eingeschlafen und wird somit nicht mehr verwendet. Es gibt 8 Teilnehmer, die sich bewusst gegen Code-Reviews entschieden haben und somit noch nie welche verwendet haben. Lediglich ein Teilnehmer sagte, dass in seinem Team Code-Reviews zwar verwendet wurden, jedoch ganz bewusste beendet wurden. Die Aufteilung der genannten Teilnehmergruppen sind noch einmal in der Abbildung 4.3 zu sehen.

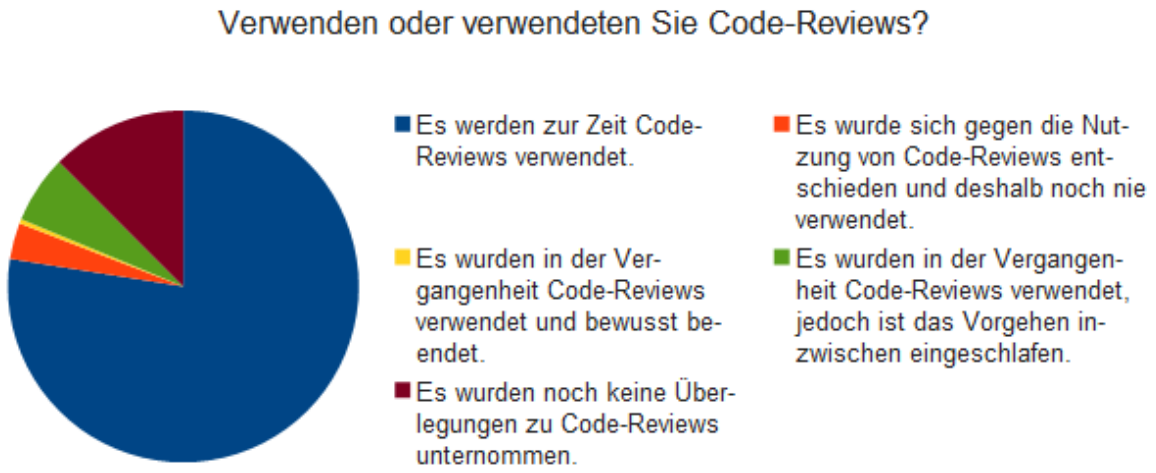

Abbildung 4.3: Verbreitung von Code-Reviews

Stellt man nun, wie in der Abbildung 4.4 zu sehen, die Reviewnutzung mit den verschiedenen Entwicklungstypen in einem Diagramm dar, fällt eine große Gruppe auf. Teams, die einen agilen Entwicklungsprozess mit Code-Reviews verwenden, stellen die größte Gruppe dar. Auch in der zweitgrößten Gruppe werden Code-Reviews verwendet, jedoch ist der Entwicklungsprozess klassisch. Ähnlich große Gruppen ergeben sich für Code-Review-Nutzung mit Ad Hoc Entwicklungsprozess, keine Überlegung zu Code-Reviews mit agilem Entwicklungsprozess und keine Überlegung zu Code-Reviews mit Ad Hoc Entwicklungsprozess.

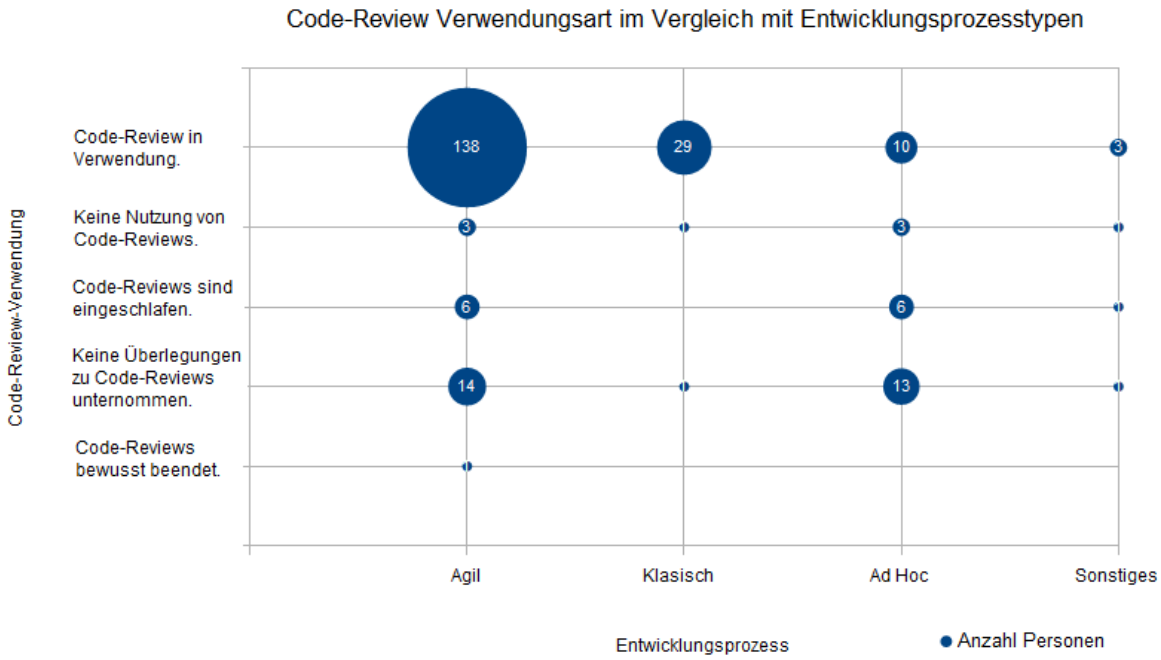

Abbildung 4.4: Reviewnutzung und Entwicklungstypen

4.3 RQ2: Wie ist der Einfluss der Kontextfaktoren auf die Review-Effekte?

Zu der Forschungsfrage 2 soll der Einfluss der Kontextfaktoren auf die Review-Effekte untersucht werden. Da jedoch zu 16 Kontextfaktoren und 10 Review-Effekten Daten vorliegen, können nicht alle Beziehungen untereinander untersucht werden.
Um die Aufgabe in Teilaufgaben zu zerlegen, bietet es sich an zu untersuchen, wie die Beziehungen einiger Kontextfaktoren auf die Effekte sind. In einer Erweiterungstabelle [59] zu einer wissenschaftlichen Ausarbeitung[60] finden sich bereits Effekte und beeinflussende Faktoren. Das Beeinflussungsniveau der Kontextfaktoren auf die Effekte ist unterschiedlich stark ausgeprägt. Es werden ausschließlich Kontextfaktoren mit den Effekten verglichen, bei denen von einer größeren Wichtigkeit der Beziehung auszugehen ist.
Die Einflüsse der Kontextfaktoren auf die sieben gewünschten Effekte und die drei unerwünschten Effekte werden nachfolgend untersucht. Die Variablen sind sowohl nominal als auch ordinal skaliert. Durch Lücken in der Datentabelle sind nicht immer Zellhäufigkeiten von mehr als 5 gegeben. Jedoch ist die Stichprobengröße größer als 50. Aus den gegebenen Voraussetzungen wird als Signifikanztest der exakte Fisher-Test gewählt . Bevor die Daten ausgewertet werden können, müssen sie begutachtet und an den Fisher-Test angepasst werden. Zuerst ist in Abbildung 4.5 zu sehen, welche Effekte die Teilnehmer als besonders wichtig eingestuft haben. Im linken Teil der Abbildung ist die Zuordnung der Prioritäten zu den farbigen Balken. Es ist sofort ersichtlich, dass als erste und zweite Priorität das „Verbessern von Code-Qualität“ und das „Finden von Defekten“ von vielen als die wichtigsten Ziele angesehen wurden.

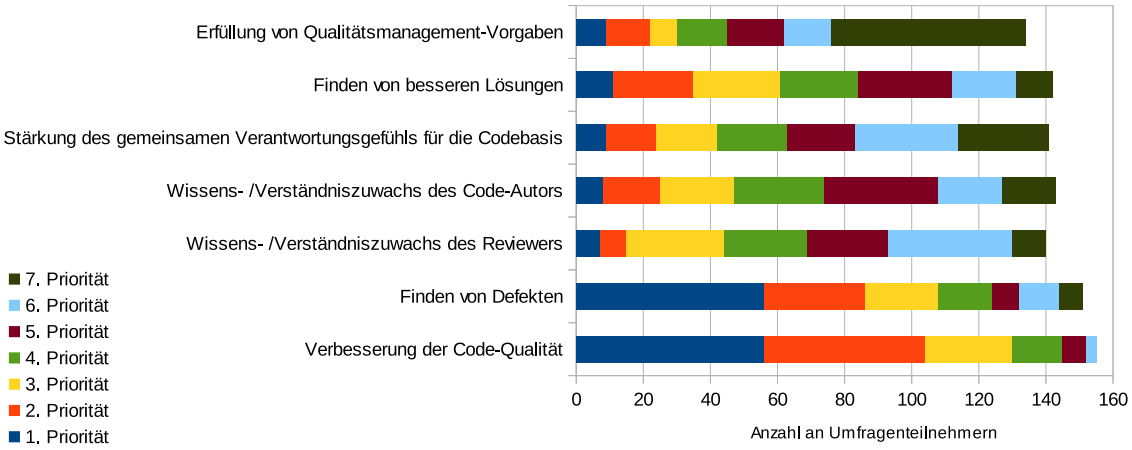

Abbildung 4.5: Priorisierung der positiven Review-Effekte

Bei den negativen Effekten in Abbildung 4.6 ist zu erkennen, dass als der am stärksten zu vermeidende Effekt die „Erhöhte Durchlaufzeit im Prozess genannt“ wird. Die abgebildeten positiven und negativen Effekte liegen in 10 Spalten vor. Jede Spalte steht dabei für einen Rang in der Ordnung. Die Ränge 1-7 der positiven Effekte und die Ränge 1-3 der negativen Effekte müssen noch für den exakten Fisher-Test zusammengeführt werden.

[59] http://tobias-baum.de/rp/detailedTable.pdf
[60] T. Baum u. a. 2016.

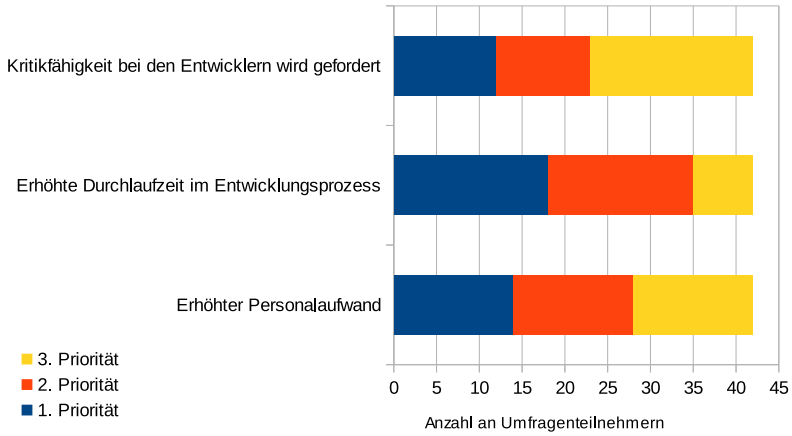

Abbildung 4.6: Priorisierung der negativen Review-Effekte

Bei den 7 Rängen der positiven Effekte wurden nur der erste Rang verwendet. Es erhält der jeweils betrachtete Effekt eine „0“, wenn er im Rang 1 auftaucht. Andere Effekte auf Rang 1 erhalten eine „1“. Dies geschieht für alle Effekte. Bei den negativen Effekten geschieht dies analog. Nach dem Anpassen der Daten liegt nun für alle positiven und negativen Effekte jeweils ein Vektor mit 2 Kategorien vor . Der Zustand „0“ besagt, dass der Teilnehmer diesem Effekt eine hohe Priorität zuordnet. Eine „1“ besagt, dass der Zustand eine niedrige Priorität erhält.

Anschließend werden in der bereits genannten Erweiterungstabelle Kontextfaktoren gesucht, die als Umfragefragen vorkommen. Auch hier müssen die Kategorien der Variablen für den exakten Fisher-Test zusammengefasst werden. Bei jeder Kontext-Variablen wird dabei die Häufigkeitsverteilung der Kategorien angeschaut, um eine ausgewogene Aufteilung vorzunehmen. Beispielsweise wurden wie bei der Schulnotenskala die 6 Kategorien in 1-3 und 4-6 zusammengeführt, um die Aussage der Antworten nicht zu verfremden. Jede Anpassung hat als Ziel, aus mehreren Antworten zu einer Frage einen Vektor mit zwei Zuständen zu erzeugen. Nachfolgend wird der Zusammenhang zwischen Kontextfaktoren und Effekten untersucht, bei denen ausreichend brauchbare Daten zur Verfügung stehen.

In der linken Spalten der Tabellen werden Kontextfaktoren aufgelistet, die im Zusammenhang zum jeweiligen Effekt stehen stehen. Hierbei gibt es zu jedem Faktor die Auskunft, ob bei den Teilnehmern die Ausprägung zutrifft „ ✓ “ oder nicht „ ✗ “.
In den mittleren Spalten findet sich die Anzahl der gezählten Einträge für die Effekte und Kontextfaktoren. Die Anzahl ist nicht mit der Teilnehmeranzahl gleichzusetzen, da durch die Betrachtung der einzelnen Effekte nur der Teil der Befragten betrachtet wird, bei dem der Effekte auf Rang 1 steht. Hierbei gibt es gibt es die Ausprägungen, dass Effekt als wichtig oder unwichtig ansehen. In der rechten Spalte sind für den Unabhängigkeitstest die Signifikanzwerte zwischen den Effekten und den Kontextfaktoren aufgelistet. Als Signifikanzniveau wird $\alpha = 0.05$ angesetzt. Im nachfolgenden wird nur die 2-seitige Signifikanz beurteilt.

Der Reihe nach werden die statistischen Ergebnisse bewertet. Als erstes wird die Ta-

belle 4.10 betrachtet. Bei dem Kontextfaktor „häufige Notwendigkeit, alten Code zu ändern“ und dem „Effekt Verbesserung der Code-Qualität“ gibt es 59 Einträge bei denen der Kontextfaktor nicht zutrifft und die den Effekt als wichtig betrachten. Bei 25 Einträgen wird der Effekt als nicht wichtig betrachten. Es ergibt sich ein p-Wert von 0,038. Somit besteht zwischen den beiden ein Zusammenhang.
Bei Teams die langfristiges Denken gibt es 41 Einträge, die den Effekt der Code-Qualitätsverbesserung als nicht wichtig betrachten. Dem stehen 61 Einträge, die den Effekt als wichtig betrachten gegenüber. Der p-Wert beträgt 0,107. Der Test ist nicht signifikant, jedoch ist eine Tendenz erkennbar, dass Teams die langfristig denken den Effekt als wichtig beurteilen.
Bei der Untersuchung eines Zusammenhangs zwischen „unerfahrene / heterogene Entwickler“ ergeben sich 0 Eintragungen und bei den Teilnehmern die den Effekt als unwichtig betrachten. Bei denen die unerfahrene/heterogene Entwicklern die den Effekt als wichtig ansehen, ergeben sich 5 Einträge. Der p-Wert liegt über dem Signifikanzniveau. Als Tendenz ist zu erkennen, dass unerfahrene Teams die Verbesserung der Code-Qualität als wichtig erachten.
Für Teams mit großer/komplexer Codebase die den Effekt als unwichtig ansehen ergeben sich 15 Einträge. Für Teams die den Effekt wichtig finden ergeben sich 18 Einträge. Mit einem p-Wert von 0,229 ist die Signifikanz nicht gegeben.

Tabelle 4.10: Kontextfaktoren tragen zur Verbesserung der Code-Qualität bei

Gewünschter Effekt: Verbesserung der Code-Qualität			
Zielniveau des Effekts wird beeinflusst durch..	Effekt als nicht wichtig angesehen	Effekt als wichtig angesehen	Exakte Signifikanz
..häufige Notwendigkeit, alten Code zu ändern ✓	23	25	2-seitig:0,038
..häufige Notwendigkeit, alten Code zu ändern ✗	24	59	1-seitig:0,023
..langfristiges Denken ✓	41	61	2-seitig: 0,107
..langfristiges Denken ✗	13	36	1-seitig: 0,071
..unerfahrene / heterogene Entwickler ✓	0	5	2-seitig: 0,143
..unerfahrene / heterogene Entwickler ✗	20	27	1-seitig: 0,077
..komplexe/große Codebase ✓	15	18	2-seitig: 0,229
..komplexe/große Codebase ✗	12	27	1-seitig: 0,150

Zu Team bei denen ernste Folgen von Defekten haben und die das Finden von Defekten als nicht wichtig betrachten, gibt es 29 Einträge. Folglich gibt es zu 71 Teams Einträge, die den Effekt als wichtig betrachten. Der Test erweist sich als signifikant, womit ein Zusammenhang gegeben ist. Bei unerfahrenen Entwicklern und dem Effekt „Finden von Defekten“ ergab sich statistisch kein Zusammenhang.

Tabelle 4.11: Review-Effekt „ Finden von Defekten“

Gewünschter Effekt: Finden von Defekten			
Zielniveau des Effekts wird beeinflusst durch..	Effekt als nicht wichtig angesehen	Effekt als wichtig angesehen	Exakte Signifikanz
..ernste Folgen von Defekten ✓	29	71	2-seitig: 0,031
..ernste Folgen von Defekten ✗	25	27	1-seitig: 0,016
..unerfahrene Entwickler ✓	8	19	2-seitig: 0,327
..unerfahrene Entwickler ✗	4	22	1-seitig: 0,182

Bei dem Effekt zum Lernen des Reviewers und den Kontextfaktoren konnte keine Signifikanz festgestellt werden. Somit ergibt sich kein Zusammenhang der statistisch bewiesen ist. Auffällig ist jedoch, dass es 125 Einträge zu Teams mit Collective-Code-Ownership gibt, die das Lernen des Reviewers als wichtig betrachten.

Tabelle 4.12: Review-Effekt „Das Lernen des Reviewers“

Gewünschter Effekt: Das Lernen des Reviewers			
Zielniveau des Effekts wird beeinflusst durch..	Effekt als nicht wichtig angesehen	Effekt als wichtig angesehen	Exakte Signifikanz
..große Unterschiede im Wissen ✓	3	24	2-seitig: 0,318
..große Unterschiede im Wissen ✗	1	32	1-seitig: 0,234
..der Collective-Code-Ownership ✓	6	125	2-seitig: 1
..der Collective-Code-Ownership ✗	1	22	1-seitig: 0,719
..komplexen/großen Codebase ✓	3	42	2-seitig: 0,287
..komplexen/großen Codebase ✗	0	27	1-seitig: 0,238

In der Tabelle 4.13 sind die Häufigkeiten der Einträge zum „Lernen des Autors“ den Kontextfaktoren zu sehen. Bei Einträgen zu der komplexen/großen Codebase und dem Effekt kann von einem Zusammenhang gesprochen werden. Der p-Wert beträgt 0,013 und liegt unter dem Signifikanzniveau. Die Einträge zu großen Wissensunterschieden im Team und der Collective-Code-Ownership liefern keinen Zusammenhang.

Tabelle 4.13: Review-Effekt „Das Lernen des Autors“

Gewünschter Effekt: Das Lernen des Autors			
Zielniveau des Effekts wird beeinflusst durch..	Effekt als nicht wichtig angesehen	Effekt als wichtig angesehen	Exakte Signifikanz
..große Unterschiede im Wissen ✓	1	26	2-seitig: 1,000
..große Unterschiede im Wissen ✗	2	31	1-seitig: 0,576
..der Collective-Code-Ownership ✓	7	124	2-seitig: 1
..der Collective-Code-Ownership ✗	2	31	1-seitig: 0,659
..komplexen/großen Codebase ✓	0	33	2-seitig: 0,013
..komplexen/großen Codebase ✗	7	32	1-seitig: 0,010

4.4 RQ3: Warum beenden Teams die Nutzung von Code-Reviews?

Die dritte Forschungsfrage dreht sich um Teams, die den Code-Review-Prozess einstellen und lautet „Warum beenden Teams die Nutzung von Code-Reviews?“. Es liegen insgesamt von 318 Teilnehmern Antworten zu deren Code-Review-Nutzung vor. Bei 15 Teilnehmern ist der Review-Prozess eingeschlafen und wird nicht mehr verwendet. Zu dem Fall, dass Code-Reviews verwendet wurden und anschließend ganz bewusst eingestellt wurden, liegt nur eine Antwort vor.

In der Abbildung 4.7 ist vertikal die Mitarbeiteranzahl der Organisationen und horizontal die Anzahl der beobachteten Fälle in Prozent, in denen der Review-Prozess eingeschlafen ist, zu sehen. Hierbei ist zu beobachten, dass bei den meisten kleineren Organisationen der Prozess einschläft. Bei großen Organisationen mit „10001 oder mehr“ steigt die Anzahl der beobachteten Fälle wieder an. Möglich ist, dass kleine Firmen nicht die Kapazitäten haben, den Entwicklungsprozess hinreichend zu überwachen. In dem einen Fall mit der großen Mitarbeiterzahl wurde angegeben, dass mangelnde Zeit und Kapazität der Grund für den Abbruch sind. [61] Im zweiten Fall wurde ein fehlender „Return on Investment“ genannt. [62]

[61] Umfrageteilnehmer : I. 121
[62] I. 281

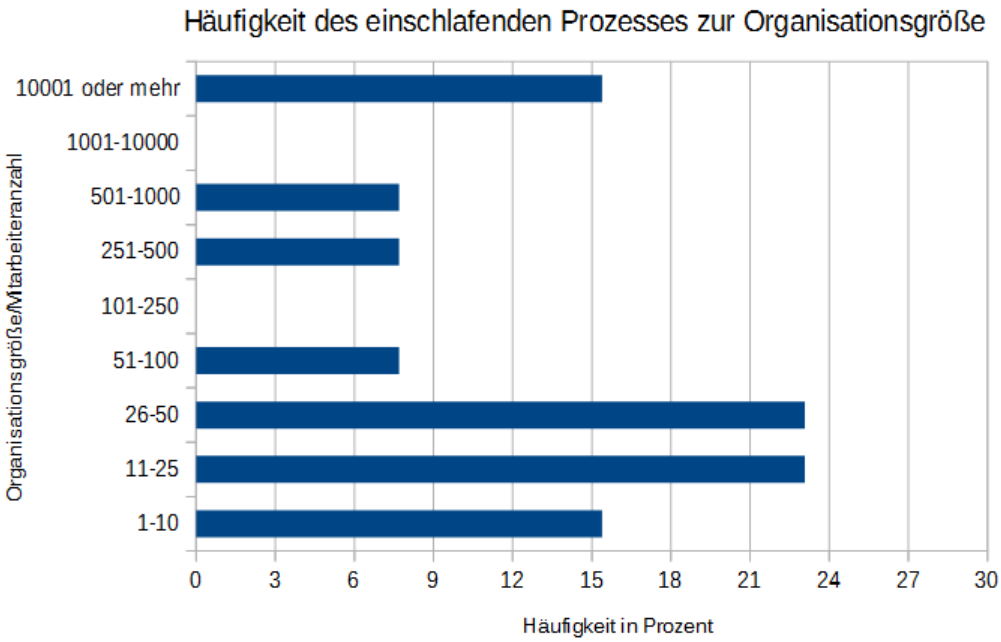

Abbildung 4.7: Häufigkeit einschlafender Prozesse zur Organisationsgröße

Es wurde in einer Ausarbeitung eine Arbeitshypothese H_1 zu eingeschlafenen Review-Prozessen aufgestellt. Die Hypothese lautet: „Code-Reviews bleiben sehr wahrscheinlich in Verwendung, wenn es in den Prozess eingebettet ist (sowie unterstützende Werkzeuge), so dass es keiner bewussten Entscheidung bedarf, eine Überprüfung durchzuführen.“[63]. Die Nullhypothese H_0 wird wie folgt definiert: „Code-Reviews bleiben nicht in Verwendung, wenn sie in den Prozess eingebettet sind(sowie unterstützende Werkzeuge), so dass es eine bewusste Entscheidung erfordert, eine Überprüfung durchzuführen.“. Im nachfolgenden wird diese Hypothese H_0 durch zwei gestellte Fragen aus dem Fragebogen näher untersucht. Teilnehmer, die zur Zeit Reviews verwenden, erhielten folgende Frage: „Wie entscheidet sich in den meisten Fällen, ob ein Code-Review angesetzt werden soll?“. Teilnehmer, die in der Vergangenheit Reviews verwendeten, bekamen die Frage in der Vergangenheitsform gestellt. In der Tabelle 4.14 ist die Häufigkeit der Antworten zum Auslösen von Reviews zu sehen. Bereits hier zeigt sich, dass 104 Teilnehmer zur Zeit Reviews verwenden und deren Prozess eingebettet ist und durch Regeln oder Konventionen ausgelöst wird. Dies stellt eine Mehrheit dar. Bei den Teilnehmern, die Reviews in der Vergangenheit verwendeten, gab die Mehrheit an, dass es keine festen Regeln gab. Beide Gruppen passen zu der aufgestellten Hypothese.

[63] Tobias Baum u. a. 2016.

Tabelle 4.14: Gründe zur Auslösung eines Reviews

		Code-Reviews [..]		
		.. werden verwendet	.. wurden verwendet	Gesamt
Wie entscheidet/entschied sich in den meisten Fällen ob ein Code-Review angesetzt werden soll/wurde?	Es gibt keine festen Regeln/Konventionen, ob ein Review stattfinden soll.	59	10	69
	Es gibt feste Regeln/Konventionen, ob ein Review stattfinden soll.	104	3	107
Gesamt		163	13	176

Bei den Variablen handelt es sich um kategoriale Variablen. Zum Überprüfen der Hypothesen eignet sich ein Chi-Quadrat-Test. In der Stichprobe gibt es Kategorien mit weniger als 20 Fällen. Somit dürfen nur die Ergebnisse des exakten Fisher-Tests verwendet werden. In der Tabelle 4.15 ist der exakte Fisher-Test rot umrandet. Die exakte Signifikanz beträgt $p = 0,005$. Das Signifikanzniveau α wird mit 5% festgelegt. Womit der p-Wert an der Grenze zum Signifikanzwert von „0,05“ ist. Ob das Ergebnis als signifikant eingestuft werden kann, ist nicht eindeutig. Für den Fall, dass es als signifikant betrachtet wird gilt, dass Reviewnutzer und ehemalige Reviewnutzer unterschiedlich häufig Regeln oder Konventionen zum Auslösen von Reviews verwenden. Die Nullhypothese H_0 wird demnach abgelehnt und H_1 als bewiesen erachtet.

Tabelle 4.15: Chi-Quadrat-Tests

Chi-Quadrat-Tests

	Wert	df	Asymptotische Signifikanz (zweiseitig)	Exakte Signifikanz (2-seitig)	Exakte Signifikanz (1-seitig)
Chi-Quadrat nach Pearson	8,379[a]	1	,004	,006	,005
Kontinuitätskorrektur[b]	6,757	1	,009		
Likelihood-Quotient	8,293	1	,004	,006	,005
Exakter Test nach Fisher				,006	,005
Zusammenhang linear-mit-linear	8,331[c]	1	,004	,006	,005
Anzahl der gültigen Fälle	176				

Als nächstes wird geprüft, wie der Zusammenhang zwischen eingeschlafenen Prozessen und dem Change-based-Review ist. Die bereits genannte Hypothese H1 besagt: „ ..Code-Reviews bleiben sehr wahrscheinlich in Verwendung, wenn es in den Prozess eingebettet ist..“[64].

[64] Tobias Baum u. a. 2016.

Der change-based-Review Ansatz besagt: „...Jedes Mal, wenn eine „Arbeitseinheit“ als „bereit für das Review“ angesehen wird, werden alle Änderungen, die im Laufe ihrer Implementierung durchgeführt wurden, als Reviewkandidaten betrachtet..“[65], d.h. es erfolgen Reviews auf Basis von Änderungen, nachdem an einer Arbeitseinheit Änderungen vorgenommen wurden. Somit erfolgt die Änderung nicht durch einen Mitarbeiter im Entwicklungsprozess, sondern auf Basis von Regeln oder Konventionen.
Somit ist zu erwarten, dass bei einem Entwicklungsprozess, bei dem der Review-Prozess eingebettet ist und die Reviews durch Änderungen ausgelöst werden, dieser seltener einschläft. Es erfolgten Fragen, ob der Review-Prozess durch Code-Änderungen ausgelöst wird. Auch wurden Fragen dazu gestellt, ob der Review-Prozess durch einen Vertrag oder ein Gesetz geregelt war. 13 Teilnehmer, bei denen der Review-Prozess eingeschlafen ist, antworteten, dass der Review-Prozess nicht auf Basis von vertraglichen oder gesetzlichen Grundlagen geregelt war. Regelungen können helfen, dass unbeabsichtigte Beenden zu verhindern. Es wurde konkret zu den Gründen des eingeschlafenen Prozesses gefragt. Es ließen sich vier Gruppen für Gründe ausmachen: Teams, die Reviews als nicht lohnenswert erachten und lieber Alternativen verwenden, Teams, die aus Zeitmangel oder fehlender Ressourcen Reviews nicht mehr durchführen, Code-Review war nicht erzwungen sowie das Vorhandensein einer höheren Personalebene, die sich wenig um Code-Reviews kümmert.
Die Übersicht der Gründe für das Beenden ist in der Abbildung 4.8 zu sehen. Als erstes wird die Vermutung bestätigt, dass ein nicht eingebettetes Review zum Abbruch führt. Dies trifft zu, wenn Code-Reviews nicht erzwungen werden. Bei zwei Teilnehmern schenkt die höhere Personalebene dem Reviewprozess zu wenig Aufmerksamkeit. Drei Teilnehmer sagten, dass ein Mangel an Ressourcen vorlag. Ob der Prozess eingebettet war, ist nicht bekannt. Dass Reviews nicht lohnenswert sind, sagte ein Teilnehmer.

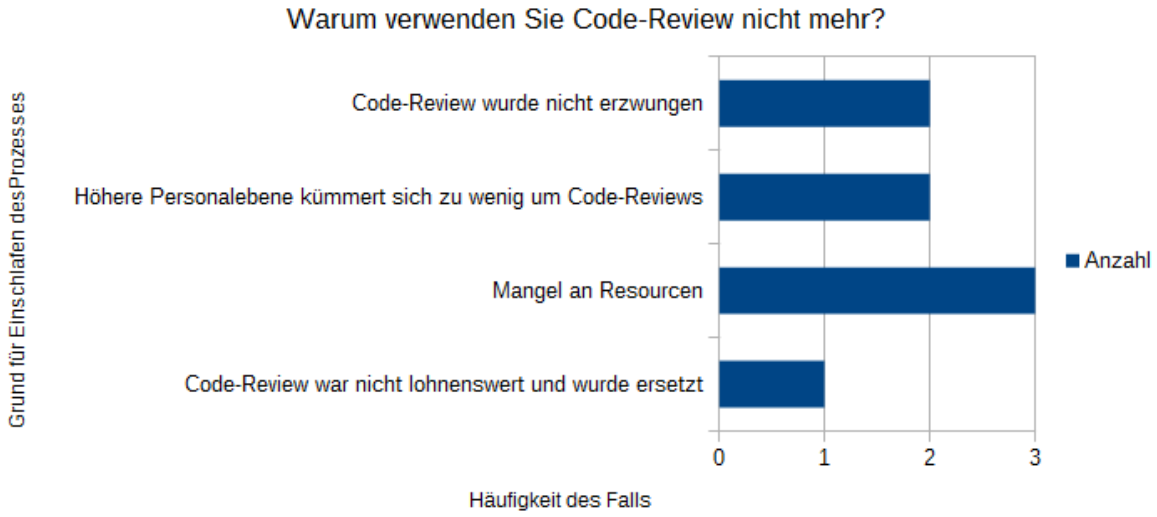

Abbildung 4.8: Gründe für das Beenden von Code-Reviews

[65] T. Baum u. a. 2016.

Es wurden auch persönliche Erfahrungen zu negativen Auswirkungen von Code-Reviews genannt, die im Zusammenhang mit einem einschlafenden Prozess stehen. Hier spielen auch der Nutzen, die aufzubringende Zeit und die soziale Herausforderungen eine Rolle. Von einem Teilnehmer, bei dem die Autoren Reviews ansetzen, hieß es: „Manchmal musste ich dem Reviewer erstmal beibringen, wie [...] funktioniert. Das kostet viel Zeit und bringt (mir) wenig.“ [66]. Dies verstärkt die Gefahr des einschlafenden Prozesses, ebenso wie „Sozialen Herausforderungen“ [67] Ein Umfragenteilnehmer gab an, in der Vergangenheit Reviews verwendet zu haben und dies bewusst beendet zu haben. Hierzu heißt es, dass Code-Review nicht mehr verwendet wird und stattdessen Paar-Programmierung als Alternative erfolgt. Zu den Teilnehmern, die bewusste Code-Reviews beenden, ist auf Grund der geringen Anzahl keine Schlussfolgerung möglich.

[66] I. 158
[67] I. 187

5 Verwandte Ausarbeitungen

Eine verwandte Arbeit aus dem Jahre 2016 führte eine Umfrage zu Code-Reviews mit „Open-Source-Software(OSS)"-Entwicklern und Entwicklern von Microsoft durch.[68] Hierbei wurde jedoch eine andere Definition zu Code-Reviews aufgestellt. Es wird von „zeitnahen Code-Reviews" gesprochen mit folgenden Eigenschaften: „..Zeitnahe Codeüberprüfung ist definiert als leichtgewichtig, mehr informell, asynchron und unterstützt von spezialisierten Werkzeugen..". Die Ergebnisse dieser Analyse ergaben, dass die Entwickler 10-15 % ihrer Zeit mit Code-Reviews verbringen und die Codeüberprüfung als wichtig betrachten. Andere Effekte wie der Wissensaustausch, der Gemeinschaftsaufbau und die Beibehaltung der Codequalität wurden ebenso als wichtig durch die Entwickler erachtet. Der vorliegende Code bei einem Review von den Teamkollegen erschafft Eindrücke, die spätere Kooperationen beeinflussen können. Es wurden viele Gemeinsamkeiten zwischen den Teilnehmern von Microsoft sowie den „OSS-Entwicklern" gefunden. Als eine interessante Erkenntnis wurde erachtet, dass die „OSS-Befragten" die Code-Überprüfung als eine wichtige Methode sahen. Wohingegen die Microsoft-Befragten die Wissensverbreitung als wichtig befanden.[69]

Zu „Softwaretests in Praxis und Forschung" wurde im Jahr 2016 durch eine Kooperation der Hochschulen Bremerhaven und Bremen, der Technischen Hochschule Köln, des German Testing Board und des Swiss Testing Board eine Befragung durchgeführt.[70] Der Fokus lag dabei auf der „Qualitätssicherung aus Sicht des Managements", der „operativen Umsetzung" und der „Forschung". Die Forscher sehen noch großen Bedarf bei der Auswahl, der Wirksamkeit und der gegenseitigen Abhängigkeit einzelner Maßnahmen zur Qualitätssicherung. Auch der Bedarf an Erkenntnissen zu den agilen Praktiken und Vorgehensweisen besteht. Als große Herausforderungen für die kommenden Jahre werden die Themen „Security und Safety", „Künstliche Intelligenz" sowie „Internet der Dinge/Industrie 4.0" angesehen.[71]

Die zuerst genannte Arbeit untersucht zwar ebenfalls mit einer Umfrage Code-Reviews. Jedoch hat sie einen anderen Fokus. Die vorliegende Arbeit hingegen versucht nicht nur die Forschungsfragen zu beantworten, sondern auch den Review-Prozess, die Review-Effekte und die Kontextfaktoren in ihrer Fülle auf Basis von wissenschaftlichen Arbeiten zu erforschen.

[68] Bosu, Carver u. a. 2016.
[69] Ebd.
[70] Vosseberg, Spillner und Winter 2016.
[71] Ebd.

6 Diskussion

Die Umfrage verlief ohne große Komplikationen seitens der Teilnehmer ab. Lediglich einmal gab es Kritik, dass Fragen zu Code-Reviews gestellt wurden, obwohl der Teilnehmer keine Reviews verwendet. [72] Da es sich um den einzigen Fall handelt, ist von einer Falschauswahl, bei der Filterfrage auszugehen. Es gab drei Teilnehmer, die es als schwierig empfanden, mit dem Mobiltelefon an der Umfrage teilzunehmen. [73], [74], [75] Von vier Teilnehmern wurde bemängelt, dass Antwortkategorien fehlten, womit das Antworten Probleme machte. [76], [77], [78], [79] Es empfanden fünf Teilnehmer manche Fragen unklar formuliert oder die Definitionen als verbesserungswürdig. [80], [81], [82], [83]

In dem Abschnitt 3.1 wird genannt, dass ein großer Teil der Firmen keine Reviews verwendet. Dies konnte nicht bestätigt werden. Die Studie ist aus dem Jahr 2002, was die Zunahme der Code-Review-Verwendung vermuten lässt. Es lag auch eine andere Zielgruppe vor. Es wird vermutet, dass es eine Verunreinigung bei der Erfassung der Zielgruppe der vorliegenden Arbeit gibt. Es sollten auch Personen teilnehmen, die selber nicht Code-Reviews verwenden. Jedoch ist unklar, wie konsequent sich die Personenfilterung umsetzen lässt, weil das Umfragethema „Code-Reviews“ Personen, die Reviews nicht verwenden, möglicherweise abschreckt.

Die Ergebnisse lassen sich nur für diese Studie anwenden, da versucht wurde, die Zielgruppe auf Personen, die in einem Team arbeiten, das kommerziell Software entwickelt, zu beschränken. Auf Grund der starken Einschränkung der Zielgruppe konnte keine vergleichbare Umfrage gefunden werden, um Aussagen über die Grundgesamtheit zu treffen. Die Studie muss als Momenterfassung angesehen werden, da davon auszugehen ist, dass die Ergebnisse im Zusammenhang mit dem Durchführungszeitraum stehen und somit eine Momentaufnahme bilden. Obwohl ausgiebig Überlegungen zu der Frageformulierung unternommen wurden, ist es möglich, dass manche Fragen den Befragten zu einer bestimmten Antwort verleitet haben. Bei der statischen Validierung wurden Kategoriegruppen nach Ermessen des Autors zusammen gefasst. Eine andere Zusammenfassung führt zu anderen Ergebnissen. Ob die Hypothesen falsch aufgestellt sind oder ob für manche untersuchten Zusammenhänge die Stichprobe zu klein ist lässt sich an dieser Stelle nicht sagen. Es ist jedoch auffällig, dass bei Vergleichen zwischen Kontextfaktoren und Review-Effekten mit hohem Signifikanzwert nur eine kleine Anzahl an Antworten vorliegt.

[72] Umfrageteilnehmer: I. 279
[73] I. 58
[74] I. 114
[75] I. 194
[76] I. 118
[77] I. 121
[78] I. 206
[79] I. 116
[80] I. 224
[81] I. 205
[82] I. 279
[83] I. 325

Die Zielgruppe der Studie sind Software-Entwicklerteams im kommerziellen Sektor. Im klassischen Ingenieursbereich werden ebenso Reviews zum Verbessern des Produkts verwendet. In den Produkten der Elektroindustrie findet sich immer mehr Software, um intelligente Produkte zu produzieren. Deshalb finden hier ebenfalls Code-Reviews Anwendung. Darüber hinaus werden in der Industrie auch Reviews von physischen Produkten wie Schaltungen vorgenommen. Viele Methoden stammen aus der Autoindustrie. So begleitet die von Toyota stammende Entwicklungsmethode „Design Review Based on Failure Mode (DRBFM)“ den Entwicklungsprozess eines Prozesses oder Produktes.[84] In einem Beitrag zu Design- und Prozessreviews stellt ein Ingenieur die Einführung von Produktdesign- und Fertigungsprozessüberprüfungen in einer Firma vor. In Ergebnissen wird ausführlich erläutert, wie Zuverlässigkeitsprobleme vermieden werden konnten, Kosten gesenkt und eine erfolgreiche Produkt- und Prozess-Einführung erfolgte.[85]

[84] Brunner 2014.
[85] Araujo 2017.

7 Abschließende Betrachtung

7.1 Zusammenfassung

Die Umfrage kann als Erfolg betrachtet werden. Es beteiligten sich 318 Personen aus 19 Ländern. Hiervon sind 208 Teilnehmerantworten vollständig und 110 unvollständig, womit sich Forschungsfragen beantworten ließen. Bei der Verbreitung von Code-Reviews ist zu beobachten, dass von 240 Teilnehmern 186 Teilnehmer zur Zeit der Beantwortung Code-Reviews nutzen. Somit ist davon auszugehen, dass die Bedeutung erkannt wurde und sich in den Teams Erfolge durch Code-Reviews eingestellt haben. Jedoch gibt es auch mit 30 Teilnehmern eine große Gruppe, die noch keine Gedanken zu Code-Reviews unternommen hat. Von den Teams, die sich noch keine Gedanken zu Code-Reviews gemacht haben, vermuten 15, dass das Verhältnis aus Nutzen und aufzubringenden Anstrengungen von Code-Review lohnenswert ist. Jedoch gaben sie an, dass Code-Reviews auf Grund von Zeitmangel nicht erfolgen.
Bei 15 Teilnehmern ist der Reviewprozess eingeschlafen und wird nicht mehr verwendet. 8 Teilnehmer haben sich bewusst gegen Code-Reviews eingestellt und noch nie verwendet. Ein Teilnehmer verwendete Code-Reviews und beendete diese bewusst.

Bei der Überprüfung der Einflüsse der Kontextfaktoren auf die Review-Effekte konnte einige Zusammenhänge durch den exakten Test nach Fisher bestätigt werden. Die statistische Auswertung ergab, dass häufige Reviews und das Verbessern der Code-Qualität im engen Zusammenhang stehen. Ebenso ist für den Kontextfaktor, dass Folgen ernste Konsequenzen haben und dem Finden von Defekten ein Zusammenhang zu sehen. Eine große und komplexe Codebase bildet mit dem Effekt des Lernens des Autors ebenso einen Zusammenhang. Viele weitere Ergebnisse können nicht als statistisch signifikant eingestuft werden, jedoch ist eine Tendenz erkennbar.

Es lässt sich festhalten, dass von 240 Software-Entwicklerteams 15 Code-Reviewprozesse eingeschlafen sind. Lediglich ein Entwicklerteam entschied sich für ein bewusstes Beenden von Code-Reviews. Somit kann gesagt werden, dass es dem Großteil der Entwicklerteams gelingt, einen eingeführten Code-Reviewprozess am Laufen zu halten.
Häufig sind es kleine Organisationen, bei denen der Review-Prozess einschläft. Die Hypothese „Code-Reviews bleiben sehr wahrscheinlich in Verwendung, wenn sie in den Prozess eingebettet sind (sowie unterstützende Werkzeuge), so dass es keine bewusste Entscheidung bedarf, eine Überprüfung durchzuführen.“ kann nicht vollständig bestätigt werden, da der p-Wert von 0,05 bei gesetztem Signifikanzniveau von 5% sich an der Grenze befindet. Allerdings heißt es in der Hypothese auch „sehr wahrscheinlich“, was auch einen Spielraum in der Bewertung lässt. Jedoch lässt sich beobachten, dass Fälle bei denen Code-Reviews in Verwendung sind, häufig mit festen Regeln und Konventionen einhergehen. Auf der anderen Seite ist auffällig, dass Teams mit abgebrochenem Code-Review-Prozess keine festen Regeln und Konventionen verwendeten. Es ist davon auszugehen, dass gesetzliche und vertragliche Regelungen helfen, den Prozess aufrecht zu halten.

7.2 Ausblick

Es ließ sich nicht für alle untersuchten Problemstellungen die Signifikanz der Ergebnisse nachweisen. Möglicherweise ist die Stichprobengröße für einzelne Aspekte zu gering, da manche Teilnehmer zu bestimmen Punkten keine Auskunft gaben. Es bedarf weiterer Untersuchungen. Denkbar wäre eine kompakte Umfrage mit wenigen Pflichtfragen. Dies würde die Schwierigkeiten in der Verwertung umgehen.
Durch die große Anzahl an Fragen konnte das Anwendungsverhalten von kommerziellen Entwicklerteams ausgiebig erfasst werden. Es steht somit eine großer Datensatz zum Code-Review-Prozess zur Verfügung, um weitere Untersuchungen durchzuführen. Die Rohdaten können dabei direkt zur Auswertung genutzt werden. Es können bei der Datenanalyse möglicherweise weitere Forschungsfragen entstehen, die ebenso für empirische Forschungen nützlich sind.

8 Literaturverzeichnis

Literatur

Araujo, E. B. (2017). „Using design and process review based on failure modes to improve development process“. In: *2017 Annual Reliability and Maintainability Symposium (RAMS)*, S. 1–7. DOI: 10.1109/RAM.2017.7889725 (siehe S. 54).

Bauer, H.H., D. Große-Leege und J. Rösger (2012). *Interactive Marketing im Web 2.0+: Konzepte und Anwendungen für ein erfolgreiches Marketingmanagement im Internet*. Vahlen, S. 307. URL: https://books.google.de/books?id=ImUW0ZE4GM8C (siehe S. 21).

Baum, T., O. Liskin, K. Niklas und K. Schneider (2016). „A Faceted Classification Scheme for Change-Based Industrial Code Review Processes“. In: *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, S. 74–85. DOI: 10.1109/QRS.2016.19 (siehe S. 2, 4 f., 13, 43, 50).

Baum, Tobias, Olga Liskin, Kai Niklas und Kurt Schneider (2016). „Factors Influencing Code Review Processes in Industry“. In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE 2016. New York, NY, USA: ACM, S. 85–96. DOI: 10.1145/2950290.2950323. URL: http://doi.acm.org/10.1145/2950290.2950323 (siehe S. 2, 11, 16, 23 ff., 48 f.).

Baur, Nina und Jörg Blasius (2014a). *Handbuch Methoden der empirischen Sozialforschung*. Springer, S. 103 (siehe S. 17).

— (2014b). *Handbuch Methoden der empirischen Sozialforschung*. Springer, S. 136 (siehe S. 17).

Baysal, Olga, Oleksii Kononenko, Reid Holmes und Michael W Godfrey (2013). „The influence of non-technical factors on code review“. In: *Reverse Engineering (WCRE), 2013 20th Working Conference on*. IEEE, S. 122–131 (siehe S. 23).

Bosu, Amiangshu und Jeffrey C Carver (2013). „Impact of peer code review on peer impression formation: A survey“. In: *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*. IEEE, S. 133–142 (siehe S. 23).

Bosu, Amiangshu, Jeffrey C Carver, Christian Bird, Jonathan Orbeck und Chris Chockley (2016). „Process Aspects and Social Dynamics of Contemporary Code Review: Insights from Open Source Development and Industrial Practice at Microsoft“. In: *IEEE Transactions on Software Engineering* (siehe S. 23 f., 52).

Brunner, Franz J (2014). *Japanische Erfolgskonzepte: KAIZEN, KVP, Lean Production Management, Total Productive Maintenance Shopfloor Management, Toyota Production Management, GD3-Lean Development*. Carl Hanser Verlag GmbH Co KG (siehe S. 54).

Developer Survey Results 2016. `http://stackoverflow.com/insights/survey/2016`. Accessed: 2017-03-07 (siehe S. 26).

Diekmann, Andreas (2008). *Empirische Sozialforschung. Grundlagen, Methoden, Anwendungen. 19. Aufl., Orig* (siehe S. 36).

Fagan, M. (1976). „Design and code inspections to reduce errors in program development“. In: *IBM Systems Journal vol. 15, no. 3*, S. 182–211 (siehe S. 3).

Fisher, Ronald A (1922). „On the interpretation of χ 2 from contingency tables, and the calculation of P“. In: *Journal of the Royal Statistical Society* 85.1, S. 87–94.

Gilb, T. und D. Graham (1993). „Software Inspection“. In: Addison-Wesley (siehe S. 3).

Höpflinger, François (2003). „Befragung. Wichtige Regeln der Fragebogen-Konstruktion“. In: *Im Netz: http://www. hoepflinger. com/fhtop/fhmethod1. html [14.08. 2009]* (siehe S. 36).

IEEE (2008). „Standard for Software Reviews and Audits, IEEE Std. 1028-2008.“ In: (Siehe S. 3).

Jacob, Rüdiger, Andreas Heinz und Jean Philippe Décieux (2013). *Umfrage: Einführung in die Methoden der Umfrageforschung*. Walter de Gruyter (siehe S. 36).

Jalote, P. und M. Haragopal. „Overcoming the nah syndrome for inspection deployment. In Proceedings of the 20th International Conference on Software Engineering“. In: *IEEE Computer Society,1998*, S. 371–378 (siehe S. 16).

K. Spohrer T. Kude, C. T. Schmidt (2013). „Knowledge creation in information systems development teams: The role of pair programming and peer code review.“ In: S. 213 (siehe S. 3).

Kuckartz, Udo, Thomas Ebert, Stefan Rädiker und Claus Stefer (2009). „Planung einer Evaluation mit Online-Befragung“. In: *Evaluation online: Internetgestützte Befragung in der Praxis*, S. 16–31 (siehe S. 19).

L. Harjumaa, I. Tervonen und A. Huttunen (2005). „Peer reviews in real life-motivators and demotivators“. In: IEEE, S. 29–36 (siehe S. 11, 13).

Laitenberger, Oliver, Sira Vegas und Marcus Ciolkowski (2002). „The State of the Practice of Review and Inspection Technologies in Germany“. In: Report: ViSEK/011/E (siehe S. 1, 23 f.).

Liddell, Douglas (1976). „Practical tests of 2× 2 contingency tables“. In: *The Statistician*, S. 295–304 (siehe S. 22).

M. Ciolkowski, O. Laitenberger und S. Biffl (2003). „Software reviews: The state of the practice“. In: IEEE software (siehe S. 11, 23 f.).

Mummendey, Hans Dieter und Ina Grau (2008). *Die Fragebogen-Methode. 5., überarbeitete und erweiterte Auflage* (siehe S. 36).

Mustonen-Ollila, E. und K. Lyytinen (2003). „Why organizations adopt information system process innovations: a longitudinal study using diffusion of innovation theory“. In: 13(3):275–297 (siehe S. 13).

P. C. Rigby B. Cleary, F. Painchaud (2012). „Contemporary peer review in action: Lessons from open source development“. In: S. 56–61 (siehe S. 3).

„Peer reviews in software: a practical guide“ (2002). In: ser. Addison-Wesley information technology series. Addison-Wesley (siehe S. 3).

Porst, R (2006). *Fragebogen. Ein Arbeitsbuch. Verlag für Sozialwissenschaften, Wiesbaden: 2008 25 Wachtler C, Brorsson A, Troein M. Meeting and treating cultural difference in primary care: a qualitative interview study.* Bd. 23, S. 111–115 (siehe S. 36).

Ratcliffe, J. (2009). „Moving software quality upstream: The positive impact of lightweight peer code review“. In: *Pacific NW Software Quality, Conference* (siehe S. 3).

Rigby, P. C. und C. Bird (2013). „Convergent contemporary software peer review practices. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering“. In: S. 202–212 (siehe S. 3).

Schnell, Rainer, Paul B Hill und Elke Esser (1999). „Methoden der empirischen Sozialforschung“. In: S. 292 (siehe S. 18).

Schumann, Siegfried (2012). *Repräsentative Umfrage: praxisorientierte Einführung in empirische Methoden und statistische Analyseverfahren*. Walter de Gruyter (siehe S. 36).

Vosseberg, Karin, Andreas Spillner und Mario Winter (2016). *Umfrage 2016-Softwaretest in Praxis und Forschung*. `http://www.softwaretest-umfrage.de/`. Accessed: 2017-04-04 (siehe S. 52).

9 Anhang

Es befindet sich auf der letzten Seite der Masterarbeit eine CD mit folgendem Inhalt:

· Ausarbeitung als PDF
· Importdatei der Umfrage
· Datensätze der Umfrage in verschiedenen Ausführungen

9.1 Umfrage auf Deutsch

In der Umfrage finden sich unter jeder Frage Hinweise, ob es sich es sich bei den Antworten um eine Einfach- oder Mehrfachankreuzfrage handelt. Dies wird aus Platzgründen weg gelassen und stattdessen werden umfragetypisch ein Kreis ◯ für eine Einfachauswahl, ein Rechteck ☐ für eine Mehrfachauswahl und ein * für eine Pflichtangabe verwendet.

F1*: Arbeiten Sie in einem Team, das kommerzielle Software entwickelt?
◯ Ja ◯ Nein

Team: „Team“ meint in dieser Umfrage eine Gruppe von Mitarbeitern eines Unternehmens mit einem gemeinsamen Verantwortungsbereich, die unter einer gemeinsamen Bezeichnung auftritt.

A1: Wie viele Software-Entwickler arbeiten in Ihrem Team (Sie inbegriffen)?
_____

Software-Entwickler: Im Kontext dieser Frage zählt jedes Team-Mitglied, das Code verfasst, als Software-Entwickler.

A2: In welchem Bereich entwickelt Ihr Team vorwiegend Software?
◯ Closed-Source-Entwicklung ◯ Open-Source-Entwicklung

A3: In welchem Land befindet sich Ihr Arbeitsplatz?
_____

F2*: Verwenden oder verwendeten Sie Code-Reviews?
◯ Es werden zur Zeit Code-Reviews verwendet.
◯ Es wurden in der Vergangenheit Code-Reviews verwendet, jedoch ist das Vorgehen inzwischen eingeschlafen.
◯ Es wurden in der Vergangenheit Code-Reviews verwendet und bewusst beendet.
◯ Es wurde sich gegen die Nutzung von Code-Reviews entschieden und deshalb noch nie verwendet.
◯ Es wurden noch keine Überlegungen zu Code-Reviews unternommen.

Code-Review ist eine Technik zur Qualitätssicherung von Software, für die folgendes gilt:
- ◦ Die Hauptprüfung wird von einem oder mehreren Menschen durchgeführt.
- ◦ Mindestens einer dieser Menschen ist nicht der Autor des Codes.
- ◦ Die Prüfung erfolgt hauptsächlich durch Betrachten und Lesen des Codes.
- ◦ Die Prüfung wird nach der Implementierung oder als Unterbrechung der Implementierung durchgeführt.

Die Menschen die die Prüfung durchführen, außer der Autor, werden folgend als „Reviewer“ bezeichnet.

A4: Wie groß ist Ihre Organisation?

	1 - 10	11 - 25	26 - 50	51 - 100	101 - 250	251 - 500	501 - 1000	1001 - 10000	10001 oder mehr
Mitarbeiter	◯	◯	◯	◯	◯	◯	◯	◯	◯

A5: Was ist das Hauptgeschäftsfeld Ihres Teams?

◯ Standard-Software
◯ In-house-IT
◯ Contractor/Auftragsentwicklung
◯ Software as a Service (SaaS)
◯ Sonstiges: _____

A6: Welche Rolle wird Ihnen am häufigsten zugewiesen?

◯ Projekt-Manager
◯ Software-Architekt
◯ Entwickler
◯ Tester
◯ Administrator/ IT-Betrieb
◯ Qualitätsbeauftragter
◯ Sonstiges: _____

A7: Wie erfolgt in Ihrem Team der Softwareentwicklungsprozess?

◯ Agile Softwareentwicklung (Softwareentwicklung erfolgt mit wenigen Regeln, meist iterativ und mit geringem bürokratischem Aufwand).
◯ Ad hoc (Softwareentwicklung als improvisierte Handlung, die speziell für einen Zweck entworfen wird).
◯ Klassische Entwicklung (Softwareentwicklung erfolgt durch Anlehnung an ein Vorgehensmodell mit größerem bürokratischen Aufwand).
◯ Sonstiges _____

B1: Soll jeder Entwickler in Ihrem Team an jedem Teil des Codes arbeiten können?
◯ Ja, jeder Entwickler soll überwiegend an jedem Teil des Codes arbeiten können. (Collective-Code-Ownership)
◯ Nein, jeder Entwickler soll überwiegend nur in seinem eingeschränkten Zuständigkeitsbereich arbeiten.

B2: Welches Ziel hat Ihr Team in Bezug auf die Wissensverteilung? Die Teammitglieder sollen überwiegend als [..]
◯ ..Generalisten arbeiten.
◯ ..Spezialisten arbeiten.

B3: Strebt Ihr Team eher einen kurzfristigen oder einen langfristigen Produkterfolg an?

kurzfristiger	kurzfristiger bis mittelfristiger	mittelfristiger	mittelfristiger bis langfristiger	langfristiger
◯	◯	◯	◯	◯

B4: Arbeitet die Mehrheit der Entwickler Ihres Teams an verteilten Standorten (z.B. arbeiten Ihre Teammitglieder in verschiedenen Städten)?
◯Ja ◯Nein

B5: Wie lange dauert der Veröffentlichungszyklus in Ihrem Team im Durchschnitt? Das Intervall zwischen zwei Veröffentlichungen unserer Software ist [..]
◯ .. kontinuierlich
◯ .. regelmäßig, angegeben im rechten Feld mit folgender Anzahl an Wochen:
◯ .. regelmäßig, angegeben im rechten Feld mit folgender Anzahl an Monaten:
◯ .. unregelmäßig

B6: Welches Source Code Management System (SCM) nutzen Sie?
☐ Microsoft Team Foundation Server
☐ Git
☐ Subversion
☐ Rational ClearCase
☐ Helix
☐ Mercurial
☐ Es wird kein SCM verwendet
☐ Sonstiges: _____

B7: Welchen Folgen kann ein Softwarefehler in Ihrem Team haben? Ein Softwarefehler kann: [..]
- ☐ ..die menschliche Gesundheit oder das Leben bedrohen.
- ☐ ..zu einem Systemausfall führen.
- ☐ ..sich auf das Geschäft Ihres Unternehmens auswirken.
- ☐ ..zu finanziellen Verlusten führen.
- ☐ ..die Umwelt beeinflussen.
- ☐ ..rechtliche Konsequenzen haben.
- ☐ ..zu einer Funktionsbeeinträchtigung des Systems (Service) führen.
- ☐ ..sich auf den Ruf des Unternehmens auswirken.
- ☐ Sonstiges: _____

C1: Wie wird festgelegt, welcher Code einem Review unterzogen wird?
- ◯ Das Review erfolgt auf Basis von Änderungen am Source-Code (z.B. für Pull Request, User-Story, Task oder Requirement)
- ◯ Das Review erfolgt für ein gesamtes Modul/Paket/Klasse
- ◯ Sonstiges: _____

C2: Nachdem Änderungen am Code vorgenommen wurden [..]
- ◯.. ,werden diese für unbeteiligte Entwickler sichtbar gemacht. Anschließend erfolgt das Code-Review. (Post-Commit-Review).
- ◯.. erfolgt ein Code-Review. Erst anschließend werden die Code-Änderungen für unbeteiligte Entwickler sichtbar gemacht. (Pre-Commit-Review, z.B Pull-Request).

C3: Werden in Ihrem Team wenige große Code-Reviews (mit großem Umfang) oder viele kleine Code-Reviews (mit kleinem Umfang) durchgeführt?
- ◯ Sehr großer Umfang (z.B. ein Review für alle Änderungen eines Releases)
- ◯ Großer Umfang (z.B. ein Review für jede User-Story)
- ◯ Mittelgroßer Umfang (z.B. ein Review für jede Entwicklungsaufgabe)
- ◯ Kleiner Umfang (z.B. ein Review für jeden Pull/Push)
- ◯ Sehr kleiner Umfang (z.B. ein Review für jedes Commit)

C4: Wie viele Personen (ohne den Verfasser des Codes) nehmen meistens am Review teil?

C5: Wie lange führt Ihr Team bereits Code-Review durch?

	Weniger als 1 Monat	1-3 Monate	4-6 Monate	7-12 Monate	1-2 Jahre	2-5 Jahre	über 5 Jahre
Verwendungsdauer von Code-Review:	◯	◯	◯	◯	◯	◯	◯

C6: Wie entscheidet sich in den meisten Fällen, ob ein Code-Review angesetzt werden soll?
- ◯ Der Reviewer entscheidet von Fall zu Fall, ob ein Review stattfinden soll.
- ◯ Der Autor entscheidet von Fall zu Fall, ob ein Review stattfinden soll.
- ◯ Ein Manager entscheidet von Fall zu Fall, ob ein Review stattfinden soll.
- ◯ Es gibt feste Regeln/Konventionen, ob ein Review stattfinden soll.
- ◯ Sonstiges: _____

B8: War in Ihrem Team der Code-Review-Prozess durch einen Vertrag oder ein Gesetz geregelt?
☐ Ja, basierend auf vertraglichen Regelungen.
☐ Ja, basierend auf gesetzlichen Regelungen.
☐ Nein.

B9: Konnte Ihr Team den Code-Review-Prozess eigenständig anpassen?

	trifft nicht zu	trifft eher nicht zu	teils-teils	trifft eher zu	trifft zu
Zustimmung:	◯	◯	◯	◯	◯

D1: Warum verwenden Sie Code-Review nicht mehr?

C7: Wie wurde festgelegt, welcher Code einem Review unterzogen wurde?
◯ Das Review erfolgte auf Basis von Änderungen am Source-Code. (z.B. für Pull Requests, User-Stories, Task oder Requirements)
◯ Das Review erfolgte für gesamte Module/Pakete/Klassen.
◯ Sonstiges: _____

B10: Führten vertragliche oder gesetzliche Regelungen an den Code-Review Prozess dazu, dass Code-Review nicht mehr verwendet wird?
☐ Ja, basierend auf vertraglichen Regelungen.
☐ Ja, basierend auf gesetzlichen Regelungen.
☐ Nein.

C8: Welche Tools verwendeten Sie für Code-Reviews?
☐ Gerrit
☐ Atlassian Crucible
☐ Atlassian Bitbucket Server (früher: Stash)
☐ JetBrains Upsource
☐ SmartBear Collaborator
☐ ReviewClipse
☐ Microsoft CodeFlow
☐ GitHub Pull Requests
☐ Phabricator
☐ Klocwork Cahoots
☐ Allgemeine Softwareentwicklungstools (IDE, SCM, Ticket-System/bug tracker)
☐ Sonstiges: _____

D2: Vermuten Sie, dass das Verhältnis aus Nutzen und aufzubringenden Anstrengungen von Code-Review lohnenswert ist?
◯ Ja ◯ Nein

D3: Warum vermuten Sie, dass die aufzubringenden Anstrengungen nicht lohnenswert sind?

D4: Warum vermuten Sie, dass die Verwendung von Code-Review lohnenswert ist, aber verwenden es nicht?

C9: Wie geschieht in Ihrem Team die Zuweisung von Reviewern zu Code-Reviews?
◯ Die gleichen Reviewer führen alle Reviews für ein bestimmtes Team oder Modul durch.
◯ Der Autor lädt eine Gruppe von Reviewern ein. Diese Reviewer wählen dann aus den für sie ausstehenden Reviews.
◯ Der Autor bestimmt, wer das Review durchführt.
◯ Der Reviewer wählt zwischen allen ausstehenden Reviews aus.
◯ Sonstiges: _____

C10: Welche Kommunikationsformen erfolgen in Ihrem Team während des Reviews am häufigsten?
◯ Die Kommunikation zwischen den Beteiligten erfolgt nur bei Bedarf, z.B. bei Rückfragen.
◯ Review-Anmerkungen werden sofort kommuniziert und asynchron diskutiert, z.B. per E-Mail oder in einem Review-Tool.
◯ Die Reviewer treffen sich für das Review mit dem Autor.
◯ Die Reviewer treffen sich für das Review, allerdings ohne den Autor.

C11: Wird in Ihrem Team der Code überwiegend parallel oder nacheinander von den Reviewern geprüft, wenn mehrere Reviewer an einem Review beteiligt sind?
◯ Die Prüfung durch die einzelnen Reviewer erfolgt parallel. Korrigiert wird, wenn alle Reviewer fertig sind.
◯ Die Prüfung durch die einzelnen Reviewer erfolgt nacheinander. Korrigiert wird nach jedem Reviewer.
◯ Es gibt bei uns niemals mehr als einen Reviewer.
◯ Sonstiges: _____

C12: Ändern Reviewer in Ihrem Team beim Entdecken von Fehlern den Code eigenständig?
◯ Reviewer dürfen den Code ändern und tun dies häufig.
◯ Reviewer dürfen den Code ändern und tun dies gelegentlich.
◯ Reviewer dürfen den Code ändern, aber tun dies nicht.
◯ Reviewer dürfen den Code nicht ändern.
◯ Reviewer können den Code technisch nicht ändern.
◯ Sonstiges: _____

C13: Wie teilt der Reviewer dem Autor die gefundenen Fehler mit?
◯ Hauptsächlich schriftlich.
◯ Hauptsächlich mündlich.
◯ Die Probleme werden hauptsächlich mündlich kommuniziert, aber zusätzlich schriftlich festgehalten.

C14: Welche Tools verwenden Sie für Code-Reviews?
- ☐ Gerrit
- ☐ Atlassian Crucible
- ☐ Atlassian Bitbucket Server (früher: Stash)
- ☐ JetBrains Upsource
- ☐ SmartBear Collaborator
- ☐ ReviewClipse
- ☐ Microsoft CodeFlow
- ☐ GitHub Pull Requests
- ☐ Phabricator
- ☐ Klocwork Cahoots
- ☐ Allgemeine Softwareentwicklungstools (IDE, SCM, Ticket-System/bug tracker)
- ☐ Sonstiges: _____

C15: Wie entschied sich in den meisten Fällen, ob ein Code-Review angesetzt wurde?
- ◯ Der Reviewer entschied von Fall zu Fall, ob ein Review stattfinden sollte.
- ◯ Der Autor entschied von Fall zu Fall, ob ein Review stattfinden sollte.
- ◯ Ein Manager entschied von Fall zu Fall, ob ein Review stattfinden sollte.
- ◯ Es gab feste Regeln/Konventionen, ob ein Review stattfinden sollte.
- ◯ Sonstiges: _____

E1: Welche der folgenden potentiellen Ziele des Codes-Reviews haben für Ihr Team die höchste Priorität?
Bitte ordnen Sie die Ziele mittels „drag and drop“ oder Doppel-Klick in „Ihre Rangfolge“ ein. Beginnen Sie oben in der Rangfolge mit dem am stärksten angestrebten Ziel.

	Ihre Rangfolge:
Verbesserung der Code-Qualität	
Finden von Defekten	
Wissens- / Verständniszuwachs des Reviewers	
Wissens- / Verständniszuwachs des Code-Autors	
Stärkung des gemeinsamen Verantwortungsgefühls für die Codebasis	
Finden von besseren Lösungen	
Erfüllung von Qualitätsmanagement-Vorgaben	

E2: Code-Reviews können manchmal unerwünschte Nebeneffekte haben.
Bitte ordnen Sie die Effekte mittels „drag and drop“ oder Doppel-Klick in „Ihre Rangfolge“ danach, wie problematisch sie in Ihrem Team sind. Beginnen Sie oben in der Rangfolge mit dem problematischsten Nebeneffekt.

	Ihre Rangfolge:
Erhöhter Personalaufwand	
Erhöhte Durchlaufzeit im Entwicklungsprozess	
Kritikfähigkeit bei den Entwicklern wird gefordert	

E3: Code-Reviews können manchmal unerwünschte Nebeneffekte haben.
Bitte ordnen Sie die Effekte mittels „drag and drop“ oder Doppel-Klick in „Ihre Rangfolge“ danach, wie problematisch sie in Ihrem Team wären. Beginnen Sie oben in der Rangfolge mit dem problematischsten Nebeneffekt.

	Ihre Rangfolge:
Erhöhter Personalaufwand	
Erhöhte Durchlaufzeit im Entwicklungsprozess	
Kritikfähigkeit bei den Entwicklern wird gefordert	

E4: Haben Sie persönliche Erfahrungen zu negativen Auswirkungen von Code-Reviews gemacht? Wenn ja, welche?

B11: Kann Ihr Team den Code-Review-Prozess eigenständig anpassen?

	trifft nicht zu	trifft eher nicht zu	teils-teils	trifft eher zu	trifft zu
Zustimmung:	◯	◯	◯	◯	◯

Definition Eigenständigkeit: Es können z.B. ohne Absprache mit einer höheren Personalebene Entscheidungen wie „Regelung zur Revieweranzahl“ getroffen werden.

B12: Wie ist der Umgang mit Fehlern (z.B. fehlerhaftes Handeln, Planen, Denken) in Ihrem Team? Welche der folgenden Situationen kommen häufiger vor?
☐ Es werden gemachte Fehler im Team offen angesprochen.
☐ Fehler werden gemeinsam im Team analysiert, um daraus zu lernen.
☐ Wenn durch Fehler Probleme entstehen, stehen alle im Team für einander ein.
☐ Fehler werden verschwiegen.
☐ Vorgesetzte schieben Schuld für eigene Fehler auf andere Mitarbeiter.
☐ Bei einem Fehler wird nicht nach der eigentlichen Ursache gesucht, sondern nach einem Schuldigen.
☐ Bestimmte Aufgaben werden einer Person nicht mehr zugeteilt, weil die Person in der Vergangenheit Fehler gemacht hat.

F3*: Vielen Dank für Ihre Mühen! Sie haben das Ende des Hauptteils erreicht. Haben Sie noch ca. 6-9 Minuten Zeit, weitere Fragen zu beantworten?
◯ Ja ◯ Nein

Zusatzteil

C16: Wie stellen Sie sicher, dass der Code einem Review unterzogen wird, bevor er an den Kunden freigegeben wird?
☐ Es wird manuell nach offenen Reviews geschaut, wenn ein Release näher rückt.
☐ Es wird der Code gereviewt, bevor dieser Code der blueMain-Codebase zugeführt wird. (Pre-Commit-Reviews)
☐ Es gibt eine ständige technische Teilung zwischen dem Entwicklungszweig (Development-Branch) und Freigabezweig (Release-Branch).
☐ Es werden diesbezüglich keine expliziten Maßnahmen getroffen.
☐ Sonstiges: _____

Main-Codebase: Der zentrale Codebase die von allen Entwicklern verwendet wird.

C17: Wie stellen Sie sicher, dass Code-Reviews möglichst schnell abgeschlossen werden?
☐ Reviews haben im Team eine höhere Priorität als andere Aufgaben.
☐ Es gibt eine Obergrenze an offenen Reviews. („work in progress“ limit)
☐ Die Reviewer planen feste Zeiten für Code-Reviews ein.
☐ Der Code-Autor muss dies überwachen.
☐ Keine speziellen Maßnahmen.
☐ Sonstiges: _____

C18: Verwenden Sie während des Code-Reviews eine Checkliste?
◯ Ja ◯ Nein

Checkliste:Liste zum Überprüfen mit Fragen oder Hinweisen.

C19: Wie ist bei den folgenden Fällen die Anzahl aller am Review beteiligten Reviewer (ohne den Verfasser des Codes)? Die übliche Teilnehmerzahl an einem Review beträgt [..]

Bitte klicken Sie Punkte aus der Liste an, bei denen eine Unterscheidung in der Reviewer-Anzahl existiert. Anschließend soll eine Zahl eingetragen werden.

☐ ..für bestimmte Komponenten (erhöhte Revieweranzahl):	
☐ ..für bestimmte Komponenten (verringerte Revieweranzahl):	
☐..bei Code von unerfahrenen Autoren:	
☐..bei Code von sehr erfahrenen Autoren:	
☐..kurz vor der Software-Veröffentlichung (Release):	
☐..bei einer Codeänderung, die einen Grenzwert (z.B. Umfang der Codeänderung) überschreitet:	
☐..bei einem durch Paarprogrammierung erstellten Code:	

Anzahl: Wenn in dem jeweiligen Fall kein Review stattfindet, geben Sie bitte eine '0' ein.

C20: Wer darf in Ihrem Team Code-Reviews durchführen? Reviews werden von [..]
- ◯ .. wechselnden Reviewern durchgeführt. Jedes Teammitglied darf reviewen.
- ◯ .. wechselnden Reviewern durchgeführt. Nur bestimmte Teammitglieder dürfen reviewen.
- ◯ .. den immer gleichen Reviewern durchgeführt.

C21: Haben in Ihrem Team die Reviewer definierte Rollen/Perspektiven?
- ◯ Es gibt mindestens einen Reviewer mit einer definierten Rolle.
- ◯ Es gibt keinen Reviewer mit einer definierten Rolle.
- ◯ Sonstiges: _____

Rollen/Perspektiven: z.B. prüft der erste Reviewer aus Sicht des Nutzers und der zweite Reviewer aus Sicht des Entwicklers.

C22: Verwenden Sie zur Unterstützung des Reviewers während des Code-Reviews die statische Code-Analyse (z.B. Sonar, CheckStyle, PMD, JDepend, Lint, JSLint)?
- ◯ Ja, sehr häufig.
- ◯ Ja, gelegentlich.
- ◯ Nein.

C23: Führen Reviewer in Ihrem Team die Software zum Testen während des Code-Reviews aus?
- ◯ Ja, sehr häufig.
- ◯ Ja, gelegentlich.
- ◯ Nein.

C24: Wie gehen Teammitglieder mit Review-Anmerkungen um? Die folgende Reaktion erfolgt [..]

	..nie	..selten	..gelegentlich	..oft	..immer
Das Teammitglied ändert den Code zeitnah entsprechend der Anmerkung des Reviewers.	◯	◯	◯	◯	◯
Das Teammitglied bittet um Klärung oder um Begründung der Anmerkung.	◯	◯	◯	◯	◯
Das Teammitglied erstellt ein Ticket, um die erforderlichen Änderungen durchzuführen.	◯	◯	◯	◯	◯
Das Teammitglied ignoriert die Anmerkungen.	◯	◯	◯	◯	◯

C25: Wertet Ihr Team systematisch Review-Daten aus (z.B. in Form von Metriken wie der Review-Effizienz)?

◯ Ja ◯ Nein

B13: Sind in Ihrem Team die Anforderungen bzgl. des Code-Review-Prozesses vertraglich oder gesetzlich geregelt?

☐ Ja, basierend auf vertraglichen Regelungen.
☐ Ja, basierend auf gesetzlichen Regelungen.
☐ Nein.

B14: Wenn in Ihrem Team eine Entscheidung zwischen den unten genannten Zielen getroffen werden muss, welches wird als wichtiger und welches als weniger wichtig eingestuft?

Bitte ordnen Sie die Ziele mittels „drag and drop“ oder Doppel-Klick in „Ihre Rangfolge“ ein.
Beginnen Sie oben in der Rangfolge mit dem wichtigsten Ziel.

	Ihre Rangfolge:
Hohe Qualität.	
Geringer Aufwand.	
Geringe Produkteinführungszeit. (’time-to-market’)	

B15: Gibt es große Wissensunterschiede in Ihrem Team? Wieviele Teammitglieder sind [..]

	.. sehr wenig erfahren?	.. wenig erfahren?..	erfahren?	.. sehr erfahren?
Keine Teammitglieder.	◯	◯	◯	◯
ca. 10% der Teammitglieder.	◯	◯	◯	◯
ca. 30% der Teammitglieder.	◯	◯	◯	◯
ca. 50% der Teammitglieder.	◯	◯	◯	◯
ca. 70% der Teammitglieder.	◯	◯	◯	◯
ca. 90% der Teammitglieder.	◯	◯	◯	◯
Alle Teammitglieder.	◯	◯	◯	◯

B16: Wie groß ist die Codebase der Software an der Ihr Team zur Zeit entwickelt? (in Lines of Code, mit Kommentaren, ohne automatisch generierten Code)

bis 10.000	10.001 - 50.000	50.001 - 100.000	100.001 - 500.000	500.001 - 1.000.000	1.000.001 - 10.000.000	10.000.001 und mehr
◯	◯	◯	◯	◯	◯	◯

Codebase: Gesamtheit der zu dem Projekt gehörenden Quelltextdateien.

B17: Unterteilen Sie User Stories in Entwicklungsaufgaben?

◯ Ja ◯ Nein

B18: Wie intensiv nutzen Sie folgende Hilfsmittel außerhalb von Code-Reviews?

	nicht	wenig intensiv	mittelmäßig intensiv	ziemlich intensiv	sehr intensiv
Tests	◯	◯	◯	◯	◯
Statische Code-Analyse (z.B. Sonar, Checkstyle, PMD, JDepend, Lint)	◯	◯	◯	◯	◯

B19: Wie intensiv nutzen Sie folgende Hilfsmittel?

	nicht	wenig intensiv	mittelmäßig intensiv	ziemlich intensiv	sehr intensiv
Tests	◯	◯	◯	◯	◯
Statische Code-Analyse (z.B. Sonar, Checkstyle, PMD, JDepend, Lint)	◯	◯	◯	◯	◯

B20: Wie häufig nutzt Ihr Team die folgenden Techniken?

	nie	selten	gelegentlich	oft	immer
Diskussion über Anforderungen vor der Implementierung	◯	◯	◯	◯	◯
Diskussion über Design-Alternativen vor der Implementierung	◯	◯	◯	◯	◯
Verwendung von „Paar-Programmierung“	◯	◯	◯	◯	◯

B21: Welches Ticket-System verwenden Sie?

◯ JIRA
◯ Bugzilla
◯ Team Foundation Server
◯ Trello
◯ GitHub Issues
◯ Basecamp
◯ FogBugz
◯ Es wird kein Ticket-System verwendet.
◯ Sonstiges: _____

Entschuldigung. Leider gehören Sie nicht zur genannten Zielgruppe. Die Zielgruppe sind Teams, die kommerziell Software entwickeln.

Ich bedanke mich trotzdem bei Ihnen für Ihre Bemühungen und freue mich, wenn Sie den Fragebogen bis zum Schluß ausfüllen.

G1: Woher haben Sie von dieser Umfrage erfahren?
◯ Persönlich von einem der Organisatoren der Umfrage.
◯ Von Freunden/Bekannten, durch Mundpropaganda.
◯ Von einer Mailingliste.
◯ Aus einem Internetforum.
◯ Sonstiges.

G2: Haben Sie weitere Anmerkungen, z.B. zu Code-Reviews oder zu Problemen bei der Bearbeitung des Fragebogens?

G3: Möchten Sie über unsere Ergebnisse informiert zu werden?
◯ Ja ◯ Nein

G4: Dürfen wir Sie für weitere Fragen in Zusammenhang mit unserer Forschung via E-Mail kontaktieren?
◯ Ja ◯ Nein

G5: Bitte hinterlassen Sie Ihre E-Mail-Adresse:

9.2 Umfrage auf Englisch

F1*: Do you work in a team that develops commercial software?
◯ Yes ◯ No

Team: "Team" in this survey stands for an identifiable group of employees with shared responsibilities within a company.

A1: How many software developers are working in your team (you included)?
_____

Software developer: In the context of this question, counts every team member who writes code as a developer.

A2: In which area does your team primarily develop software?
◯ Closed source development ◯ Open source development

A3: In which country is your workplace?
_____

F2*: Do you use code reviews or have used it previously?
◯ Code reviews are currently used.
◯ Code reviews have been used in the past, but are no longer a used procedure.
◯ Code reviews have been used in the past and were deliberately discontinued.
◯ It has been decided against the use of code reviews and they were therefore never used.
◯ No considerations to code reviews were made so far.

Code review is a software quality assurance activity with the following properties:
- The main checking is done by one or several humans.
- At least one of these humans is not the code's author.
- The checking is performed mainly by viewing and reading source code.
- It is performed after implementation or as an interruption of implementation.

The humans performing the checking, excluding the author, are called "reviewers".

A4: How big is your organization?

	1 - 10	11 - 25	26 - 50	51 - 100	101 - 250	251 - 500	501 - 1000	1001 - 10000	10001 or more
employees:	◯	◯	◯	◯	◯	◯	◯	◯	◯

A5: What is the main business of your team?
◯ Standard software
◯ In-house IT
◯ Contractor/order development
◯ Software as a Service (SaaS)
◯ Other: _____

A6: Which role is most frequently assigned to you?
◯ Project Manager
◯ Software Architect
◯ Developer
◯ Tester
◯ Administrator / IT Operation
◯ Quality Assurance Representative ◯ Other: _____

A7: How is the software development process in your team done?
◯ Agile software development process (software development takes place with few rules, mostly iterative and with little bureaucratic effort).
◯ Ad hoc process (software development as an improvised action, specifically designed for a purpose).
◯ Classical development process (software development is based on an procedure model with greater bureaucratic complexity).
◯ Other _____

B1: Should every developer in your team be able to work on any part of the code?
◯ Yes, mostly every developer should be able to work on any part of the code. (Collective Code Ownership)
◯ No, mostly every developer should work in his restricted area of responsibility.

B2: What is the goal of your team in terms of knowledge distribution? The team members should work mostly as [..]
◯ .. generalists.
◯ .. specialists.

B3: Is your team striving towards short-term or long-term success of the product?

Short-term	Short-term to medium-term	Medium-term	Medium to long-term	Long-term
◯	◯	◯	◯	◯

B4: Does the majority of your team's developers work in different locations (for example, are your team members working in different cities)?
◯ Yes ◯ No

B5: How long does the release cycle take on average in your team? The interval between two releases of our software is [..]

◯ .. continuous
◯ .. regular, as indicated in the field on the right in weeks:
◯ .. regular, as indicated in the field on the right in months:
◯ .. irregular

B6: Which source code management system (SCM) do you use?

☐ Microsoft Team Foundation Server
☐ Git
☐ Subversion
☐ Rational ClearCase
☐ Helix
☐ Mercurial
☐ No SCM is used
☐ Other: _____

B7: What are the consequences of a software error in your team? A software error can [..]

☐ ..threaten human health or life.
☐ ..cause a system failure.
☐ ..affect the business of your company.
☐ ..lead to financial losses.
☐ ..affect the environment.
☐ ..have legal consequences.
☐ ..cause a system malfunction (service).
☐ ..affect the reputation of the company.
☐ Other: _____

C1: When you do a review: How is the code that has to be reviewed normally determined?

◯ The review is done based on changes performed to source code (for example for pull request, user story, task or requirement)
◯ The review is done for a whole module/package/class
◯ Other: _____

C2: After changes to the code have been done [..]

◯ .. they are made visible to uninvolved developers. Subsequently, the code review takes place. (post commit review).
◯ .. a code review is done. Only then are the code changes made visible to uninvolved developers. (pre commit review, e.g. pull requests).

C3: Are there few large code reviews (with a large scope) or many small code reviews (with a small scope) done by your team?

◯ Very big extent (e.g. a review for all changes of releases)
◯ Large scope (e.g. a review for each user story)
◯ Medium-sized scope (e.g. a review for each development task)
◯ Small scope (e.g. a review for each pull / push)
◯ Very small extent (e.g. a review for each commit)

C4: How many people (without the author of the code) usually take part in the review?

C5: How long has your team already used code review?

	Less than 1 month	1-3 months	4-6 months	7-12 months	1-2 years	2-5 years	Over 5 years
Use duration of code review:	◯	◯	◯	◯	◯	◯	◯

C6: How does it get decided in most of the cases whether a code review is should take place?
◯ The reviewer decides on a case-by-case basis whether a review should take place.
◯ The author decides on a case-by-case basis whether a review should take place.
◯ A manager decides on a case-by-case basis whether a review should take place.
◯ There are fixed rules / conventions as to whether a review should take place.
◯ Other: _____

B8: Was your team's code review process regulated by a contract or by law?
☐ Yes, based on contractual regulations.
☐ Yes, based on legal regulations.
☐ No.

B9: Could your team independently customize the code review process?

	Disagree	Rather disagree	Neither agree nor disagree	Rather agree	Agree
Agreement:	◯	◯	◯	◯	◯

D1: Why you do not use code review anymore?

C7: When you did a review: How was the code that had to be reviewed normally determined?
◯ The review was done based on changes performed to source code (for example for a pull request, user story, task or requirement)
◯ The review was done for a whole module/package/class.
◯ Other: _____

B10: Did a change in contractual or legal regulations of the code review process lead to the code review being discontinued?
☐ Yes, based on contractual regulations.
☐ Yes, based on legal regulations.
☐ No.

C8: Which tools did you use for code reviews?
- ☐ Gerrit
- ☐ Atlassian Crucible
- ☐ Atlassian Bitbucket Server (formerly: Stash)
- ☐ JetBrains Upsource
- ☐ SmartBear Collaborator
- ☐ ReviewClipse
- ☐ Microsoft CodeFlow
- ☐ GitHub Pull Requests
- ☐ Phabricator
- ☐ Klocwork Cahoots
- ☐ General software development tools (IDE, SCM, ticket system / bug tracker)
- ☐ Other: _____

D2: Do you think that the relation between the benefits and the efforts of conducting code review is worthwhile?
◯ Yes ◯ No

D3: Why do you think that the efforts to be made are not worthwhile?

D4: Why do you not use it, if you think that the use of code review is worth the effort?

C9: How is the assignment of reviewers to code reviews done in your team?
- ◯ The same reviewers perform all the reviews for a particular team or module.
- ◯ The author invites a group of reviewers. These reviewers then choose from their pending reviews.
- ◯ The author decides who performs the review.
- ◯ The reviewer chooses from all pending reviews.
- ◯ Other: _____

C10: What forms of communication are most common in your team during the review?
- ◯ The communication between the parties is only done on-demand, e.g. for questions.
- ◯ Review notes will be communicated instantly and discussed asynchronously, e.g. via e-mail or within a review tool.
- ◯ The reviewers meet for the review with the author.
- ◯ The reviewers meet for the review without the author.

C11: Is the code mostly reviewed in parallel or sequentially by the reviewers in your team, when several reviewers are involved in a review?
- ◯ The singular reviewers work in parallel. Review remarks will be corrected when all reviewers have finished their work.
- ◯ The singular reviewers work sequentially. Review remarks will be corrected once each time a reviewer has finished its work.
- ◯ There is never more than one reviewer involved.
- ◯ Other: _____

C12: Do reviewers modify the code on their own when errors are detected?
- ◯ Reviewers are allowed to change the code and do so frequently.
- ◯ Reviewers are allowed to change the code and do so occasionally.
- ◯ Reviewers are allowed to change the code, but do not do it.
- ◯ Reviewers are not allowed to change the code.
- ◯ Reviewers can not change the code technically.
- ◯ Other: _____

C13: How does the reviewer report found errors to the author?
- ◯ Mainly in writing.
- ◯ Mainly orally.
- ◯ Errors are mainly communicated orally, but additionally recorded in writing.

C14: What tools do you use for code reviews?
- ☐ Gerrit
- ☐ Atlassian Crucible
- ☐ Atlassian Bitbucket Server (formerly: Stash)
- ☐ JetBrains Upsource
- ☐ SmartBear Collaborator
- ☐ ReviewClipse
- ☐ Microsoft CodeFlow
- ☐ GitHub Pull Requests
- ☐ Phabricator
- ☐ Klocwork Cahoots
- ☐ General software development tools (IDE, SCM, ticket system / bug tracker)
- ☐ Other: _____

C15: How was it decided in most cases whether a code review had to be done?
- ◯ The reviewer decided on a case-by-case basis whether a review should take place.
- ◯ The author decided on a case-by-case basis whether a review should take place.
- ◯ A manager decided on a case-by-case basis whether a review should take place.
- ◯ There were fixed rules / conventions, whether a review should be taken.
- ◯ Other: _____

E1: Which of the following potential goals of code review have the highest priority for your team?
Double-click or drag-and-drop items in the left list to move them to „Your ranking“ on the right.
Start at the top with the goal that you are aiming for the most.

	Your ranking:
Improving the code quality	
Finding defects	
Knowledge / increasing the understanding of the reviewer	
Knowledge / increasing the understanding of the code author	
Strengthen the sense of shared responsibility for the codebase	
Finding better solutions	
Fulfillment of quality management requirements	

E2: Code reviews can sometimes have undesired side effects.
Please sort the mentioned effects according to how problematic they are in your team. Double-click or drag-and-drop items in the left list to move them to „Your ranking“ on the right. Start at the top with the most problematic side effect.

	Your Ranking:
Increased personnel expenses	
Increased time to finish a story/requirement	
Ability to take criticism required by the developers	

E3: Code reviews can sometimes have undesired side effects.
Please sort the mentioned effects according to how problematic they would be in your team. Double-click or drag-and-drop items in the left list to move them to „Your ranking“ on the right. Start at the top with the most problematic side effect.

	Your ranking:
Increased personnel expenses	
Increased time to finish a story/requirement	
Ability to take criticism required by the developers	

E4: Have you made personal experiences to negative effects of code reviews? If so, which?

B11: Can your team customize the code review process independently?

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
agreement:	◯	◯	◯	◯	◯

Definition Independence: For example, without consulting a higher management level, decisions such as „regulation on the number of reviewers“ can be made autonomously.

B12: What is the handling of errors (such as mistakes in planning or thinking) in your team? Which of the following situations are more common?

☐ Mistakes are openly discussed in the team.
☐ Errors are analyzed jointly in the team in order to learn from them.
☐ If problems arise through errors, everyone in the team stands in for each other.
☐ Mistakes are concealed.
☐ Supervisors pass on guilt of their own mistakes to other employees.
☐ In the case of an error, the search is not aimed at the actual cause, but for a guilty person.
☐ Certain tasks are no longer assigned to a person because the person has made mistakes in the past.

F3*: Thank you for your effort! You have reached the end of the main part. Do you still have about 6-9 minutes to answer further questions?

◯ Yes ◯ No

Additional part

C16: How do you ensure that the code is reviewed before it is released to the customer?

☐ Open reviews are manually looked out for when a release is approaching.
☐ The code is reviewed before it is included into the bluemain codebase. (Pre-commit reviews)
☐ There is a constant technical separation between the development branch and the release branch.
☐ No explicit measures are taken.
☐ Other: _____

Main codebase: The central codebase used by all developers.

C17: How do you ensure that code reviews are completed as soon as possible?

☐ Reviews have a higher priority than other tasks in the team.
☐ There is an upper limit on the number of open reviews. („Work in progress“ limit)
☐ The reviewers schedule fixed times for code reviews.
☐ The code author has to monitor this.
☐ No special measures.
☐ Other: _____

C18: Do you use a checklist during the code review?
◯ Yes ◯ No

Checklist: Checklist with questions or hints.

C19: What is the number of all reviewers (without the code author) in the following cases? The usual number of participants at a review is [..]

Please click on points from the list for which there is a distinction in the number of reviewers. Afterwards a number should be insert.

☐ ..for certain components (increased number of reviewers):	
☐ ..for certain components (reduced number of reviewers):	
☐..with code of inexperienced authors:	
☐..with code of very experienced authors:	
☐..shortly before the software release:	
☐..with a code change that exceeds a limit (e.g. the size of the code change):	
☐..with a code change done by means of pair programming:	

number: Please insert '0' when no review is done.

C20: Who is allowed to do code reviews in your team? Reviews are performed by [..]
◯ .. changing reviewers. Each team member may perform reviews.
◯ .. changing reviewers. Only certain team members may perform reviews.
◯ .. the same reviewers all the time.

C21: Do the reviewers have defined roles/perspectives in your team?
◯ There is at least one reviewer with a defined role.
◯ There is no reviewer with a defined role.
◯ Other: _____

roles/perspectives: e.g. the first reviewer checks from the perspective of a user and the second reviewer from the perspective of a developer.

C22: Do you use static code analysis (e.g., Sonar, Check Style, PMD, JDepend, Lint, JSLint) to support the reviewers during the code reviews?
◯ Yes, very often.
◯ Yes, sometimes.
◯ No.

C23: Do reviewers in your team run the software for testing during the code review?
◯ Yes, very often.
◯ Yes, sometimes.
◯ No.

C24: How do team members deal with review comments? The following reaction occurs [..]

	..never	..rarely	..occasionally	..often	..always
The team member changes the code promptly according to the comment of the reviewer.	◯	◯	◯	◯	◯
The team member asks for clarification or justification of the comment.	◯	◯	◯	◯	◯
The team member creates a ticket to make the necessary changes.	◯	◯	◯	◯	◯
The team member ignores the annotations.	◯	◯	◯	◯	◯

C25: Is your team systematically evaluating review data (for example, in terms of metrics such as review efficiency)?

◯ Yes ◯ No

B13: Are the requirements for the code review process in your team contractually or legally regulated?

☐ Yes, based on contractual regulations.
☐ Yes, based on legal regulations.
☐ No.

B14: If your team needs to make a decision between the objectives listed below, which one is considered more important and which is considered less important?

Double-click or drag-and-drop items in the left list to move them to the right - most important goal item should be on the top right, moving through to your lowest ranking item.

	Ihre Rangfolge:
High quality.	
Lower expenses.	
Shorter product launch time. ('Time-to-market' time)	

B15: Are there any large knowledge differences in your team? How many team members are [..]

	.. very little experienced?	.. little experienced?	.. experienced?	.. very experienced?
No team members.	◯	◯	◯	◯
approx. 10approx. 30approx. 50approx. 70approx. 90All team members	◯	◯	◯	◯

B16: How big is the codebase of the software your team is currently developing? (In Lines of Code, with comments, without auto-generated code)

Up to 10,000	10,001 - 50,000	50,001 - 100,000	100,001 - 500,000	500,001 - 1,000,000	1,000,001 - 10,000,000	10,000,001 and more
◯	◯	◯	◯	◯	◯	◯

Codebase: The entirety of the source files which are part of a project.

B17: Do you partition user stories into development tasks?

◯ Yes ◯ No

B18: How intensively do you use the following tools outside of code reviews?

	Not	Little intense	Average intensive	Rather intense	Very intensive
Testing	◯	◯	◯	◯	◯
Static code analysis (e.g. Sonar, Check-Style, PMD, JDepend, Lint, JSLint)	◯	◯	◯	◯	◯

B19: How intensively do you use the following tools outside of code reviews?

	Not	Little intense	Average intensive	Rather intense	Very intensive
Testing	◯	◯	◯	◯	◯
Static code analysis (e.g. Sonar, Check-Style, PMD, JDepend, Lint, JSLint)	◯	◯	◯	◯	◯

B20: How often does your team use the following techniques?

	never	rarely	occasionally	often	always
Discussion about requirements before implementation	◯	◯	◯	◯	◯
Discussion about design alternatives before implementation	◯	◯	◯	◯	◯
Use of „pair programming“	◯	◯	◯	◯	◯

B21: Which ticket system do you use?

◯ JIRA
◯ Bugzilla
◯ Team Foundation Server
◯ Trello
◯ GitHub Issues
◯ Basecamp
◯ FogBugz
◯ No ticket system is used.
◯ Other : _____

Sorry but unfortunately, you are not part of the target group. The target group are teams that are developing software commercially.

However, I would like to thank you for your efforts and time and I would be grateful if you could complete the questionnaire up to the end.

G1: How did you hear about this survey?
◯ Personally by one of the organizers of the survey.
◯ By friends / acquaintances, through word of mouth.
◯ From a mailing list.
◯ From an internet forum.
◯ Others.

G2: Do you have other comments to make, e.g. regarding code reviews in general or regarding problems you encountered while completing the questionnaire?

G3: Would you like to be informed about our results?
◯ Ja ◯ Nein

G4: May we contact you via e-mail for further questions regarding our research?
◯ Ja ◯ Nein

G5:Please leave your e-mail address: