HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

DIPLOMARBEIT

Herr Ing.
Philipp Januskovecz

Wirtschaftliche Analyse des technischen Softwareentwicklungsprozesses einer Webapplikation

Mittweida, 2021

Fakultät Wirtschaftsingenieurwesen

DIPLOMARBEIT

Wirtschaftliche Analyse des technischen Softwareentwicklungsprozesses einer Webapplikation

Autor:
Herr Ing.

Philipp Januskovecz

Studiengang:
Wirtschaftsingenieurwesen

Seminargruppe:
KW17wNA

Erstprüfer:
Prof. Dr. rer. pol. Andreas Schmalfuß

Zweitprüfer:
Prof. Dr. rer. oec. Johannes N. Stelling

Einreichung:
Mittweida, 26.07.2021

Verteidigung/Bewertung:
Mittweida, 2021

Faculty Industrial Engineering

DIPLOM THESIS

Economic analysis of the technical software development process of a web application

author:
Mr. Ing.

Philipp Januskovecz

course of studies:
Economics for Engineers

seminar group:
KW17wNA

first examiner:
Prof. Dr. rer. pol. Andreas Schmalfuß

second examiner:
Prof. Dr. rer. oec. Johannes N. Stelling

submission:
Mittweida, 26.07.2021

defence/ evaluation:
Mittweida, 2021

Bibliografische Beschreibung:

Januskovecz, Philipp:
Wirtschaftliche Analyse des technischen Softwareentwicklungsprozesses einer Webapplikation (mit besonderem Fokus auf die eingesetzten Werkzeuge/Tools, entwickelt durch ein skalierbares und flexibles Team) - 2021 - 5, 62, 9 S.
Mittweida, Hochschule Mittweida, Fakultät Wirtschaftsingenieurwesen, Diplomarbeit, 2021

Referat:

Die vorliegende Arbeit befasst sich mit der wirtschaftlichen Analyse des technischen Softwareentwicklungsprozesses einer Webapplikation, entwickelt durch ein skalierbares und flexibles Team. Der Hauptfokus liegt darin, die eingesetzten Werkzeuge/Tools zu beurteilen. Das Ziel der Arbeit ist die Kernaussage, „Welche Werkzeuge (Tools) soll man nützen bzw. welche Prozesse einhalten, um die nachhaltige Entwicklung einer Software mit unterschiedlich großen Teams zu ermöglichen?“, beantworten zu können.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Es konnten keine Einträge für ein Tabellenverzeichnis gefunden werden.

Abkürzungsverzeichnis

IT	Informationstechnik
IDE	Integrated Development Environment
SCM	Source Code Management
MVC	Model-View-Controller
MVVM	Model-View-View-Model
URL	Uniform Resource Locator
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
XSS	Cross-Site-Scripting
API	Application Programming Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol

1 Einführende Bemerkung

1.1 Einführung in die Problemstellung

Computersysteme, insbesondere dessen Software bestimmen unser Leben, wahrscheinlich mehr als uns selbst bewusst ist. Zahlungen im Supermarkt, Zug- oder Autofahrten oder Operationen sind von Software abhängig. Dies waren nur ein paar Beispiele und wir alle wollen nicht darüber nachdenken, welche Folgen es in manchen Situationen für uns hätte, sollten diese Softwaresysteme ausfallen beziehungsweise nicht in gewohnter Weise funktionieren.

Aufgrund der essenziellen Rolle von Computerprogrammen in der heutigen Gesellschaft, liegt es auf der Hand, dass es davon immer mehr gibt und auch größere Mengen in der Zukunft erwartet werden.

Das Paradoxe in der Softwareentwicklung ist, dass Personen, welche keine Erfahrung mit der Programmierung haben, leicht über den Aufwand und der Qualität so mancher implementierten Funktionalität getäuscht werden können. Einzelne bestimmte Anwendungsfälle, ohne Rücksicht auf verschiedene Umweltbedingungen wie Endgeräte, hohe Lasten und Rückverfolgungen, sind meist in kurzer Zeit implementiert. Der Nutzen bei einem nachlässigen Softwareentwicklungsstil führt jedoch nach kurzer Zeit mit an Sicherheit grenzender Wahrscheinlichkeit zur Enttäuschung.

Ein Beispiel dazu, um das in der Theorie beschriebene Problem zu verdeutlichen. Ein Webshop, welcher nicht dynamisch auf verschiedene Produkte erweitert werden kann. Der technisch nur in einem bestimmten Betriebssystem, in einem ausgewählten Browser, mit einer festgelegten Bildschirmauflösung und nicht wie angenommen auf iOS, Android, Huawei, Windows, Mac OSX, Linux und allen vorstellbaren Bildschirmauflösungen vom iPhone Mini, bis zu einem 32 Zoll 4K Monitor lauffähig ist. Wenn mehrere Benutzer gleichzeitig ein Produkt erwerben wollen, aus diesem Grund die Server zusammenbrechen, dadurch genau in diesem Moment die Zahlungsabwicklung eines einzelnen Kunden fehlschlägt und diese aber wegen fehlender Rückverfolgbarkeit nicht analysiert werden kann. Dann schlägt die anfängliche Zufriedenheit in herbe Enttäuschung um, es ist zu spät und zu aufwändig, diese Software noch zu retten und als Auftraggeber steht man vor einem „Scherbenhaufen“. Das investierte Geld wurde umsonst ausgegeben, das Einzige was übrig bleibt, sind verärgerte Kunden. Dies alles ist

nur passiert, weil jemand aufgrund fehlenden Wissens über professionelle Softwareentwicklung getäuscht wurde und nicht erkannt hat, dass eine Webapplikation mehr ist als ein schönes Formular auf einem Bildschirm.

Natürlich ist das Beispiel übertrieben, wobei es vermutlich Vorfälle wie diese schon viel zu viele gab. Das Problem in der Softwareentwicklung ist, dass sich aufgrund fehlender Kenntnisse der Auftraggeber und Anwender die Beurteilung auf oberflächliche Merkmale beschränkt. Aber gerade in diesem Punkt liegt die große Chance für alle IT-Techniker, ihre Expertise in Projekte einzubringen und die Softwareentwicklung vor allem in punkto Leistungsfähigkeit, Stabilität und Sicherheit voranzutreiben.

Beim Hausbau ist es für die meisten Leute klar, dass man den Keller eines Einfamilienhauses nicht mit der Schaufel per Hand ausheben wird, sondern lieber in einen Bagger investiert. In der Softwareentwicklung wird der Vorteil der Nutzung von Werkzeugen oft leider erst zu spät erkannt, was den Zusatzaufwand für den späteren Einbau in die Höhe treibt. Aus Auftraggebersicht ist es klar, dass der investierte Aufwand in automatisierten Tests, oft für die Lieferzeit lähmende Code-Reviews, komplexes Aufsetzen und Wartung von Tools für die Bereitstellung der Software und dessen Betriebsüberwachung, gerade in den ersten Phasen des Projekts nicht verstanden werden kann, da bei diesen Investitionen kein direkter Geschäftswert entsteht.

Wichtig ist aber auch, dass die richtigen und nicht alle möglichen Tools in der Softwareentwicklung eingesetzt werden. Im Synonym für Werkzeuge steckt das richtige Wort. Es handelt sich um Hilfsmittel. Sie sollen helfen, Aufwände erleichtern und die Probleme nicht komplizierter machen, wie sie sind. Ein Gartenhaus würde man auch nicht mit einem Kran bauen. Es kommt immer auf die jeweilige Situation darauf an.

1.2 Abgrenzung des Untersuchungsgegenstandes

Der Untersuchungsgegenstand dieser Diplomarbeit ist folgende Frage: Durch die Nutzung welcher Werkzeuge, Tools beziehungsweise Einhaltung von Prozessen kann die Entwicklung einer Webapplikation durch ein flexibles und skalierbares Team wirtschaftlich möglichst gut unterstützt werden? Es sollen die vorhandenen Möglichkeiten an Hilfsmittel gemäß dem Stand der Technik analysiert werden, sowie deren Einsatz auf Aufwand und Nutzen bewertet werden.

Ziel dieser Diplomarbeit ist keine wirtschaftliche Berechnung auf Kostenebene, da es ein allgemeiner Leitfaden werden soll, welche Tools verwendet und Prozesse im technischen

Softwareentwicklungsprozess eingehalten werden sollen, um eine Webapplikation nachhaltig, mit in der Größe immer wieder veränderten Teams zu entwickeln. Besonders interessant ist diese Ausarbeitung für Auftraggeber von Softwareentwicklungsprojekten um die Arbeitsweise besser zu verstehen.

1.3 Gang der Untersuchung

Im Anschluss an einer Literaturrecherche der aktuell vorhandenen Tools und Prozesse des technischen Softwareentwicklungsprozesses einer Webapplikation, sollen die ermittelten Informationen analysiert und bewertet werden, um deren Nutzen zu beurteilen.

1.4 Definitorische Grundlagen

Der Begriff Software bezeichnet ein umfangreiches Themengebiet. Es handelt sich in der Regel um ein Zusammenspiel mehrerer einzelner Programme, welche auf einer bestimmten Hardware laufen und mit expliziten Daten Informationsverarbeitung bereitstellen. Ein Teil der Software ist aber auch die notwendige Dokumentation, um den Betrieb und die Weiterentwicklung zu ermöglichen. [1]

Bei einem Programm handelt es sich um eine Verarbeitungsvorschrift bzw. einen Algorithmus, welcher in einer Programmiersprache angefertigt wird, und durch die Eingabe bestimmter Daten auf einem Computer ausgeführt wird und als Ergebnis Daten liefert. [2]

Mehrere Programme, welche in der selben Programmiersprache entwickelt wurden, bilden eine sogenannte Softwarekomponente. Das Charakteristische an einer Softwarekomponente ist dessen geschlossene und eigenständige Lauffähigkeit. [3]

Einige Softwarekomponenten bilden gemeinsam ein Softwaresystem. Dies zeichnet sich durch eine Schnittstelle zum Betriebssystem und zum Benutzer aus. Das System, auf dem die Software ausgeführt wird, nennt man Betriebssystem. [4]

---

[1] Vgl. [Einführung-Softwaretechnik-2021], Seiten 16-17

[2] Vgl. [Einführung-Softwaretechnik-2021], Seite 18

[3] Vgl. [Einführung-Softwaretechnik-2021], Seite 20

Die Aufschlüsselung eines Softwaresystems in mehrere Softwarekomponenten, sowie diese weiter in Programme aufzutrennen und die Schnittstellen zwischen allen Bestandteilen festzulegen, ist Teil des Fachgebiets der Softwarearchitektur. [5]

[4] Vgl. [Einführung-Softwaretechnik-2021], Seite 20

[5] Vgl. [Einführung-Softwaretechnik-2021], Seite 21

2 Theoretische Grundlagen der Softwareentwicklung

2.1 Fachlicher Entwicklungsprozess

Der Prozess der Softwareentwicklung hat viele Ähnlichkeiten mit dem Bau eines Gebäudes. Beide Projekte entstehen mit einer Idee, dann werden Pläne ausgearbeitet und in Arbeitspakete niedergeschrieben. Anschließend werden diese Arbeitsaufträge nacheinander abgearbeitet, bis das fertige Gebäude beziehungsweise die fertige Software genutzt werden kann. Bei beiden Arten von Projekten ist es notwendig den kompletten Prozess, von der initialen Vorstellung bis hin zum ausgereiften System abzubilden.[6] Ganz nach dem Motto, „So wie das Bauen eines Hauses mehr ist als Mauern, so ist Softwareentwicklung mehr als Programmieren.“ [7]

Weitere Parallelen zwischen Hausbau und Entwicklung einer Software sind, dass es sich bei beiden Projekten um komplexe Themen handelt. Einerseits gilt mit Unvorhergesehenem zurechtgekommen, andererseits sind bei beiden Vorhaben Menschen unterschiedlichster Fachrichtungen zur Umsetzung involviert. Dies bedeutet wiederum, dass Kommunikation zwischen den einzelnen Handelnden ein wesentliches Thema ist und somit sehr wichtig ist. Auch der Einsatz von Werkzeugen verhält sich ähnlich, um effizienter und rascher zum Ziel zu kommen. Bei den Maurern ist es die Kelle und der Hammer, wie bei den Programmierern die Entwicklungsumgebung und die Versionsverwaltung.[8] Dazu mehr im Kapitel 2.3 Tools (Werkzeuge).

Da es wichtig ist, genauso wie beim Hausbau, die Entwicklung einer Software organisiert und strukturiert abzuwickeln, haben sich bestimmte Vorgehensmodelle gebildet. Diese legen fest, wie gleichartige Projekte ablaufen, stimmen die Rolle der Beteiligten ab und stellen auch Methoden zur Aufgabenbewältigung zur Verfügung. Durch solche Standards

---

[6] Vgl. [Softwareent-wicklung-KompUndV-2020], Seiten 1-2

[7] [Softwareent-wicklung-KompUndV-2020], Seite 2

[8] Vgl. [Softwareent-wicklung-KompUndV-2020], Seiten 2-3

und Best Practices, dabei handelt es sich um anerkannte empfohlene Vorgehensweisen, wird die Komplexität der Projekte reduziert und gleichzeitig die Kommunikation und Vergleichbarkeit zu anderen Vorhaben verbessert.[9]

Grundsätzlich bauen alle Vorgehensmodelle auf demselben Basismodell auf. Die Idee wird in der Phase der Auftragserklärung als greifbares IT-Projekt ausgearbeitet. [10]

Danach beginnt die Phase der Konzeption, sofern die Auftragserklärung als inhaltlich ausgereift, wirtschaftlich empfehlenswert, das Vorhaben machbar und auch die Integrität mit den Unternehmenszielen gegeben ist. In der Konzeptionsphase wird der Leistungsumfang eines IT-Systems aufgestellt. Dabei werden unter anderem die Ziele beziehungsweise Kriterien, die Funktionalitäten und die Datenhaltung definiert. Als Ergebnis der Konzeptionsphase liegt eine vollständige Systembeschreibung, sowie ein grober Lösungsweg vor. In dieser Phase ist der Kunde als fachlicher Experte besonders gefordert. [11]

Anschließend folgt die Phase des Designs, in welcher geklärt wird, wie das System technisch aufgebaut wird. Die Architektur der Softwarekomponenten und Verbindungsart zu anderen Systemen wird in dieser Phase ausdefiniert. In dieser Phase liegt die Hauptarbeit bei den IT-Technikerexperten. [12]

Daraufhin wird mit der Programmierung begonnen. Es handelt sich um die Phase der Realisierung. Neben der Entwicklung der Anforderungen spielt in dieser Phase ebenfalls die Tätigkeit des Testens, welche aus verschiedenen Perspektiven (fachlich und technisch) ausgeführt wird, eine immense Rolle um die Qualität des gesamten Produktes zu gewährleisten. [13]

[9] Vgl. [Softwareent-wicklung-KompUndV-2020], Seiten 3-5

[10] Vgl. [AgileSoftwareentwicklung-2015], Seiten 6-10

[11] Vgl. [AgileSoftwareentwicklung-2015], Seiten 11-14

[12] Vgl. [AgileSoftwareentwicklung-2015], Seiten 14-15

[13] Vgl. [Softwareent-wicklung-KompUndV-2020], Seiten 15-16

Abschließend erfolgt die Phase der Einführung, welche die Abnahme des Systems vom Kunden, die Überführung in den Betrieb und auch die Einweisung der AnwenderInnen beinhaltet.[14]

Aus dem gerade beschriebenen Basismodell haben sich verschiedene spezielle Vorgehensmodelle entwickelt, die jeweils den Blick auf einen bestimmten Fokus legen. Daraus ergeben sich unterschiedliche Vor- beziehungsweise Nachteile. Im Folgenden werden die Modelle kurz vorgestellt. [15]

Das Wasserfallmodell ist auf der Idee aufgebaut, dass das Resultat einer Phase in die nächste Phase einfließt. Die Abschnitte gilt es strikt nacheinander abzuarbeiten. Bezüglich Rollen und Methoden stellt dieses Modell keinen Leitfaden zur Verfügung. [16]

Beim V-Modell handelt es sich um die Erweiterung des Wasserfallmodells. Dabei wird ein besonderer Fokus auf das Testen gelegt. [17]

Das Spiralmodell basiert auf dem Grundsatz, dass im Projekt immer wieder ähnliche Zyklen zu durchlaufen sind. Das Vorhaben wird in kleinere Anforderungspakete aufgeteilt, für welche jeweils die einzelnen Abschnitte, wie Konzeption, Implementierung und Tests durchlaufen werden müssen. Man nähert sich der finalen Lösung Zyklus für Zyklus an. [18]

Heutzutage werden Projekte zum Großteil nach den agilen Ansätzen der Softwareentwicklung durchgeführt. Dadurch kann einfacher auf geänderte Anforderungen reagiert werden und dem Kunden wird durchgehend in einem Projekt der maximale Geschäftswert geliefert. Der große Unterschied des agilen Ansatzes zu konventionellen Modellen ist, dass alle Arten von Risiken schon während der Entwicklung möglichst schnell identifiziert werden und gleichzeitig unterbunden werden. Die Art und Weise sich der finalen Lösung iterativ Zyklus für Zyklus anzunähern, ist bei den agilen Ansätzen ähnlich wie bei dem schon beschriebenen Spiralmodell. Durch diese Methode wird für den Kunden ein qualitativ höherwertiges Produkt entwickelt, da dem Auftraggeber jederzeit

---

[14] Vgl. [Softwareent-wicklung-KompUndV-2020], Seiten 16-18

[15] Vgl. [Softwareent-wicklung-KompUndV-2020], Seiten 23

[16] Vgl. [Softwareent-wicklung-KompUndV-2020], Seiten 24

[17] Vgl. [Softwareent-wicklung-KompUndV-2020], Seiten 25

[18] Vgl. [Softwareent-wicklung-KompUndV-2020], Seiten 26-27

der aktuelle Status präsentiert werden kann. Damit ist es für den Auftraggeber leichter vorstellbar ist, wo der Weg hinführt. Eventuelle Fehlentwicklungen können dadurch früh erkannt werden. [19]

Die Leitsätze und Prinzipien, Methoden und Prozesse agiler Softwareentwicklung werden in dem Kapitel 3.2.1 „Bestandteile agiler Softwareentwicklung“ noch näher erläutert.

2.2 Software-Engineering

Beim „Software Engineering“ handelt es sich um die Implementierungsphase des Softwareentwicklungsprozesses, der Programmierung. Die Programmierung ist der zentrale Teil bei der Entwicklung einer Software. Die Tätigkeiten in den anderen Phasen haben das Hauptziel die Programmierung bestmöglich zu unterstützen. [20]

Sowohl ein gut ausgearbeitetes Konzept als auch schnelle und zuverlässige Testrückmeldungen an die Entwickler, ob es noch Mängel gibt, ist unerlässlich für die Phase der Implementierung. So kann diese Phase besonders gut bewältigt werden. [21]

Die Implementierungsphase hat einen enormen Einfluss auf die Qualität des Endproduktes. Zum einen lässt sich diese Qualität anhand der Fehler („Bugs“) messen, zum anderen fallen noch andere Faktoren, wie die Wartbarkeit in das Gewicht. [22]

Der Code kann funktionieren, aber nicht strukturiert und durchdacht geschrieben sein, sowie keine Dokumentation und automatisierte Tests aufweisen. Diese nicht funktional beeinträchtigen Mängel büßt man mit an Sicherheit grenzender Wahrscheinlichkeit in der Wartung beziehungsweise im Betrieb der Softwareapplikation. Ein nicht strukturiert geschriebener Programmcode ist nur sehr schwer änderbar. Wenn keine Dokumentation vorhanden ist, besteht das Risiko, dass ein anderer Entwickler nur mit sehr großen Hürden zu einer bestehenden Software beitragen kann. Automatisierte Tests sind ein Muss. Durch automatisierte Tests werden Fehler durch Code Änderungen bei nicht

[19] Vgl. [AgileSoftwareentwicklung-2015], Seite 11-12

[20] Vgl. online [lerneProgrammieren-com], „Programmierung vs. Softwareentwicklung“

[21] Vgl. online [lerneProgrammieren-com], „Das Vorgehensmodell wirkt sich auf die Programmierung aus“

[22] Vgl. online [lerneProgrammieren-com], „Der Einfluss auf die Qualität“

unmittelbar betroffenen Teilen der Software erheblich vermindert. Bei großen Softwareprojekten sind automatisierte Tests eine erhebliche Aufwands- und Zeitersparnis der menschlichen Ressource, da es sich bei Regressionstests, um einfache, sich wiederholende Tätigkeiten handelt. Es wird dabei der komplette Funktionsumfang, aus Anwendersicht, einer Software durchgetestet. [23]

Wie beim Hausbau, bei dem die Qualität des Hauses abhängig von der Arbeitsleistung der Maurer ist, kann eine Software ebenfalls nur so gut sein, wie die Personen, welche sie entwickelt haben. Dazu ein passendes Zitat, um die Verbindung zwischen dem Team und der Technik zu verdeutlichen:

„Um ein Team einzuschätzen, muss man sich nur die Software ansehen, die das Team produziert. Falls diese langsam und aufgebläht ist, ist das Team langsam und aufgebläht. Falls sie schlank und schnell ist, ist das Team schlank und schnell.“ [24]

2.3 Tools (Werkzeuge)

Softwareentwicklung ist nur oder wesentlich besser mit dem Einsatz der richtigen Werkzeuge beziehungsweise Hilfsmittel möglich. Für jede Phase der Entwicklung gibt es geeignete Werkzeuge, die als Hilfe zur Umsetzung zur Verfügung stehen. Die Planungs- und Konzeptionsphase wird in der Regel durch gängige Standard Office Programme unterstützt. [25]

Der Fokus dieser Arbeit liegt auf den Tools, welche die Programmierung beziehungsweise Implementierungsphase unterstützen.

Eine Entwicklungsumgebung beziehungsweise „Integrated Development Environment“ (IDE) ist der „Werkzeugkasten“ jedes Programmierers. Das Herzstück jeder Entwicklungsumgebung ist der Code Editor, in dem die Programmzeilen geschrieben werden. Dabei handelt es sich um eine besondere Art von Texteditor, welcher die Programmierung mittels Syntaxhervorhebung, Syntaxprüfung, automatische Vervollständigung und Fehlerprüfung unterstützt. Eine weitere Funktionalität einer

---

[23] Vgl. online [lerneProgrammieren-com], „Weitere Tätigkeiten in der Implementierungsphase“

[24] [AgileSoftwareentwicklung-2015], Seite 137

[25] Vgl. [BestPracticeSoftwareEng-2010], Seiten 405-406

Entwicklungsumgebung ist die Bereitstellung und Ausführung der Software, lokal direkt auf dem Computer des Entwicklers. Dies ist notwendig, um eine Änderung so schnell und unkompliziert wie möglich testen zu können. Weiters ist in der modernen Softwareentwicklung ein „Debugger“ nicht mehr wegzudenken. Dies ist ein Werkzeug, welches bei der Fehlersuche unterstützt. Ein Debugger kann eine Software während der Laufzeit analysieren, sowie die verschiedenen Zustände eines Programms darstellen und auch detaillierte Informationen über Fehler liefern. Zum Beispiel kann die Art eines Fehlers analysiert werden und auch die Programmzeile, in dem dieses Problem auftritt, lokalisiert werden. Die Funktionalität von Entwicklungsumgebungen wird mittels Plugins erweitert. Heute gibt es für viele verschiedene Problemstellungen Komponenten, welche man in der IDE importieren und aktivieren kann. Konfigurationen bezüglich des Code Schreibstils wie Abstände, Leerzeilen und Dokumentation werden sinnvollerweise sehr oft zentral für ein ganzes Projektteam vorgenommen. Dies erleichtert die Integration neuer Arbeitskräfte und mindert die Zulieferung fremder Teams zu einem existierenden Projekt. [26]

Source Code Verwaltungsprogramme (SCM) sind in jedem Softwareprojekt unumgänglich. Diese ermöglichen, dass innerhalb eines Teams, mehrere Entwickler an einer Software gleichzeitig arbeiten können und ihre Arbeit anschließend passend zusammengeführt wird. Des Weiteren ermöglichen SCM Systeme Versionierung eines Source Codes. Dies ist vor allem für die Nachvollziehbarkeit notwendig, damit jederzeit auf frühere Versionen des Quellcodes zugegriffen werden kann. Der Entwicklungsverlauf jedes Teiles des Programms und jeder Beitrag des einzelnen Entwicklers kann damit eruiert werden. [27]

Es wird von den Industrien, die stark auf die Informationstechnologie angewiesen sind, eine Menge Geld investiert, um die Code Qualität besser steuern und objektiv messen zu können. Statische Code Analyse Tools überprüfen die Programmzeilen auf korrekt angewandte Patterns (Muster), Redundanz und Komplexität. Code-Review ist ein weiteres Werkzeug, welches mittlerweile Stand der Technik und unverzichtbar ist. Bevor die Änderung eines Entwicklers in das gesamte Softwarepaket eingeführt wird, wird sie

[26] Vgl. [BestPracticeSoftwareEng-2010], Seiten 402-403

[27] Vgl. [BestPracticeSoftwareEng-2010], Seiten 381-388

von anderen Entwicklern angesehen und freigegeben. Damit erreicht man, dass diese auch mit der Änderung einverstanden sind und den Code verstehen. [28]

Der Kompilierungs- und Bereitstellungsprozess der Software sollte möglichst automatisiert und schnell ablaufen. Da es sich beim Zusammenbauen und Bereitstellen einer Software um eine wiederkehrende Tätigkeit handelt, die relativ viel Zeit in Anspruch nimmt und aufgrund der möglichst schnellen Bereitstellung sehr oft zu wiederholen ist, würde dies, wenn es nicht automatisiert wäre, den zusätzlichen Aufwand jedes Entwicklers immens erhöhen. Werkzeuge für diese Tätigkeit müssen die Software auf Lauffähigkeit überprüfen, kompilieren, und schließlich in der gewünschten Umgebung bereitstellen. Meist gibt es vor der echten Umgebung eine Testumgebung, diese wird häufig auch Abnahmeumgebung genannt. Dort wird die Applikation zuerst bereitgestellt. Erst nach Abnahme durch den Auftraggeber, wird die Software in das Produktionssystem eingesetzt. [29]

Werkzeuge, welche die Qualitätssicherungsphase (Testen) einer Software unterstützen, sind für die jeweiligen Testebenen zu unterscheiden. Bevorzugte Instrumente der Entwicklertests, sind der schon erwähnte Debugger und auch sogenannte Unit Tests. Dabei handelt es sich um einen eigens geschriebenen Code, welcher nur dem Zweck dient, die Funktionalität der einzelnen Methoden zu überprüfen. Dies ist in großen Projekten notwendig, da die Methoden von verschiedenen Komponenten einer Software aufgerufen werden. Somit kann eine kleine Änderung für die Komponente A die Komponente B beeinträchtigen. Der Integrationstest wird von Werkzeugen unterstützt, welche als Ziel haben, die Schnittstellen zu überprüfen. Dieser Test wird oft automatisiert durchgeführt. Mittels klar definierter Eingaben und erwarteten Ausgaben werden die Schnittstellen durchgetestet. Der Systemtest hingegen legt den Fokus auf die gesamte Applikation. Dieser wird durch Werkzeuge unterstützt welche einzelne Anwendungsfälle aus der Benutzersicht automatisiert durchspielen. Anschließend folgt der Abnahme Test des Kunden, dies erfolgt in der Regel manuell, da sich der Kunde selbst ein Bild macht, ob seine Anforderungen korrekt erfüllt wurden. Hilfsmittel werden auf dieser Ebene verwendet, um den Kunden möglichst gut zu unterstützen, die Fehler beziehungsweise Änderungswünsche in einer strukturierten Form, mit den notwendigen Daten dem Team

[28] Vgl. [CleanCode-2009], Seiten 25-43

[29] Vgl. [BestPracticeSoftwareEng-2010], Seiten 392-402

zu übermitteln. Dies ist notwendig, da aus Programmierer Sicht, um sich einen Fehler anschauen zu können, oft Informationen eine Rolle spielen, welche dem Benutzer nicht relevant erscheinen, wie Zeitstempel, Protokolle (Logging Einträge) und versteckte Fehlermeldungen. [30]

Bei der letzten Phase des Softwarelebenszyklus handelt es sich um den Betrieb und der Wartung. Dabei gibt es Hilfsmittel, welche sich grundsätzlich auf verschiedene Perspektiven aufteilen lassen. Die Abwicklung von Fehlern im täglichen Betrieb wird normalerweise durch ein Ticketing System unterstützt. Dabei können Benutzer ihre Probleme beziehungsweise Anliegen einmelden. Der Techniker benötigt eine geeignete Überwachung (Monitoring) und Logging Werkzeug des Software-Systems, damit er die Einmeldung abarbeiten kann. Die Umwelt und Umgebung rund um ein Softwaresystem ändert sich laufend und erfordert deshalb ständig Anpassungen. Aufgrund von Änderungen bei Schnittstellen, Betriebssystem und Hardware lebt eine Software kontinuierlich. Solche Anpassungen und Änderungen sind Teil des Betriebs und der Wartung. Somit existieren eine Vielzahl von Simulatoren, welche verschiedenste Umweltbedingungen von Endgeräten und Infrastrukturzustände nachbilden. [31]

[30] Vgl. [QualitätssicherungDurchSoftwaretests-2013], Seiten 1 - 38

[31] Vgl. [LehrbuchDerSoftwaretechnik-2011], Seiten 529-541

3 Flexible und skalierbare Entwicklung einer Webapplikation

3.1 Aufgabe und Ziele

Ziel dieser Diplomarbeit ist, die Nutzung von Werkzeugen im technischen Softwareentwicklungsprozess zu beurteilen, um die wirtschaftliche (nachhaltige) Entwicklung einer Webapplikation mit einem flexiblen und skalierbaren Team zu ermöglichen.

Welche Werkzeuge sichern die nachhaltige Softwareentwicklung ab, damit eine Webapplikation keine Angst vor neuer Hardware haben muss? Mit welchen Prozessen kann sichergestellt werden, heute mit fünf Entwicklern zu arbeiten und in ein paar Monaten mit 30 Entwicklern?

3.2 Möglichkeiten in den Unterbereichen

Die Bestandteile (Fachlich, Technisch und Werkzeuge) der Softwareentwicklung, welche im Kapitel davor in der Theorie erläutert wurden, werden in diesem Kapitel in der Praxis, in Bezug auf den modernen Softwareentwicklungsprozess, beschrieben. Außerdem wird die aktuelle Marktsituation bezüglich verfügbarer Tools analysiert.

3.2.1 Bestandteile agiler Softwareentwicklung

3.2.1.1 Leitsätze und Prinzipien

Respekt, Offenheit, Mut, Fokus und Selbstverpflichtung bilden die Grundwerte für das Agile Vorgehen, für die richtige Ausrichtung, um qualitativ hochwertige Software zu entwickeln. Sie wurden durch die folgenden vier Leitsätze und zwölf Prinzipien im Jahr 2001 im Agilen Manifest verdeutlicht. [32]

[32] Vgl. [AgileInDerUnternehmenspraxis-2017], Seite 17-18

Die Leitsätze der Agilen Softwareentwicklung sind:

- „Individuen und Interaktionen mehr als Prozesse und Werkzeuge“ [33]
  Die einzelnen beteiligten Personen des Projekts stehen im Zentrum. Der definierte Prozess überstimmt niemals die Wünsche beziehungsweise Bedürfnisse jedes Einzelnen. [34]
- „Funktionierende Software mehr als umfassende Dokumentation“ [35]
  Das Wesentliche an einem Softwareprojekt ist, dass sie läuft und funktioniert. Alle Aktivitäten sollen dieses gemeinsame Ziel befolgen. [36]
- „Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung“ [37]
  Der Kunde soll sein gewünschtes Softwareprodukt in der Entwicklungsphase immer wieder verwenden, um mit dem Auftraggeber die nächsten Schritte abzustimmen. Ein Beispiel zu diesem Punkt zeigt eine Analogie mit dem Hausbau. Die Sockelleisten werden nicht am Anfang des Projekts bestimmt und entschieden, sondern meist erst, wenn der Boden verlegt und die Wände gestrichen sind. [38]
- „Reagieren auf Veränderung mehr als Befolgen eines Plans“ [39]
  Änderungen sind normal. Der Kunde, beziehungsweise Anwender, soll bei seinen Änderungswünschen bestmöglich unterstützt werden. [40]

Es gibt zwölf Prinzipien nach denen in der Praxis in einem agilen Projekt gearbeitet werden soll:

1. Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen. [41]

---

[33] [AgileSoftwareentwicklung-2015], Seite 14

[34] Vgl. [AgileSoftwareentwicklung-2015], Seite 14

[35] [AgileSoftwareentwicklung-2015], Seite 14

[36] Vgl. [AgileSoftwareentwicklung-2015], Seite 14

[37] [AgileSoftwareentwicklung-2015], Seite 15

[38] Vgl. [AgileSoftwareentwicklung-2015], Seite 15

[39] [AgileSoftwareentwicklung-2015], Seite 15

[40] Vgl. [AgileSoftwareentwicklung-2015], Seite 15

2. Heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden. [42]
3. Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne. [43]
4. Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten. [44]
5. Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen. [45]
6. Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht. [46]
7. Funktionierende Software ist das wichtigste Fortschrittsmaß. [47]
8. Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können. [48]
9. Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität. [49]
10. Einfachheit die Kunst, die Menge nicht getaner Arbeit zu maximieren ist essentiell. [50]
11. Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams. [51]
12. In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an. [52]

---

[41] [AgileInDerUnternehmenspraxis-2017], Seite 8

[42] [AgileInDerUnternehmenspraxis-2017], Seite 8

[43] [AgileInDerUnternehmenspraxis-2017], Seite 8

[44] [AgileInDerUnternehmenspraxis-2017], Seite 8

[45] [AgileInDerUnternehmenspraxis-2017], Seite 8

[46] [AgileInDerUnternehmenspraxis-2017], Seite 8

[47] [AgileInDerUnternehmenspraxis-2017], Seite 8

[48] [AgileInDerUnternehmenspraxis-2017], Seite 8

[49] [AgileInDerUnternehmenspraxis-2017], Seite 8

[50] [AgileInDerUnternehmenspraxis-2017], Seite 8

[51] [AgileInDerUnternehmenspraxis-2017], Seite 8

3.2.1.2 Methoden

Unter den Agilen Methoden versteht man verschiedene Vorgehensmodelle, welche alle auf den im Kapitel davor beschriebenen Leitsätzen und Prinzipien beruhen. Die Modelle bieten einen Rahmen von definierten Prozessen und Techniken, die einzuhalten sind. Adaptive Software Development, eXtreme Programming, Clean Code und Refactoring und Kanban sind nur einige von ihnen. Das bekannteste agile Vorgehensmodell, auf dem alle anderen Modelle beruhen, ist Scrum. Für viele ist agile Softwareentwicklung gleich Scrum. Daher wird anhand dieses Vorgehensmodells, im nächsten Kapitel der agile Prozess genauer erläutert. [53]

3.2.1.3 Prozesse

Der Scrum Prozess ist die Folge sogenannter Sprints, welche eine Dauer zwischen einer und vier Wochen haben. Im Produkt Backlog sind alle offenen, noch nicht implementierten Anforderungen des Softwareprodukts enthalten. Durch die Sprintplanung wird der Aufwand der einzelnen Anforderungen geschätzt, diese werden priorisiert und der Sprint Backlog damit befüllt. Der Sprint Backlog enthält alle Anforderungen, welche für die Umsetzung eines Sprints geplant sind. Innerhalb eines Sprints werden alle notwendigen Tätigkeiten, wie Design, Programmierung und Testen vom Team eigenverantwortlich durchgeführt. Dazu ist es notwendig, dass alle Rollen, wie Designer, Entwickler und Tester im Team vertreten sind. [54]

Das Team hält jeden Tag ein sogenanntes „Daily Standup Meeting“ ab. Dies hat eine maximale Länge von 15 Minuten und sollte jeden Tag zur gleichen Zeit, am gleichen Ort stattfinden. Der beste Zeitpunkt dafür ist am Morgen, da dabei gut die Arbeit für den Tag besprochen werden kann. Jedes Teammitglied muss Auskunft geben, was es seit dem letzten Daily Standup gemacht hat, ob Probleme dabei aufgetreten sind und welche Umsetzungen bis zum nächsten Daily Standup geplant sind. [55]

---

[52] [AgileInDerUnternehmenspraxis-2017], Seite 8

[53] Vgl. [AgileInDerUnternehmenspraxis-2017], Seiten 1-9

[54] Vgl. [AgileInDerUnternehmenspraxis-2017], Seite 13

[55] Vgl. [AgileInDerUnternehmenspraxis-2017], Seiten 13-14

Dieses Meeting wird vom Scrum Master moderiert. Der Scrum Master hat die Aufgabe das Team zu motivieren und eine effiziente Arbeitsumgebung bereitzustellen, sowie jedem einzelnen Teammitglied den Rücken von Störungen freizuhalten. [56]

Zum Abschluss jedes Sprints findet das Sprint Review statt. Dabei wird vom ganzen Team dem Product Owner (Auftraggeber bzw. Auftraggebervertreter) und auch anderen Interessenten, die aktuelle Software, die Sprint Lieferung, vorgestellt und neue Funktionalitäten demonstriert. [57]

Um den Prozess sowohl für das jeweilige Projekt als auch für das Team zu verbessern und auch aus der Erfahrung zu lernen, findet immer wieder eine Retrospektive statt. Dabei nimmt das ganze Team teil und es wird besprochen, was gut läuft, was schlecht läuft und was die Konsequenzen aus diesen Erkenntnissen sind. In der folgenden Grafik ist dieser Ablauf illustriert. [58]

[56] Vgl. [AgileInDerUnternehmenspraxis-2017], Seite 14

[57] Vgl. [AgileInDerUnternehmenspraxis-2017], Seiten 14-15

[58] Vgl. [AgileInDerUnternehmenspraxis-2017], Seiten 14-16

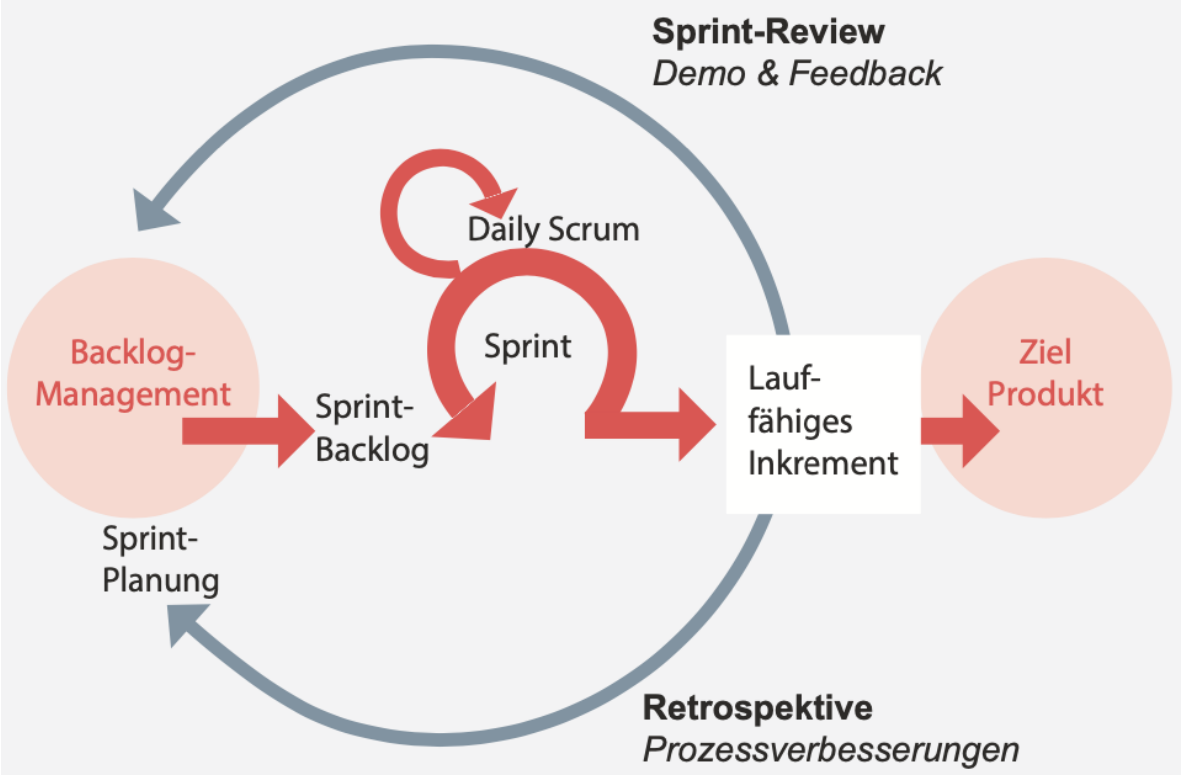


Abbildung 1: Scrum-Prozess, [AgileInDerUnternehmenspraxis-2017]

In der Regel wird der Scrum Prozess in kleinen Teams (6-10 Entwickler) angewandt. Dieser Prozess hat sich in den letzten Jahren stark weiterentwickelt, nämlich dass dieses Grundkonzept auch bei größeren Projekten angewendet werden kann. Die gängigste Anpassung des Vorgehensmodell auf größere Projekte ist das sogenannte „Scrums von Scrums“. Dabei sendet jedes kleine Scrum Team einen Vertreter in ein übergelagertes Scrum Team, in dem die gleichen Prozesse befolgt werden. [59]

Scrum ist in der Softwareentwicklung das aktuell mit Abstand am meisten verwendete Vorgehensmodell. Es hat sich vermutlich sowohl aufgrund der Einfachheit als auch aufgrund der richtigen Balance zwischen Freiraum und Kontrolle durchgesetzt. Jedes einzelne Teammitglied hat sehr viele Freiräume und den individuellen Meinungen wird hohe Beachtung geschenkt. Jede Stimme trägt bei den Arbeitsschätzungen bei. Somit kann in der Theorie nicht passieren, dass einzelne Arbeitskräfte überlastet werden. Dieses „demokratische“ Vorgehen wirkt sich positiv auf die Teammotivation und auf die

[59] Vgl. [AgilityKompakt-2009], Seite 70

Verantwortung aus, die jeder Einzelne in Bezug auf das zu entwickelnde Produkt hat. Mitentscheidung führt automatisch dazu, dass sich eine Person mehr mit ihrer Arbeit identifiziert und mehr Verantwortung für das Gesamtergebnis empfindet. Damit der freie Arbeitsstil nicht in ein Chaos läuft und das Sprintziel nicht aus den Augen verloren wird, sorgen die täglichen Daily Standups und dienen somit der Kontrolle. Bei den Planungsbesprechungen für die einzelnen Sprints können die Projektverantwortlichen sehr schnell und flexibel auf Veränderungen im Projekt reagieren. Kurz zusammengefasst kann man sagen, dass innerhalb der Sprint Leistung jedem Teammitglied viel Freiraum gelassen wird. Die Projektleitung kontrolliert jeden Tag für 15 Minuten die aktuelle Lage. Der große Hebel im Hinblick auf Verantwortlichkeiten liegt aber in der Planung der zukünftigen Arbeitsabschnitte, der sogenannten „Sprints“. [60]

3.2.2 Software-Engineering

3.2.2.1 Architektur

Architektur im Sinne der Softwareentwicklung bedeutet der Aufbau der technischen Struktur eines Systems. In der Regel wird der Hauptfokus auf die Stabilität, Erweiterbarkeit und Anpassungsfähigkeit gelegt. Weitere Schwerpunkte hängen von den langfristigen Zielen des zu entwickelnden IT-Systems ab. Die Wahl der Technologie in welcher Programmiersprache und Infrastruktur das Vorhaben umgesetzt wird, ist ebenfalls eine Disziplin der Architektur. Dabei handelt es sich um eine sehr heikle Entscheidung, ob auf Bewährtes, oder ob auf vielversprechende neue Technologien gesetzt wird. Neue Technologien haben den Vorteil, dass sie vielversprechende, einfache und performante Lösungen für bekannte Unschönheiten in der Software Branche liefern. Auf der anderen Seite darf aber nicht übersehen werden, dass über neue Technologien nur wenig Erfahrung in der Branche vorhanden ist. Somit ist eine Herausforderung Entwickler zu finden, welche sich mit der neuen Technologie auseinandersetzen wollen. Denn das zu erwartende Risiko von aufkommenden Problemen in der Implementierungsphase ist jedenfalls erhöht. [61]

---

[60] Vgl. [AgilityKompakt-2009], Seite 70

[61] Vgl. [AgileSoftwareentwicklung-2015], Seiten 137-162

Bei der Entwicklung von Webapplikationen gibt es drei gängige Paradigmen, zwischen denen es zu entscheiden gilt, wie die technische Struktur eines Systems aufgebaut werden soll. Abhängig von der Anforderung der jeweiligen Software wird zwischen folgenden drei Ansätzen entschieden:

1. Model-View-Controller (MVC): Dies ist das am häufigsten verwendete Paradigma. Dabei handelt es sich um die Separierung zwischen Daten (Model), Ansicht (View) und der fachlichen Logik (Controller) in die jeweiligen Komponenten. Die Grundidee liegt darin, die einzelnen Komponenten getrennt voneinander ändern zu können.
2. Model-View-ViewModel (MVVM): Bei diesem Ansatz ist neben den Daten auch die fachliche Logik Teil der Model Komponente. Bei der View handelt es sich wieder um die graphische Benutzeroberfläche. In der ViewModel Komponente hingegen erfolgt die Datenverarbeitung vom Model Teil für den View Teil. Die Daten werden für die Anzeige geeignet aufbereitet.
3. Three-tier organization: Applikationen nach diesem Paradigma sind nach drei Schichten aufgebaut. Erstens, der Client, das ist der Softwareteil, welcher am Endgerät des Benutzers ausgeführt wird. Zweitens der Applikationsserver, welcher das Bindeglied zwischen Client und dem Datenbankserver bildet. Beim Datenbankserver handelt es sich um die dritte Schicht, welcher sich um die Datenhaltung sowie Datenbereitstellung kümmert.

Die Architektur hat die Aufgabe nach der Entscheidung der passenden Struktur ebenfalls die geeignete Technologie auszuwählen. Auf die Möglichkeiten der Technologie wird in den nächsten beiden Unterkapiteln, Programmiersprachen und Infrastruktur, genauer eingegangen. [62]

3.2.2.2 Programmiersprachen

Jede Webapplikation hat voneinander getrennte Frontend und Backend Teile. Beim Frontend Teil handelt es sich um die Oberfläche, mit welcher der Benutzer interagiert,

[62] Vgl. online [BestWebDevelopmentFrameworks-2021], „Architectural Patterns“

welches innerhalb eines Browsers ausgeführt wird. Zumeist wird dieser Teil bei Webapplikationen heute mittels eines Frameworks entwickelt, aber dazu später mehr. [63]

Am Ende wird dieser Teil aber immer in den Programmiersprachen JavaScript (Logikteil - Prüfung und Auswertung der Benutzereingaben, Kommunikation mit dem Server, Berechnungen), HTML (Strukturierung der Seite) und CSS (Formatierung/Styling der Seite) kompiliert, da Browser nur diese Sprachen interpretieren und ausführen können. Daher handelt es sich bei den Technologien JavaScript, HTML und CSS um die einzigen Sprachen im Web. Alle anderen Programmiersprachen, mit denen Webapplikationen entwickelt werden können, sind nur Hilfsmittel/Werkzeuge (Frameworks) bei der Erstellung der Applikation und haben technisch gesehen nichts mit der laufenden Software zu tun. [64]

Beim Backend Teil handelt es sich um die Bereitstellung der benötigten Daten für das Frontend. Dieser Teil kann in vielen verschiedenen Programmiersprachen, abhängig vom Server programmiert werden. Die gängigsten Sprachen für Backend Programmierung sind .NET, Java, PHP, Python und Ruby. [65]

Heute werden die Programme nicht mehr rein in den genannten Programmiersprachen codiert, sondern es wird mit Hilfe von sogenannten Frameworks gearbeitet. Dabei handelt es sich um eine Sammlung von fertigen Funktionalitäten, welche in einer bestimmten Form verwendet und erweitert werden können. [66]

Durch die Hilfe von Frameworks ist die Entwicklung von Software schneller und billiger möglich. Für viele Anforderungen gibt es schon vorgefertigte Lösungen. Die Stabilität und Sicherheit ist dadurch auch höher. Die Komponenten wurden schon von tausenden Entwicklern verwendet und sind somit auf deren Leistung und Zuverlässigkeit geprüft. Da es heutzutage Standard ist Frameworks einzusetzen, finden sich neue Entwickler viel schneller zurecht. Ein weiterer großer Vorteil besteht in der Wartung der Systeme. Sie unterstützen von selbst neue Umgebungen. Somit hält sich der Aufwand in Grenzen, wenn neue Endgeräte unterstützt werden müssen, da diese Tätigkeit fast zur Gänze vom

[63] Vgl. online [BestWebDevelopmentFrameworks-2021], „Frontend and Backend Web Frameworks“

[64] Vgl. [WebseitenProgrammierung-2012], Seiten 6-7

[65] Vgl. online [BestWebDevelopmentFrameworks-2021], „Frontend and Backend Web Frameworks“

[66] Vgl. online [BestWebDevelopmentFrameworks-2021], „What is a Web Framework?“

Framework übernommen werden kann. Einen weiteren Vorteil bieten Frameworks mit ihren Optimierungen in Bezug auf die Performance von Softwareanwendungen. Eine erstklassige Performance zu erreichen ist mittels komplett selbst geschriebener Programme nahezu unmöglich. [67]

Laut einer Statista Umfrage im Jahr 2020 sind die drei meist verwendeten Web Frameworks, jQuery (43,3%), React (35,9%) und Angular (25,1%). [68]

Bei jQuery handelt es sich um ein Framework, welches die Erstellung von Benutzeroberflächen unterstützt. Diese Technologie ist aufgrund ihrer Einfachheit sowie „Cross Browser Unterstützung“, das ist die Fähigkeit von Webinhalten sich in jedem Browser weitestgehend ident zu verhalten, sehr beliebt. Es handelt sich um ein relativ kleines leichtgewichtetes Framework. Es bringt eine Vielzahl von Erweiterungen mit, welche von der IT-Community gefertigt werden. [69]

React ist ein Open Source Framework, welches von Facebook angeführt und unterstützt wird. Der Vorteil in React liegt darin, dass sowohl kleine, aber auch sehr große Applikationen relativ leicht und performant entwickelt werden können. [70]

Angular ist die Konkurrenz dazu von Google. Es ist ebenfalls Open Source, hat sich jedoch auf größere Applikationen spezialisiert. Der Vorteil von Angular ist ein ziemlich breites Set an Hilfsmittel. Aus diesem Grund ist es gerade in der Industrie sehr beliebt. [71]

3.2.2.3 Infrastruktur

Um eine Webapplikation betreiben zu können, benötigt man einen Server. Bei einem Server handelt es sich um einen Rechner, welcher Daten bereitstellt und über das Internet mit einer URL (Netzwerkadresse) erreichbar ist. [72]

---

[67] Vgl. online [BestWebDevelopmentFrameworks-2021], „Advantages of Using Frameworks“

[68] Vgl. online [BestWebDevelopmentFrameworks-2021], „What are the most used Web Frameworks?“

[69] Vgl. online [BestWebDevelopmentFrameworks-2021], „Most Popular Web Development Frameworks”

[70] Vgl. online [BestWebDevelopmentFrameworks-2021], „Most Popular Web Development Frameworks”

[71] Vgl. online [BestWebDevelopmentFrameworks-2021], „Most Popular Web Development Frameworks”

Stand der Technik ist nicht mehr die Software über einen Server bereitzustellen, sondern über mehrere Rechner, die sich die Arbeit geschickt untereinander aufteilen. Dabei handelt es sich um sogenannte Cloud Technologien. Über das Internet spielen mehrere Server zusammen, um sich die Bereitstellung von Software untereinander aufzuteilen. Dies hat den Vorteil das einzelne Rechner ohne Auswirkung auf die Applikationsbereitstellung ausfallen können, da die Tätigkeit sofort von einem anderen Server übernommen wird. Ein weiterer Vorteil der modernen Technologie ist dessen Skalierbarkeit. Früher war die maximale Last eines Servers von dessen Hardware und der Internetverbindung abhängig. Je mehr Rechenkapazität ein Server hatte, desto mehr Anfragen konnte dieser in derselben Zeit abwickeln. Da Webapplikationen jedoch oft stark unterschiedlicher Nutzung ausgesetzt sind, wie zum Beispiel die Vorweihnachtszeit bei Amazon, setzt man auf die Nutzung von Cloud Technologien. Denn die Abwicklung von Spitzenlasten funktioniert nur mit viel Hardware und hohen Kapazitäten eines Servers, die wiederum sehr kostenintensiv sind und nur zu einem geringen Prozentsatz genutzt werden können. [73]

Es wird grundsätzlich zwischen den drei Cloud Servicemodellen SaaS (Software as a Service), PaaS (Platform as a Service) und IaaS (Infrastructure as a Service) unterschieden, die Grenzen sind jedoch fließend. [74]

Bei SaaS handelt es sich um Software Bereitstellung über das Internet. Dies ist das bekannteste Service, da es direkt vom Endbenutzer genutzt wird. Beispiele dafür sind Office 365, ownCloud und GitHub. [75]

Die PaaS wird von Software Entwickler benutzt. Dieses Service umfasst die flexible Bereitstellung einer Umgebung, um Applikationen auszuführen. Die bekanntesten PaaS-Services sind Amazon Web Services, Microsoft Azure und OpenShift. [76]

---

[72] Vgl. [WebseitenProgrammierung-2012], Seite 12

[73] Vgl. [DerWegInDieCloud-2020], Seite 7

[74] Vgl. [DerWegInDieCloud-2020], Seite 8

[75] Vgl. [DerWegInDieCloud-2020], Seite 8-9

[76] Vgl. [DerWegInDieCloud-2020], Seite 9-10

Bei IaaS handelt es sich um die Service Bereitstellung eines virtuellen Rechenzentrums, es ist die moderne Alternative im Gegensatz zum Kauf eigener Server Hardware für das Hosting der Software. Dieses Service wird von IT-Architekten für die Infrastruktur in großen Firmen genutzt. Im Gegensatz zum PaaS Service ist bei IaaS die Justierbarkeit viel höher. Dieses Service wird unter anderem von Microsoft, Google und Amazon zur Verfügung gestellt. [77]

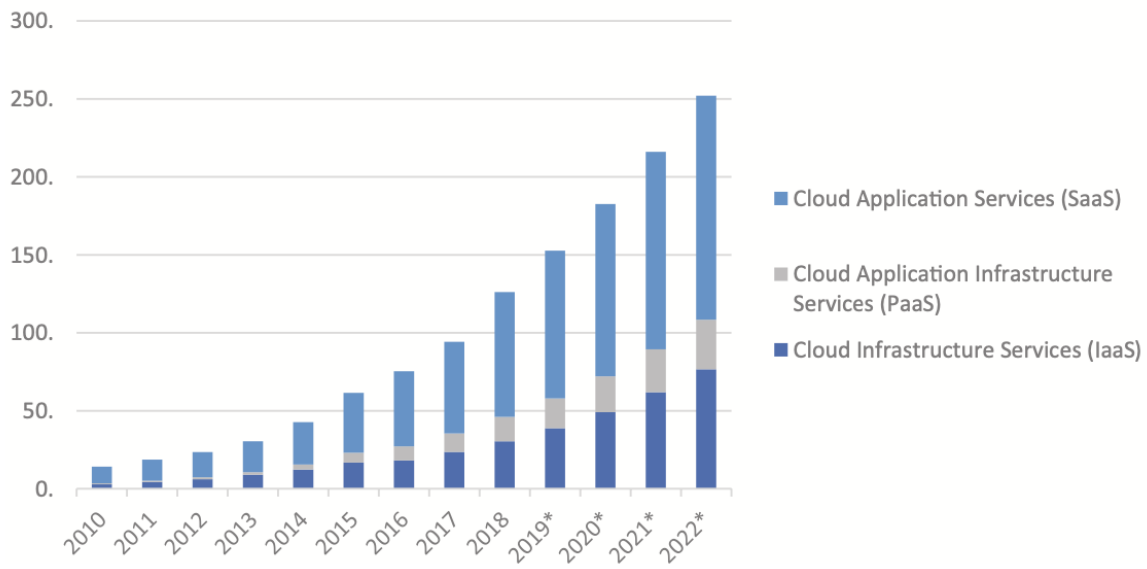


Abbildung 2: Umsatz (in Milliarden US-Dollar) von Cloud-Technologien nach Segment bis 2018 sowie Prognose bis 2022, [DerWegInDieCloud-2020]

[77] Vgl. [DerWegInDieCloud-2020], Seite 9-10

3.2.3 Tools (Werkzeuge)

3.2.3.1 Entwicklungsumgebung

Die beliebtesten IDEs (Integrated Development Environments) beziehungsweise Entwicklungsumgebungen, oder wie im Kapitel 2.3, als Werkzeugkasten eines Programmierers bezeichnet, für Webapplikationen sind Visual Studio von Microsoft, IntelliJ IDEA von Jetbrains und die kostenlose Open Source Alternative Aptana Studio 3. [78]

Visual Studio zeichnet die flinke automatische Vervollständigung aus, welche durch künstliche Intelligenz vom Stil des Entwicklers lernt. Weiters ist die live Zusammenarbeit von mehreren Entwicklern innerhalb der IDE möglich. Visual Studio hat neben der besonderen Unterstützung auf die Web-Technologien (JavaScript/HTML/CSS) einen besonderen Fokus auf .NET, C++, Python und Node.js. Außerdem hat Microsoft in ihrer Entwicklungsumgebung das Zusammenspiel mit der hauseigenen Cloud perfekt optimiert. [79]

IntelliJ IDEAs Beschlagenheit ist neben der ausgezeichneten Code Indizierung die Out of the Box Erfahrung für jeden Entwickler. Ohne zusätzlichen Konfigurationsaufwand von Erweiterungen hat IntelliJ ausgezeichnete Werkzeuge wie zum Beispiel zum Debuggen oder für die Versionskontrolle direkt an Board. IntelliJ IDEA bietet neben dem Fokus auf Web-Technologien eine besondere Unterstützung für Java und SQL. [80]

Aptana Studio 3 großer Vorteil im Gegensatz zu den bereits erläuterten IDEs ist die Tatsache, dass es sich dabei um eine Open Source Software handelt und sie aus diesem Grund nicht kostenpflichtig ist. Die individualisierbare Oberfläche ist eine weitere eklatante Unterscheidung zur Konkurrenz von Microsoft und Jetbrains. [81]

---

[78] Vgl. online [10BestIdeSoftware-2021], “The 10 Best IDE for Web Development”

[79] Vgl. online [10BestIdeSoftware-2021], “Visual Studio”

[80] Vgl. online [10BestIdeSoftware-2021], “IntelliJ IDEA”

[81] Vgl. online [10BestIdeSoftware-2021], “Aptana Studio 3”

3.2.3.2 Source Code Verwaltung

In den 1980er Jahren kamen SCMs (Source Code Management Systeme), wie das Revision-Control-System erstmals auf. Dies hat noch auf Basis einzelner Dateien gearbeitet und war dadurch nicht für verteilte Teamarbeit geeignet. Daher entwickelten sich später serverbasierte SCMs, wie Concurrent-Version-System und Subversion, bei denen ein Server den Programmcode zentral verwaltet und die Entwickler ihre lokale Arbeit immer wieder abgleichen. Stand der Technik sind heutzutage jedoch verteilte Systeme für die Versionskontrolle, wie GIT und Mercurial. Diese Systeme lösen aufgrund der besonders in großen Teams entscheidenden Effizienzvorteile die zentral verwalteten SCMs mehr und mehr ab. [82]

Weiters bieten verteilte Systeme das Werkzeug des Code-Reviews an. Dabei handelt es sich um eine Überprüfung, die bei der Zusammenführung einer Code-Änderung zur Code-Basis durchgeführt wird. Dies wird mittels Anzeige der veränderten Stellen einer Software ermöglicht. Dieser Schritt soll bewirken, dass Fehler möglichst früh entdeckt und behoben werden, dass der neue Code den Richtlinien entspricht, dass möglichst effizient entwickelt wird und dass die Wissensverteilung innerhalb des Teams vorangetrieben wird. Junge Entwickler können in der Regel durch Code-Reviews sehr viel von ihren erfahrenen Kollegen lernen. [83]

3.2.3.3 Qualitätskriterien für den Code

Eines der meistverwendeten Werkzeuge um die Code Qualität zu fördern in der Entwicklung von Webapplikationen, ist ESLint. Es handelt sich dabei um ein statisches Analysetool, welches Fehler, wie zum Beispiel nie erreichbare Source-Code Blöcke, unnötige Konsolenausgaben oder auch nicht einheitliche Formatierungen aufzeigt. Dieses Tool prüft die Programmzeilen nach definierten Regeln, welche bei Bedarf angepasst werden können. Hilfsmittel, welche die Programmzeilen automatisch formatieren, wie Prettier und Standard JS sind ebenfalls auf ESLint aufgebaut. [84]

---

[82] Vgl. [BestPracticeSoftwareEng-2010], Seiten 381-382

[83] Vgl. online [12BestCodeReviewTools-2020]

[84] Vgl. online [StaticAnalysisJavaScript-2020], „ESLint“

Bei SonarCloud handelt es sich um eine umfassende Werkzeugbox für die statische Analyse der Qualität einer Software. Dieses Tool bewertet anhand von Messgrößen wie Zuverlässigkeit, Sicherheit, Wartbarkeit, Unit-Test Abdeckung und Programmcode-Wiederholungen die Qualität eines Source Codes. Diese Maßeinheiten können je Projekt, je Modul oder je Dateiebene ausgewertet werden. In der folgenden Abbildung wird eine Auswertung der SonarCloud dargestellt. [85]

Abbildung 3: Auswertung der SonarCloud, [StaticAnalysisJavaScript-2020]

[85] Vgl. online [StaticAnalysisJavaScript-2020], „SonarCloud“

Außerdem wird für die Verbesserung der Code-Qualität heutzutage auch ein Code-Review durchgeführt. Diese Tätigkeit wurde bereits im Kapitel 3.2.3.2 Source Code Verwaltung erklärt.

3.2.3.4 Kompilierungs- und Bereitstellungsprozess

Ein bekanntes Open-Source Werkzeug, auf dem der Kompilierungs- und Bereitstellungsprozess stattfindet, ist der Jenkins. Dessen Vorteile sind die einfache Installation auf unterschiedlichen Betriebssystemen, verständliche Bedienbarkeit, hoher Funktionsumfang (Benachrichtigungen, Zeitpläne, …) erweiterbar durch eine Menge von Plugins, welche von einer großen aktiven Community bereitgestellt und gewartet werden. [86]

Bamboo ist im Gegensatz dazu ein Tool welches sich mehr um das gesamte „Release Management“ kümmert. Dieses kostenpflichtige Werkzeug unterstützt den fachlichen Entwicklungsprozess und ist durch ihre Funktionalität wie Testunterstützung sowie die Aktivierung unterschiedlicher Softwareversionen auch von nicht Technikern bedienbar. [87]

TeamCity ist ein Hilfsmittel von der Firma JetBrains, welches sich als Integrationswerkzeug zwischen den anderen Tools positioniert. Es bietet eine Integration mit verschiedenen IDEs, der Projektmanagementsoftware JIRA, der Kommunikationsanwendung Slack und auch der Azure Cloud an. Es hebt sich nicht als einzelnes Werkzeug hervor, welches in der modernen agilen Softwareentwicklungswelt ein eklatanter Vorteil ist, da sowieso schon sehr viele Hilfsmittel verwendet werden. TeamCity kann als freies oder auch kostenpflichtiges Produkt genutzt werden. [88]

[86] Vgl. online [Best-CI/CD-Tools-2020], „Jenkins”

[87] Vgl. online [Best-CI/CD-Tools-2020], „Bamboo”

[88] Vgl. online [Best-CI/CD-Tools-2020], „TeamCity”

3.2.3.5 Tests

Das bekannteste Open-Source Testwerkzeug für automatisierte Oberflächenfunktionstests bei Webapplikationen ist Selenium. Es ist auf den gängigen Betriebssystemen (Windows, Linux, Mac) und Browsern (Chrome, Internet Explorer, Firefox, Headless Browser) lauffähig. Bei Selenium handelt es sich nicht um ein einzelnes Werkzeug, sondern um eine Kollektion mehrerer Testhilfsmittel. Die Selenium IDE ist die Entwicklungsumgebung, Selenium RC die „Fernbedienung“, Selenium WebDriver die Schnittstelle zum Browser und Selenium Grid die Umgebung in der Selenium Tests parallel ausgeführt werden können. Die Selenium Tests werden als Code geschrieben und sind daher nur durch Softwareentwickler erstellbar. [89]

In der folgenden Abbildung ist die Selenium-IDE dargestellt. Mit diesem Werkzeug können Testfälle erstellt und ausgeführt werden. [90]

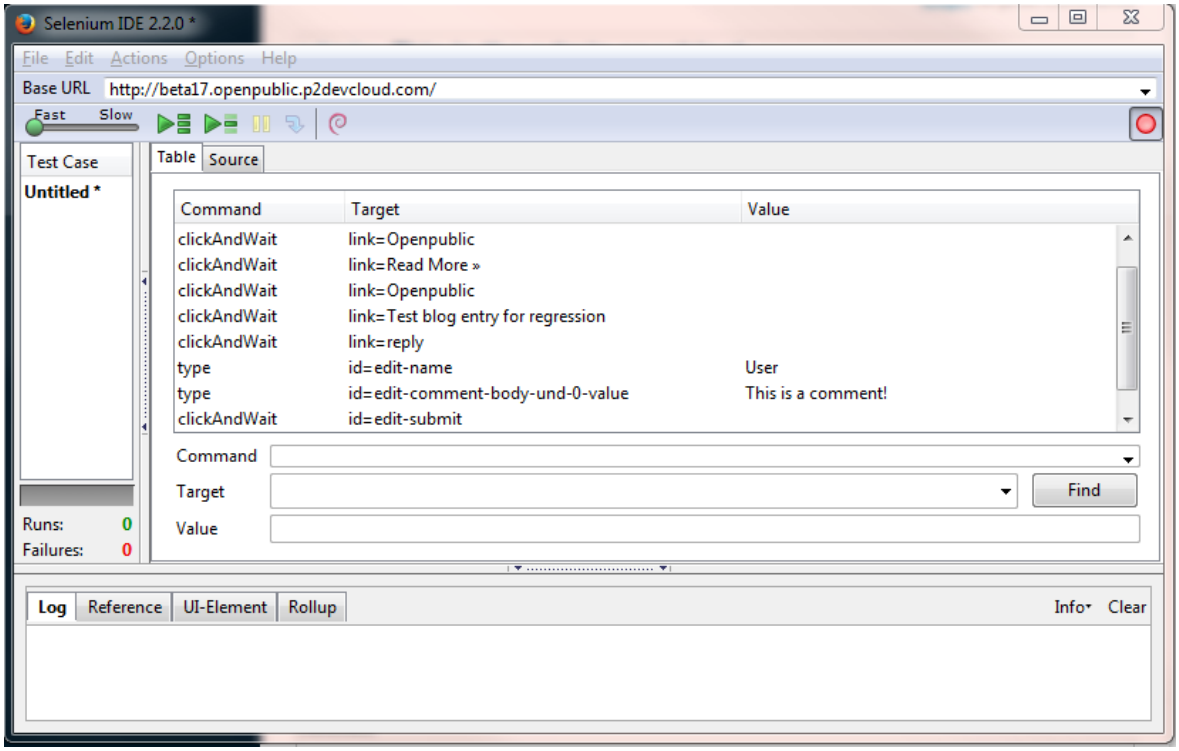


Abbildung 4: Selenium IDE, [Selenium-IDE-2013]

[89] Vgl. online [BestTestingToolsWeb-2021], „Selenium”

[90] Vgl. online [Selenium-IDE-2013], „Selenium IDE: Automated Testing Made Easy“

SoapUI ist ein Testwerkzeug, welches auf der Integrationstestebene angewandt wird. Es kann APIs verschiedener Technologien, wie zum Beispiel REST, SOAP und JMS durchtesten. Es bietet eine benutzerfreundliche Oberfläche, mit welcher die Tests einfach erstellt und verwaltet werden können. [91]

Um die Kompatibilität von einer Webapplikation auf die vielen am Markt verfügbaren Browsern zu prüfen, gibt es Werkzeuge wie zum Beispiel BrowserStack. Dieses Tool muss nicht installiert, da es in der Cloud läuft, und unterstützt circa 2000 verschiedene Browser auf den gängigen Betriebssystemen iOS, Android, Opera Mobile, Windows und Mac. Weiters ermöglicht es auf verschiedenen Bildschirmgrößen und Auflösungen zu testen. BrowserStack bietet eine Integration mit Selenium an. [92]

Lasttests bei Webapplikationen werden zumeist mit dem Werkzeug LoadRunner durchgeführt. Es bietet auf der einen Seite die Funktionalität, die gewünschte Last an Benutzeranfragen an das System zu schicken, und auf der anderen Seite einen Bericht von dessen Umgang mit der Last zu erstellen. Durch dieses Hilfsmittel ist ermittelbar, welche Teile der Software, mit Ressourcen Allokation oder Codeänderung, angepasst werden sollen, um der erwarteten Last Stand zu halten. LoadRunner kann mit allen State of the art Technologien umgehen. [93]

Bei Netsparker handelt es sich um ein Tool um die Sicherheitsrisiken einer Webapplikation möglichst früh zu identifizieren. Es testet eine Applikation automatisiert auf Schwachstellen, wie SQL-Injections, Cross-Site-Scripting (XSS) und Not-Found Fehlern (HTTP-404) durch. [94]

3.2.3.6 Betriebsüberwachung

Bei einer Webapplikation liegt das Hauptaugenmerk bei der Überwachung des Betriebs auf der generellen Verfügbarkeit der Webapplikation, die Fehlerrate der APIs

[91] Vgl. online [BestTestingToolsWeb-2021], „SoapUI"

[92] Vgl. online [BestTestingToolsWeb-2021], „BrowserStack"

[93] Vgl. online [BestTestingToolsWeb-2021], „LoadRunner"

[94] Vgl. online [BestTestingToolsWeb-2021], „Netsparker"

(Schnittstellen) und die Performance der Applikation. Die Verfügbarkeit wird normalerweise durch Werkzeuge präventiv, die Fehlerrate der Services und die Performance durch Protokollierung/Überwachung des laufenden Systems ermittelt. [95]

Es gibt jede Menge Werkzeugkisten für die Betriebsüberwachung am Markt. Neben den schon im Absatz davor genannten Fokusse auf die Verfügbarkeitsüberwachung, Fehlerrate und Leistungsmessung gibt es zusätzlich Angebote für die Überwachung unterschiedlichster Funktionalitäten, zum Beispiel die Warnung für den Ablauf von Zertifikaten, E-Mail-Benachrichtigungen, jede Menge Logs und Protokolle bis hin zum Ressourcen Management einer Cloud. Sematext, Pingdom und Site 24x7 sind nur ein paar bekannte Tools. Sie alle vereint, dass ihr voller Funktionsumfang kostenpflichtig ist. Die einzelnen Werkzeuge unterscheiden sich leicht voneinander und dadurch ist die Entscheidung, welches Tool eingesetzt wird, von der jeweiligen Webapplikation abhängig. [96]

[95] Vgl. online [WebsiteMonitoringTools-2021], „Main Types of Website Monitoring Tools”

[96] Vgl. online [WebsiteMonitoringTools-2021], „Top 12 Website Monitoring”

4 Untersuchung der Möglichkeiten

In diesem Kapitel werden die niedergeschriebenen Daten und Fakten untersucht, welche Werkzeuge und Prozesse für die Softwareentwicklung die größte Bedeutung haben.

4.1 Ausgestaltung des Messinstruments

4.1.1 Messinstrument

Die empirische Sozialforschung beschäftigt sich mit der methodischen Erhebung von Daten über soziale Sachverhalte, um Erkenntnisse in der Sozialwissenschaft zu erlangen. Sie bildet neben der speziellen Soziologie und der allgemeinen Soziologie, die dritte Säule in der Soziologie. [97]

4.1.1.1 Gütekriterien

Bei einem Messinstrument handelt es sich in der empirischen Sozialforschung um die Methode zur Erhebung von Daten. Ein Messinstrument muss folgende Merkmale (Gütekriterien) erfüllen: [98]

- interne Validität – Die Ergebnisse und deren Interpretation der Datenerhebung sind glaubwürdig. [99]
- externe Validität – Das Fazit aus der Datenerhebung ist auf verwandte Themen übertragbar. [100]
- Reliabilität – Die Datenerhebung ist nachvollziehbar gestaltet und eine Reproduktion führt zum gleichen Ergebnis. [101]

---

[97] Vgl. [ForschungsmethodenUndEvaluation-2016], Seiten 4-9

[98] Vgl. [ForschungsmethodenUndEvaluation-2016], Seite 42

[99] Vgl. [ForschungsmethodenUndEvaluation-2016], Seite 109

[100] Vgl. [ForschungsmethodenUndEvaluation-2016], Seite 109

[101] Vgl. [ForschungsmethodenUndEvaluation-2016], Seite 109

- Objektivität – Interessen, Perspektiven oder Vorurteile beeinflussen nicht die Datenerhebung. [102]

4.1.1.2 Forschungsansatz

Grundsätzlich wird beim Forschungsprozess in der empirischen Sozialforschung zwischen den Forschungsarten quantitativ, qualitativ sowie einer Mischung von beiden unterschieden. [103]

Beim quantitativen Verfahren wird überwiegend mit großen Mengen an standardisierten Daten, wie zum Beispiel mit vorgegebenen Antwortmöglichkeiten bei Fragebögen, gearbeitet. Aufgrund der Eindeutigkeit der Daten können diese problemlos mittels statistischer Methoden ausgewertet werden. [104]

Das qualitative Verfahren hingegen hantiert mit kleinen Mengen an nicht standardisierten Daten. Die automatische Datenanalyse ist aufgrund der inhomogenen Daten nicht möglich. [105]

Beim sogenannten „Mixed-Method-Verfahren“ handelt es sich um eine Mischung aus beiden gerade beschriebenen Ansätzen. Meistens wird dabei der Forschungsprozess zweimal durchlaufen, jeweils qualitativ und quantitativ. Daher ist dieses Verfahren aufwendiger. Der große Vorteil bei diesem Verfahren ist die Steigerung des Gütekriteriums der Validität. [106]

---

[102] Vgl. [ForschungsmethodenUndEvaluation-2016], Seite 110

[103] Vgl. [ForschungsmethodenUndEvaluation-2016], Seiten 22-23

[104] Vgl. [ForschungsmethodenUndEvaluation-2016], Seiten 23-25

[105] Vgl. [ForschungsmethodenUndEvaluation-2016], Seiten 25-26

[106] Vgl. [ForschungsmethodenUndEvaluation-2016], Seiten 26-28

4.1.2 Datenerhebung

4.1.2.1 Arten der Datenerhebung

Um die in der Theorie erläuterten Möglichkeiten im technischen Softwareentwicklungsprozess auf deren Wirtschaftlichkeit zu beurteilen wurden die folgenden drei Datenerhebungsmethoden genauer untersucht.

Eine Methode der wissenschaftlichen Datenerhebung ist die Beobachtung. Bei diesem Instrument wird menschliches Handeln, sowie dessen Zusammenhänge analysiert. Diese Kategorie der Datenerhebung ist den qualitativen Forschungsarten zuzuordnen. Es wird oft eingesetzt, wenn nur wenig über das Forschungsthema bekannt ist oder sich die Teilnehmer anders einschätzen als sie sich tatsächlich verhalten. Nachteile der Beobachtung sind einerseits objektiv zu bleiben und andererseits eine aussagekräftige Auswertung über die nur schwer vergleichbaren Daten zu erreichen. [107]

Das Interview ist eine wissenschaftliche Methode, die durchgeführt wird, um persönliche Ansichten für die Beantwortung der Forschungsfrage zu ermitteln. Es handelt sich ebenfalls um eine qualitative Forschungsart. Die Befragung kann strukturiert, halb strukturiert oder nicht strukturiert aufgebaut sein. Dadurch kann die Standardisierung der Daten und somit dessen Auswertung beeinflusst werden. Ein Interview wird angewandt um neben den Antworten auf die jeweiligen Fragen, ebenfalls Auskunft über die Haltung und Emotion der Befragten zu dem Thema erhalten. [108]

Eine weitere Methode der wissenschaftlichen Datenerhebung ist der Fragebogen. Eine Umfrage kann als qualitative Forschungsart mit offenen Antwortmöglichkeiten und weniger Teilnehmern aber auch als quantitative Art, standardisierte Antwortmöglichkeiten und viele Teilnehmer eingesetzt werden. Diese Methode kann gut angewandt werden, um Forschungsfragen mit praxisrelevanten Erfahrungen zu beantworten. [109]

Die Forschungsfrage erfordert eine allgemeine Aussage zum Thema Einsatz geeigneter Werkzeuge im technischen Softwareentwicklungsprozess um die Entwicklung einer Webapplikation mit einem skalierbaren Team möglichst wirtschaftlich zu bewerkstelligen.

---

[107] Vgl. [ForschungsmethodenUndEvaluation-2016], Seiten 323-327

[108] Vgl. [ForschungsmethodenUndEvaluation-2016], Seiten 356-360

[109] Vgl. [ForschungsmethodenUndEvaluation-2016], Seiten 398-408

Damit soll ein grober Leitfaden entstehen, der nicht auf Detailfragen zu bestimmten Bedingungen eingeht. Um möglichst viele Meinungen einzuholen, wurde die quantitative Variante des Fragebogens als richtiges Mittel gewählt.

4.1.2.2 Theoretische Grundlagen des Fragebogens

Bei der Erstellung eines Fragebogens muss zunächst der Zweck der Befragung, die erwarteten Ergebnisse und auch die Zielgruppe festgelegt werden. Der Inhalt einer Befragung ist vom Ziel zur Beantwortung der Forschungsfrage abhängig. Sollte der Fragenbogenersteller selbst nicht zur Zielgruppe des Fragebogens gehören, ist es sinnvoll, dass er sich Unterstützung von Experten holt. Zum Beispiel kann mittels eines Experteninterviews das fachliche Wissen erworben werden, um einen passenden Fragebogen auszuarbeiten. [110]

Um sicherzustellen, dass die im Kapitel 4.1.1.1 erläuterten Gütekriterien, durch den Fragebogen erfüllt sind, sollten die folgenden Punkte alle mit Ja beantwortet werden: [111]

1. Erfasst der Fragebogen wirklich die angegebene Messung? [112]
2. Ist die Messung genau genug, dass eine Wiederholung des Fragebogens nahezu gleiche Ergebnisse liefern würde? [113]
3. Wird der Fragebogen von den Befragten unabhängig vom Fragebogenersteller ausgefüllt? [114]
4. Sind die ermittelten Ergebnisse eines Fragebogens generalisierbar? [115]
5. Ist ein Nutzen durch die Befragung für die Zielgruppe vorhanden? [116]

---

[110] Vgl. [Fragebögen-2016], Seiten 5-6

[111] Vgl. [Fragebögen-2016], Seiten 6-7

[112] Vgl. [Fragebögen-2016], Seite 7

[113] Vgl. [Fragebögen-2016], Seite 7

[114] Vgl. [Fragebögen-2016], Seite 7

[115] Vgl. [Fragebögen-2016], Seite 7

[116] Vgl. [Fragebögen-2016], Seite 7

6. Ist der Umfang des Fragebogens groß genug, um die relevanten Daten zu ermitteln, andererseits jedoch so minimal wie möglich, um die Befragten nicht unnötig zu fordern. [117]
7. Ist die Beantwortung des Fragebogens für alle Befragten möglich? [118]

Um der befragten Person den Nutzen, die Ziele und die Datenverwendung des Fragebogens zu vermitteln, sollte die Umfrage mit einem Begleitschreiben überliefert werden. Bei persönlicher Übermittlung kann anstatt eines Schreibens, diese Mitteilung natürlich auch mündlich kommuniziert werden. [119]

Das Layout eines Fragebogens sollte möglichst einheitlich sein und den Fokus auf das Wesentliche legen, um die Motivation bei den Befragten zu steigern und die Fehlerrate zu minimieren. Weiters ist es wichtig, die Umfrage als wertvoll zu präsentieren, sowie deren Länge aus Sicht des Befragten akzeptabel zu halten. Wie so oft ist auch bei einem Fragebogen der erste Eindruck von immenser Bedeutung. Neben dem schon erwähnten Begleitschreiben hat den größten Anteil daran das Titelblatt der Umfrage. Der Titel sollte möglichst kurz, verständlich und motivierend sein. Ein weiterer wichtiger Punkt ist die Erklärung des Fragebogens. Aus Sicht des Befragten muss alles Notwendige für das Ausfüllen erklärt werden, um Missverständnisse und somit falschen Antworten vorzubeugen. [120]

Bei den Fragen ist zwischen folgenden drei verschiedenen Typen zu unterscheiden: [121]

- Offene – Es gibt keine vorgegebenen Antwortkategorien, dadurch ist der Befragte sehr frei bei seiner Antwort. Es können Aspekte erläutert werden, welche zum Zeitpunkt der Konstruktion des Fragebogens noch nicht bekannt waren. [83]
- Geschlossene – Dabei handelt es sich um die klassischen Ankreuzfragen, Ja/Nein oder Skalenauswahl. Der Vorteil ist klar die einfache statistische Auswertung dieser Fragentypen. [83]

---

[117] Vgl. [Fragebögen-2016], Seite 7

[118] Vgl. [Fragebögen-2016], Seite 7

[119] Vgl. [Fragebögen-2016], Seiten 7-8

[120] Vgl. [Fragebögen-2016], Seiten 8-10

[121] Vgl. [Fragebögen-2016], Seite 12

- Halboffene – Dies ist eine Mischung aus den offenen und geschlossenen Fragentypen. Die klassische halboffene Frage ist eine geschlossene Auswahlfrage, mit einer Restkategorie (zum Beispiel „Sonstiges“), welche mit einem Freitext ausgefüllt werden kann. [83]

Die Sprache muss so einfach wie möglich gewählt werden. Wenn die Verwendung von mehrdeutigen Worten nicht umgangen werden kann, sollte deren Bedeutung genauer erläutert werden. Nebensätze und versteckte Informationen gilt es genauso wie doppelte Verneinungen zu vermeiden. Die Antwortkategorien sollten innerhalb eines Fragebogens homogen sein. [122]

Bei Skalenpunkten gilt es die richtige Balance zwischen zu geringer und zu großer Breite zu finden. In der Regel sind zwischen fünf und sieben Skalenabstufungen angemessen, mehr führen zur Überforderung, weniger reduzieren die Reliabilität. Wenn nur die Pole der Skalenpunkte beschriftet werden, kreuzen die Befragten normalerweise intuitiver, mit ihrer eigenen Interpretation an. Die Skalenrichtung soll den Lesekonventionen folgen, daher in Mitteleuropa von links nach rechts. Das einfachste Skalenniveau ist die Nominalskala. Dabei unterscheiden sich die Antwortmöglichkeiten klar voneinander. Bei einer Ordinalskala hingegen stehen die Optionen in einer Beziehung und sind sortierbar. Die Antwortmöglichkeiten in der Intervallskala sind ebenfalls sortierbar und weisen sogar gleich große Abstände zwischen den Skalenpunkten auf. Dies ermöglicht eine sinnvolle Berechnung von Mittelwerten. Die Ratio- oder Verhältnisskala ist auf der Intervallskala aufgebaut und hat zusätzlich einen echten Skalenmittelpunkt. [123]

Zum wichtigen ersten Eindruck einer Umfrage zählt nicht nur das Titelblatt, sondern auch die ersten Fragen. Diese sollten für die Befragten sinnvoll, spannend und leicht zu beantworten sein. In der Regel gilt, dass Personen, welche die ersten paar Fragen ausfüllen, sich „verpflichtet“ fühlen, den gesamten Fragebogen zu beantworten. Mehrere Fragen zu den gleichen Themen sollten in Blöcke, jeweils mit Überschriften, zusammengefasst werden, um schlagartige Änderungen innerhalb des Fragebogens zu vermeiden. Spezifische Informationen über die befragte Person sollten lieber am Ende

[122] Vgl. [Fragebögen-2016], Seiten 11-14

[123] Vgl. [Fragebögen-2016], Seiten 14-20

einer Befragung sein, da sie am Beginn eher abschreckend wirken. Dies gilt auch für komplexe sowie heikle Fragen. [124]

Bei einer Umfrage sollte eine Testdurchführung mit einer kleinen Zielgruppe durchgeführt werden, um den Fragebogen auf deren Merkmale zu prüfen. Wenn bei der Probedurchführung Fehler identifiziert wurden, sollte nach der Ausbesserung die Testdurchführung erneut durchgeführt werden. [125]

Da es nahezu unmöglich ist, bei einem Fragebogen zu 100% Rücklauf zu erhalten, ist es notwendig eine Stichprobe aus der Zielpopulation zu befragen, um Schlussfolgerungen für die Grundgesamtheit zu ziehen. Ein Fehler in der Stichprobenauswahl kann mit hoher Wahrscheinlichkeit zu einer völlig verzerrten Auswertung der Umfrage führen. Daher sollte die Stichprobenmenge bei einem großen Umfang nach dem Zufallsprinzip, bei einem kleineren nach bestimmten Charakteristiken ausgewählt werden. [126]

Oft werden bei Umfragen alle Personen befragt, welche bereit dazu sind und zufällig dem Befragenden begegnen. Das Problem dabei ist, dass durch diese fehlende Auswahl keine Schlussfolgerungen auf die Zielgruppe gezogen werden können. Dies ist bei unsystematischen Internet- oder Zeitungsbefragungen häufig der Fall. [127]

4.1.2.3 Ausarbeitung des Fragebogens

Der Zweck des Fragebogens ist klar, er soll die Daten liefern um die Forschungsfrage, welche Werkzeuge beziehungsweise Hilfsmittel im technischen Softwareentwicklungsprozess eingesetzt werden sollen, um die wirtschaftliche Entwicklung einer Webapplikation durch ein skalierbares und flexibles Team zu ermöglichen, beantworten zu können.

Das erwartete Ergebnis ist einen Leitfaden bereitzustellen, der eine Aussage zulässt, wie wichtig der Einsatz beziehungsweise die Nutzung der jeweiligen Tools ist, um die

---

[124] Vgl. [Fragebögen-2016], Seiten 21-22

[125] Vgl. [Fragebögen-2016], Seiten 24

[126] Vgl. [Fragebögen-2016], Seiten 24-25

[127] Vgl. [Fragebögen-2016], Seite 25

wirtschaftliche Entwicklung einer Webapplikation durch ein skalierbares und flexibles Team zu ermöglichen.

Die Zielgruppe des Fragebogens umfasst alle Personen, welche Erfahrung mit der Ausübung einer Rolle des technischen Softwareentwicklungsprozesses haben. Da sich die Zielgruppe nicht nur auf Deutschsprachige beschränkt, wird der Fragebogen nur in englischer Sprache konzipiert, mit dem Titel, „Please help me with your opinion to evaluate the importance of tools within a technical software development process of an web-application, developed by an scalable, flexible team in terms of cost and benefit“. Die deutsche Übersetzung des Titels lautet, „Bitte helfen Sie mir mit Ihrer Meinung, die Bedeutung von Tools innerhalb des technischen Softwareentwicklungsprozesses einer Webanwendung entwickelt durch ein skalierbares, flexibles Team hinsichtlich Kosten und Nutzen zu bewerten“.

Als Medium für den Fragebogen wird das Internet gewählt, um Personen unabhängig von dessen Standort erreichen zu können. Die Bitte um die Beantwortung des Fragebogens wird mittels E-Mail-Versand und zusätzlich über die sozialen Netzwerke (WhatsApp, LinkedIn, …) an die Zielgruppe verbreitet. Diese Information wird mit folgendem Zusatztext übermittelt:

Hello,

I am currently writing my thesis about the use of tools in the software development process.

There, I would kindly ask you to provide me your opinion within this questionnaire (https://forms.gle/nNA1icvLHwFipGUt5) to evaluate the importance/usefulness of tools.

It would really help me a lot and it's really just a short selection form, without personal data.

Derzeit schreibe ich meine Diplomarbeit über den Einsatz von Werkzeugen im Softwareentwicklungsprozess.

Dabei benötige ich bitte Deine Meinung in diesem Fragebogen (https://forms.gle/nNA1icvLHwFipGUt5), um die Bedeutung/Nützlichkeit von Tools zu bewerten.

Es würde mir wirklich sehr helfen, es handelt sich nur um ein kurzes Auswahlformular, ohne personenbezogene Daten.

Thanks for your support!/Danke für Deine Unterstützung ☺

Best Regards/Liebe Grüße,
Philipp

Aufgrund der angenehmen statistischen Auswertung mit einem sehr kleinen Interpretationsspielraum, wird zum Großteil der geschlossene Fragentyp eingesetzt.

Als Inhaltsgrundlage für die Fragen dienen die im Kapitel 3.2.3 erläuterten Werkzeuge. Unter Berücksichtigung der im Kapitel 4.1.1 erläuterten Gütekriterien enthält die Umfrage folgende Fragen, welche mit der Verhältnisskala nicht wichtig (-2) bis sehr wichtig (+2), beantwortet werden sollen. Zusätzlich ist zu den englischsprachigen Fragen aus dem Fragebogen in dieser Diplomarbeit eine kurze Erklärung in deutscher Sprache beigefügt.

1. IDE – Integrated Development Environment; es handelt sich um die Kategorie über die Entwicklungsumgebung
    a. Native Support for Programming Language; Unterstützung für die Programmiersprache
    b. Debugger; Werkzeug für die Fehlersuche
    c. Built-in Version Control Management; Eingebaute Unterstützung für die Source Code Verwaltung
    d. Automatic Code Completion; Automatische Vervollständigung des Programmcodes
    e. Built-in Infrastructure Integration; Eingebaute Verwaltungsfunktionalität für die Infrastruktursteuerung
    f. Quick and Reliable Source Code Indexing; Schnelle und zuverlässige Quellcode Indizierung
    g. Out-of-the-box Experience (without configuration effort); Verhalten direkt nach der Installation, ohne zusätzlichem Konfigurationsaufwand
    h. Expandability by Plug-Ins; Erweiterbarkeit mittels Zusatzpakete

   i. Live Work Collaboration (pair programming support for several remote developers); Unterstützung gemeinsamer Programmierung mehrerer verbundener Entwickler innerhalb einer Entwicklungsumgebung
2. Source Code Management; Frage zu dem Bereich der Source Code Verwaltung
   a. General use of distributed source code management system in every project (eg. GIT); Allgemeiner Einsatz von verteilten Source Code Verwaltungssystemen, wie zum Beispiel GIT, in jedem Projekt
3. Code Quality; es handelt sich über die Kategorie der Code Qualität
   a. Linter for Static Code Analysis; Verwendung von statischen Sourcecode Analysetools
   b. Auto Formatting of the Code; Automatische Formatierung des Sourcecodes
   c. Regular extensive execution of Static Code Analysis (e.g SonarCloud at every build); Regelmäßige Durchführung Statischer Code Analysetools als Teil des Bereitstellungsprozesses
   d. Code Review - to improve Code Quality; Durchführung des Code-Reviews, um die Code Qualität zu verbessern
   e. Code Review – to improve Knowledge Distribution; Durchführung des Code-Reviews, um Wissen zu verteilen
4. Continuous-Integration Process Tool; Fragen über die Hilfsmittel für den Bereitstellungsprozess
   a. Easy installation and operation of the tool; Einfache Installation und unkomplizierter Betrieb des Werkzeuges
   b. Many features (e.g schedules, notifications, …); Viele Funktionalitäten wie Zeitplanung und Benachrichtigungen
   c. Expandability by Plug-Ins; Erweiterbarkeit mittels Zusatzpakete
   d. Usable by non-developers; Verwendbar auch von nicht Entwicklern
   e. Comfortable integration between the rest of the used tools (e.g IDE, project management software, communication tool and infrastructure); Komfortable Integration zwischen den weiteren eingesetzten Tools, wie der IDE, Projektmanagement-Software, Kommunikationstool und Infrastruktur
5. Testing; es handelt sich über die Kategorie des Testens
   a. Automated UI-Tests; Automatische Oberflächenfunktionstests
   b. Automated API Tests (Integration Tests); Automatische Tests der APIs (Backend Schnittstellen)

    c. Compatibility test (e.g different browsers/screen sizes); Kompatibilitätstests in verschiedenen Browsern und mit unterschiedlichen Auflösungen
    d. Load Test; Lasttest
    e. Penetration Test; Penetrationstest (Test auf Sicherheitslücken)
    f. Creation/execution and maintenance of the tests should be possible with as little technical knowledge as possible; Erstellung, Durchführung und Wartung der Tests sollte ohne umfangreiches technisches Wissen möglich sein
6. Operation and Maintenance; Fragen über die Bedeutung der Nutzung von Werkzeugen zur Unterstützung des Betriebs und der Wartung
    a. Monitoring the Availibility of the application (alerting in the event of failure); Überwachung der Verfügbarkeit einer Applikation und Benachrichtigung im Falle des Ausfalls
    b. Monitoring the Error Rate of the APIs; Überwachung der Fehlerrate der Backend Schnittstellen
    c. Monitoring the Performance of the application; Überwachung der Performanz einer Applikation
    d. Notifications (e.g expiring certificates); Benachrichtigung zum Beispiel im Falle eines auslaufenden Zertifikates
    e. Logging as detailed as possible; Protokollierung so genau wie möglich
    f. Logging as far back as possible; Protokollierung so weit wie möglich in die Vergangenheit
7. In General (all segments from IDE to Operation); Allgemeine Fragen über alle Kategorien
    a. Tool supported and maintained by a company; Werkzeug wird unterstützt und gewartet von einer Firma
    b. Tool is Open-Source; Tool wird von einer Open-Source Gemeinschaft unterstützt und gewartet

Neben den inhaltlichen Fragen wird am Ende des Bogens die Rolle und Erfahrung des Befragten abgefragt, um die Antworten nach diesen Kategorien gruppieren zu können. Es stehen folgende Rollen zur Auswahl:

- Project Manager/Scrum Master; Projektleiter/Scrum Master
- Requirements Engineer; Anforderungsanalyst
- Developer; Entwickler

- Tester
- Designer; Grafiker
- Operation Engineer; Infrastrukturspezialist
- Architect; Architekt

und folgende Möglichkeiten für die Bekanntgabe der Jahre an Erfahrung in der Softwareentwicklung:

- 1
- 2
- 3+
- 5+
- 8+
- 13+
- 20+
- 40+

Der komplette Fragebogen ist bei den Anlagen, als Teil 1, zu dieser Diplomarbeit beigefügt.

4.1.3 Datenauswertung

4.1.3.1 Theoretische Grundlagen der Datenauswertung

Bevor die Datenauswertung erfolgt, muss geklärt sein, inwiefern fehlende Antworten in der Auswertung berücksichtigt werden. Es gibt mehrere Möglichkeiten. Die Rückmeldungen mit fehlenden Antworten komplett aus der Auswertung auszunehmen. Nur die jeweiligen nicht vorhandenen Antworten bei der Auswertung spezifischer Fragen nicht zu berücksichtigen. Dadurch ergeben sich unterschiedliche Anzahlen an Antworten bei den verschiedenen Fragen. Oder fehlende Antworten standardisiert zu ersetzen. [128]

Um eine Befragung auszuwerten, sollten alle Antworten in einem Statistik- oder Tabellenkalkulationsprogramm erfasst sein. Aufgrund der Nachverfolgung der Antworten

[128] Vgl. [Fragebögen-2016], Seiten 29-30

und eventueller Korrektur, sollte jede Rückmeldung mit einem eindeutigen Merkmal versehen sein. Die einzelnen Antworten als Datenpunkte sind Messwerte. Durch mathematische Berechnungen werden diese in Messungen umgewandelt. Das übergeordnete Ziel in der statistischen Auswertung ist die Datenkomplexität zu reduzieren und mit so wenigen Parametern wie möglich aussagekräftige Ergebnisse für logische Interpretation zu erhalten. [129]

Als erster Schritt werden Daten in der Auswertung zumeist graphisch dargestellt, da dies einen guten ersten Eindruck über die Datenverteilung vermittelt. Vor der Darstellung muss aber zuerst die Entscheidung getroffen werden, was mit der Grafik überhaupt dargestellt werden soll. Um den Konsumenten einer Grafik nicht in die Irre zu führen, sollen diese nicht zu viele Details enthalten. Außerdem muss die optische Größe innerhalb der Veranschaulichung mit den Zahlenverhältnissen in der Datenverteilung übereinstimmen. Auf passende Beschriftungen und Quellenangaben, falls Fremddaten verwendet werden, darf ebenfalls nicht vergessen werden. [130]

Ein Strukturvergleich gibt Auskunft über den Anteil eines Bestandteils an der Gesamtverteilung. Dies wird mit einem Kreisdiagramm dargestellt. Um Daten, welche in einer räumlichen oder zeitlichen Folge zueinanderstehen, werden mittels Balken- oder Säulendiagramm dargestellt. Wichtig ist zu bedenken die beiden Skalen immer mit einem sinnvollen Wert beginnen zu lassen. Zeitliche Verläufe oder Vergleiche sind am besten mit einem Linien- bzw. Kurvendiagramm darzustellen. Keinesfalls dürfen die Achsen dabei gestaucht oder gestreckt werden, da dies zu einer optischen Täuschung und somit falschen Interpretation führt. Ein Punktdiagramm eignet sich für die Darstellung von großen Menge an Daten verschiedener Werte. [131]

4.1.3.2 Durchführung der Datenauswertung

Insgesamt besteht die Umfrage aus 36 Einfachauswahl Fragen. Diese sind im Kapitel 4.1.2.3 Ausarbeitung des Fragebogens aufgelistet. Bis auf zwei, der Job-Rolle und der Anzahl an Erfahrungsjahren, sind alle Fragen mit der Verhältnisskala nicht wichtig (-2) bis

---

[129] Vgl. [Fragebögen-2016], Seite 30

[130] Vgl. [Fragebögen-2016], Seiten 31-32

[131] Vgl. [Fragebögen-2016], Seiten 32-33

sehr wichtig (+2) zu beantworten. Keine einzige Frage ist verpflichtend auszufüllen. Insgesamt wurden in 14 Tagen 52 Rückmeldungen gesammelt. 1858 von insgesamt 1872 möglichen Antworten wurden gesammelt. Somit wurden 99,25% aller Fragen in den Rückmeldungen beantwortet.

Bei dieser Datenauswertung werden nur die jeweiligen Fragen, die nicht beantwortet wurden, nicht in die Auswertung einbezogen, die gesamte Rückmeldung wird trotzdem berücksichtigt. Die einzelnen Punkte waren nämlich optional zu beantworten, da davon auszugehen ist, dass nicht jede Person zu jedem Werkzeug in einer Phase des technischen Softwareentwicklungsprozess eine Meinung über dessen Wichtigkeit hat.

In den folgenden Grafiken wird das arithmetische Mittel über alle Rückmeldungen der Beantwortung der selben Frage dargestellt. Um dies verständlich darzustellen wurden die Diagramme nach Kategorien der Fragen gegliedert.

Der wirtschaftlichen Bedeutung wurden die Antwortkategorien 1 - 2 wichtig, 0 - 1 nice to have, wie -2 – 0 unwichtig zugeordnet.

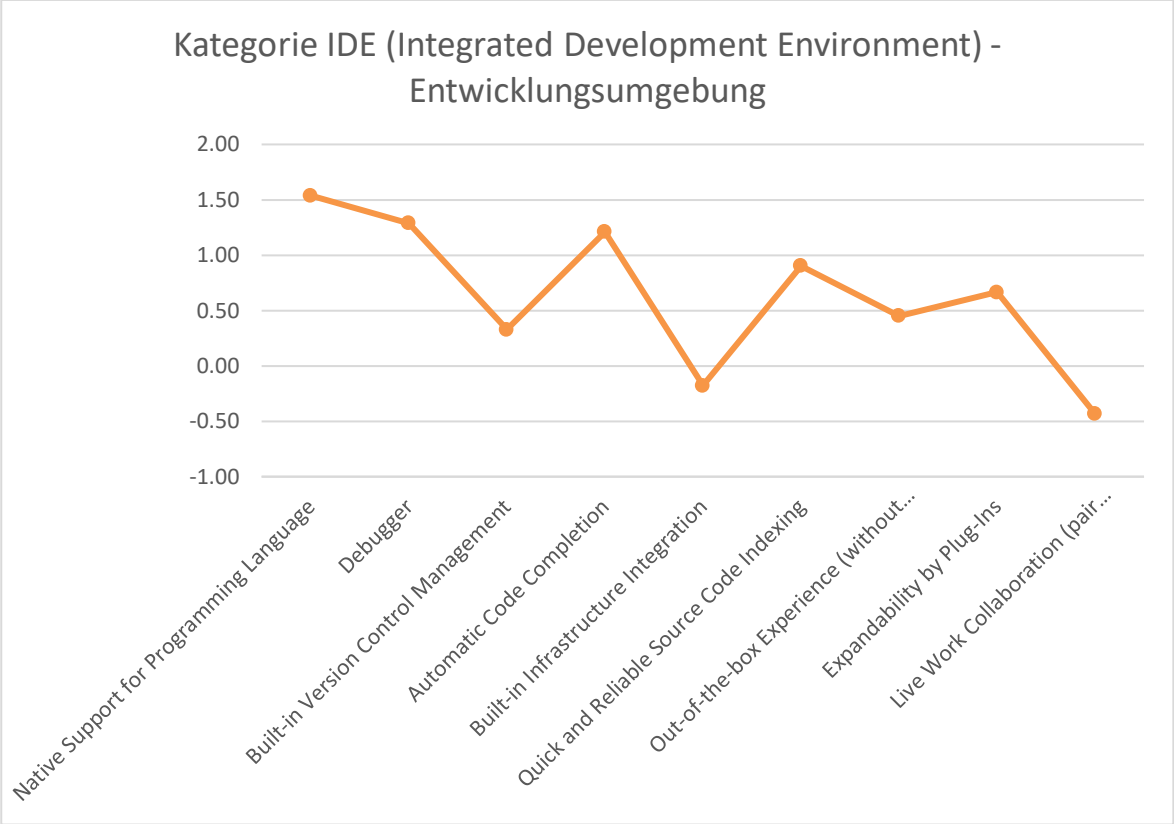


Abbildung 5: Arithmetisches Mittel aller Antworten der Kategorie IDE (Integrated Development Environment)

Das bedeutendste Hilfsmittel im Werkzeugkasten eines Entwicklers ist die Unterstützung für die Programmiersprache, wie zum Beispiel der Syntaxhervorhebung und Prüfung. Weiters ist sehr wichtig die Nutzung eines Debuggers für die Fehlersuche wie auch eine automatische Code Vervollständigung.

Die restlichen Werkzeuge, wie Source Code Verwaltung innerhalb der Entwicklungsumgebung, schnelle und zuverlässige Quellcode Indizierung, Verhalten der IDE direkt nach der Installation und auch die Erweiterbarkeit durch Zusatzpakete wurden alle in die Kategorie „Nice to have“ eingestuft.

Eine direkt in die Entwicklungsumgebung eingebettete Schnittstelle für die Steuerung der Software Infrastrukur hingegen, wurde als nicht wichtig bewertet. Als am wenigsten wichtig eingeschätzt wurde jenes Hilfsmittel, welches die gemeinsame Programmierung mehrerer Entwickler an verschiedenen Standorten unterstützt.

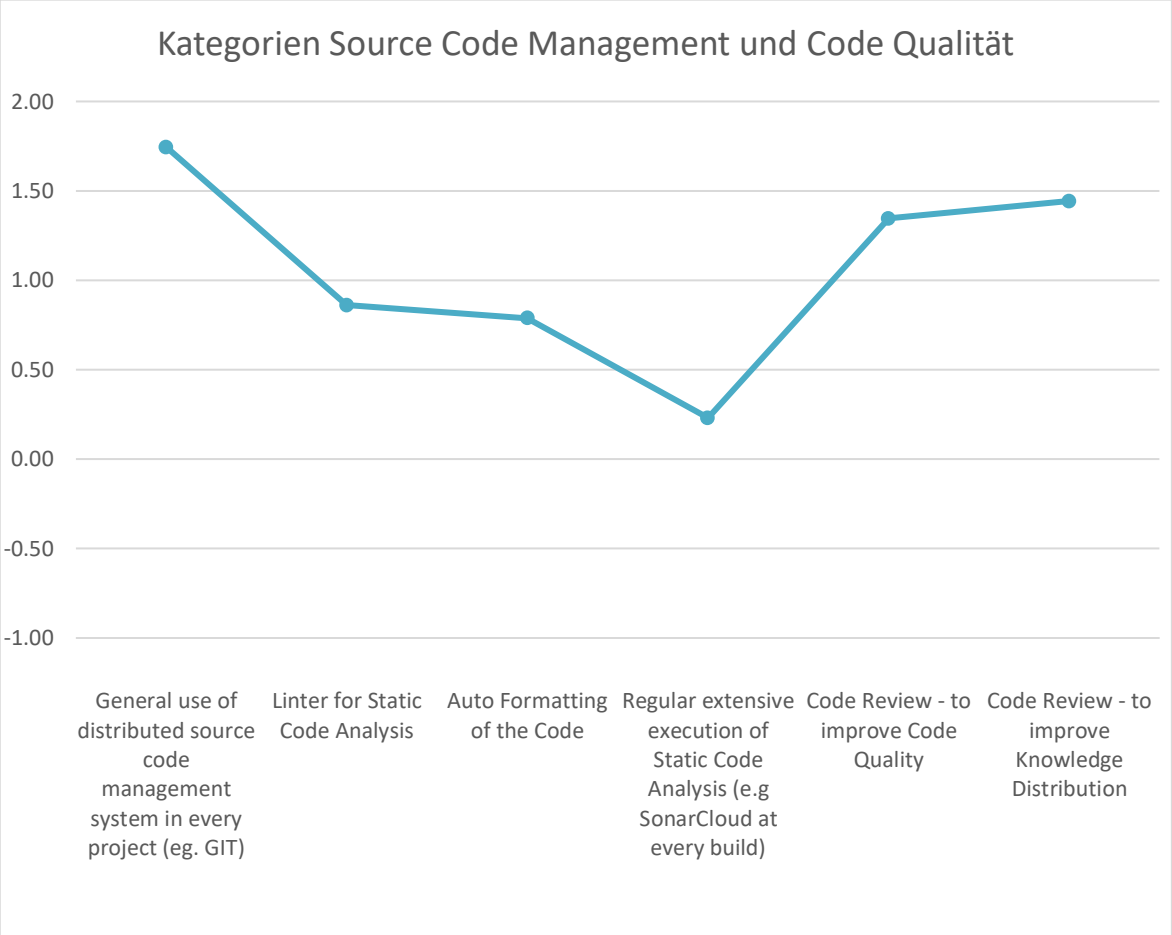


Abbildung 6: Arithmetisches Mittel aller Antworten der Kategorien Source Code Management und Code Qualität

Der Einsatz, beziehungsweise die Nutzung eines verteilten Source Code Verwaltungssystems wie GIT stellte sich als unumgänglich heraus, dies wurde durch den Fragebogen mit eindeutiger und klarer Zustimmung untermauert. Für die wirtschaftliche Entwicklung einer Webapplikation wird die Bedeutung des Code-Review für die Verbesserung der Code Qualität einerseits und auch um die Wissensverteilung innerhalb des Teams zu verbessern andererseits, positiv geschätzt.

Statische Sourcecode Analysetools, im Kleinen in Form von Lintern, als auch im Großen die regelmäßige Durchführung als Teil des Bereitstellungsprozesses, wurden genauso wie die automatischen Sourcecode Formatierung, im Durchschnitt als optional angesehen.

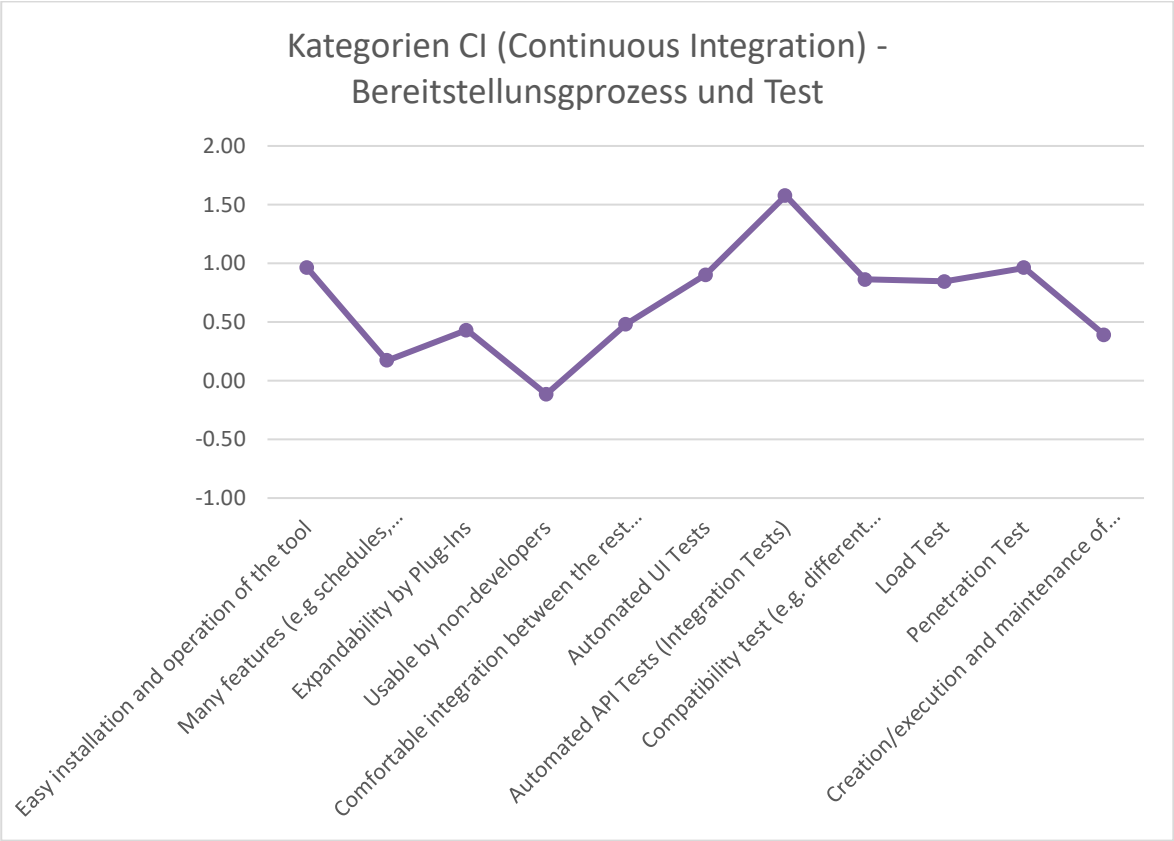


Abbildung 7: Arithmetisches Mittel aller Antworten der Kategorien Bereitstellungsprozess und Test

Das arithmetische Mittel aller Bewertungen in den Kategorien Bereitstellungsprozess und Testen bekräftigt klar die Wichtigkeit automatisierter Tests der Backend Schnittstellen (API Tests).

Bezüglich Continuous Integration, wurde die einfache Installation und der unkomplizierte Betrieb des Werkzeuges, Funktionen wie Zeitplanung und Benachrichtigung, die Erweiterbarkeit über Plug-Ins, wie auch die komfortable Integration innerhalb der restlichen genutzten Systeme als „Nice to have“ eigeordnet. Automatische Oberflächenfunktionstests, Kompatibilitätstests, Lasttests, Penetrationsstests und auch die Wartung und Durchführung der Tests ohne technisches Fachwissen fallen nach der statistischen Auswertung ebenfalls in die Kategorie „Nice to have“.

Dass die Hilfsmittel innerhalb des Bereitstellungsprozesses auch von nicht Technikern genutzt werden können, wurde insgesamt als nicht wichtig erachtet.

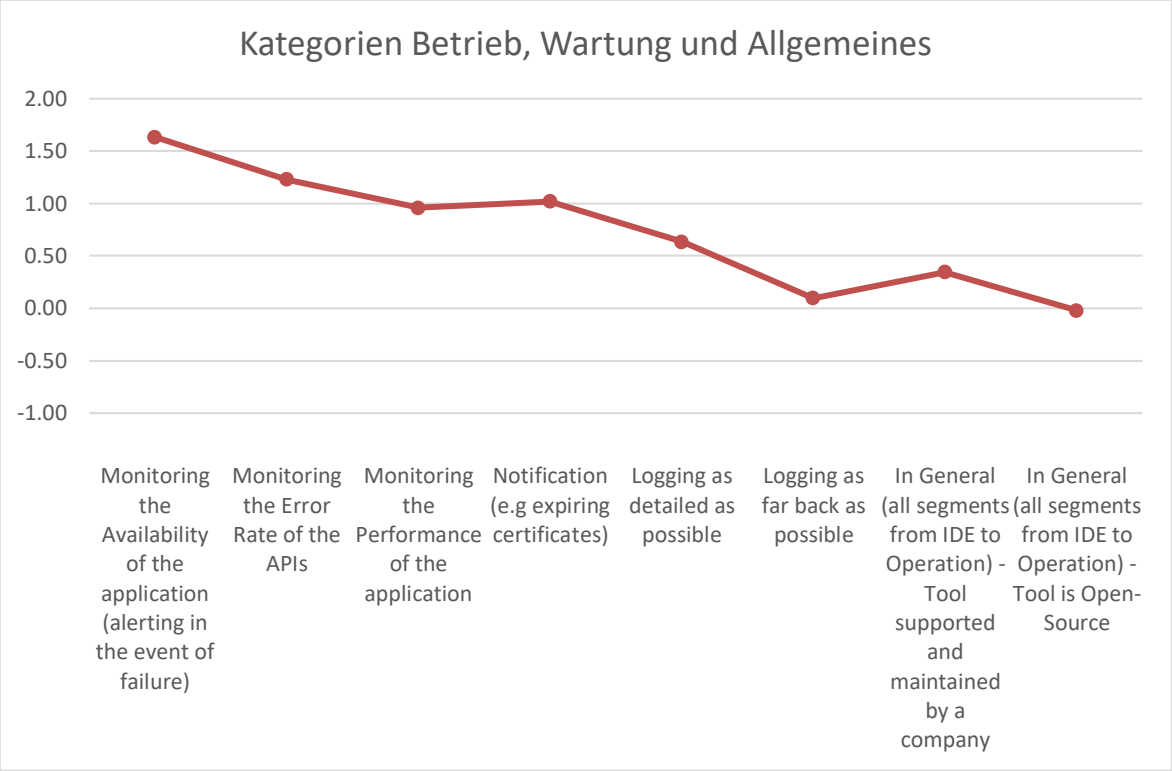


Abbildung 8: Arithmetisches Mittel aller Antworten der Kategorien Betrieb, Wartung und Allgemeines

Die Verfügbarkeit einer Webapplikation zu überwachen ist ebenso ein Muss. Genauso wichtig ist es die Fehlerrate der Backend Services im Auge zu behalten. Eine Benachrichtigung als Warnung im Falle von auslaufenden Zertifikaten zum Beispiel, wird ebenfalls als wichtig erachtet.

Monitoring der Performanz, detaillierte und auch weit historische Protokollierung werden als optionale Werkzeuge eingestuft.

Laut der Befragung werden allgemein Tools, welche von großen Firmen entwickelt, gewartet und unterstützt werden, im Gegensatz zu Open-Source Werkzeugen favorisiert.

Die Rollenverteilung bei den Rückmeldungen sieht wie folgt aus:

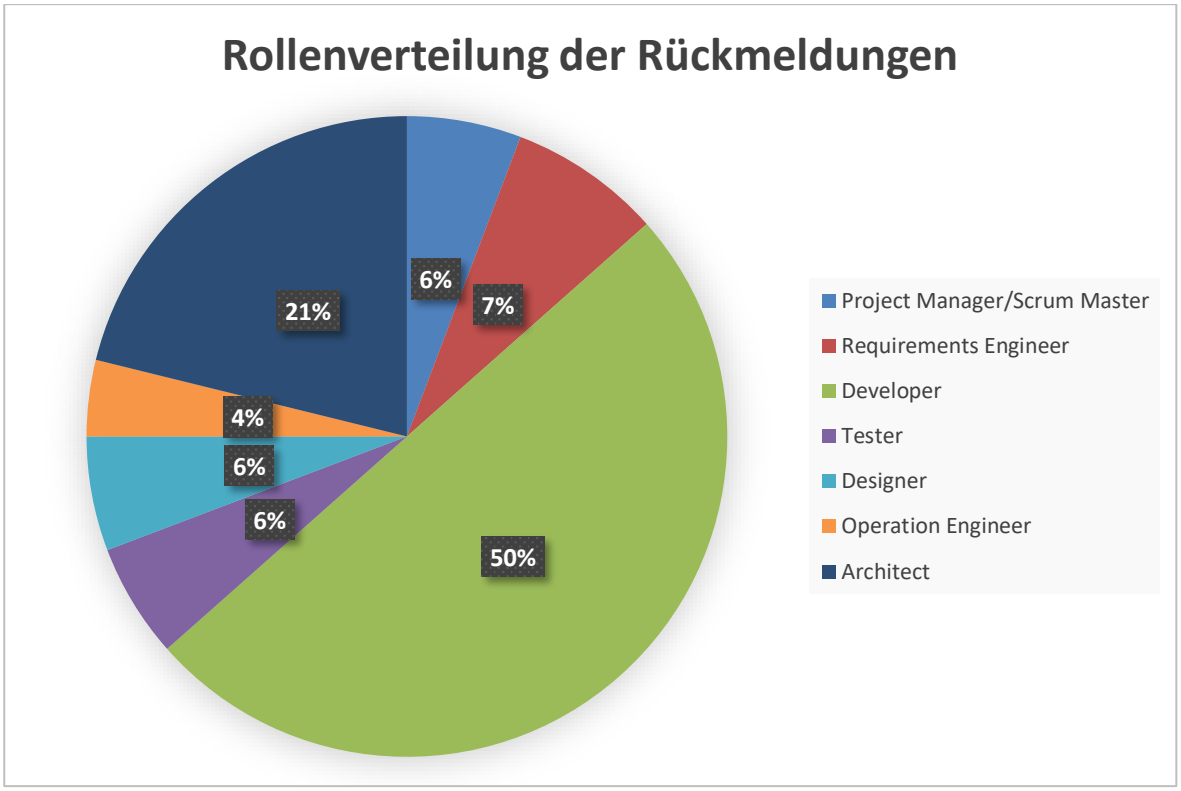


Abbildung 9: Anteil der Rückmeldungen der jeweiligen Rollen an den Gesamtrückmeldungen

Genau die Hälfte (26) aller Rückmeldungen stammen von Software-Entwicklern. Weitere 21%, es handelt sich um 11 Antworten des Fragebogens, stammen von Software-Architekten. Die restlichen 29% teilen sich zu ähnlichen Anteilen auf Analysten (4), Projektleiter oder Scrum-Master (3), Tester, (3), Grafiker (3) und Infrastrukturspezialisten (2) auf.

Da Entwickler und Architekten am meisten mit den Werkzeugen des technischen Softwareentwicklungsprozesses in der täglichen Arbeit zu tun haben, werden die Antworten der beiden Gruppen nachfolgend genauer ausgewertet und auf Unterschiede, im Gegensatz zu den allgemeinen Ergebnissen hervorgehoben.

Für Software-Entwickler ist in der Entwicklungsumgebung, eine schnelle und zuverlässige Quellcode Indizierung nicht nur „Nice-to-have“ sondern notwendig. Für die Verwendung

von Lintern um die Code-Qualität zu erhöhen, einfache Installation und Wartung des Continuous Integration Tools, automatische Oberflächenfunktionstests, Penetrationstests und Überwachung der Performance, zählen ebenfalls zu den Tools, welche für Entwickler notwendig erscheinen.

Ob ein Werkzeug von einer Firma oder einer Community unterstützt wird, ist für Entwickler nicht so relevant wie für den Rest der Befragten.

Die einzelne Betrachtung der Meinungen aus der Gruppe der Architekten hat hingegen nahezu idente Ergebnisse wie die Gesamtauswertung der Befragung. Der einzige Unterschied in der Gesamtmeinung ist über die Erweiterbarkeit der Entwicklungsumgebung mittels Zusatzpakete. Für Architekten ist dies ein Muss.

Die Verteilung der Erfahrung bei den Befragten ist wie folgt:

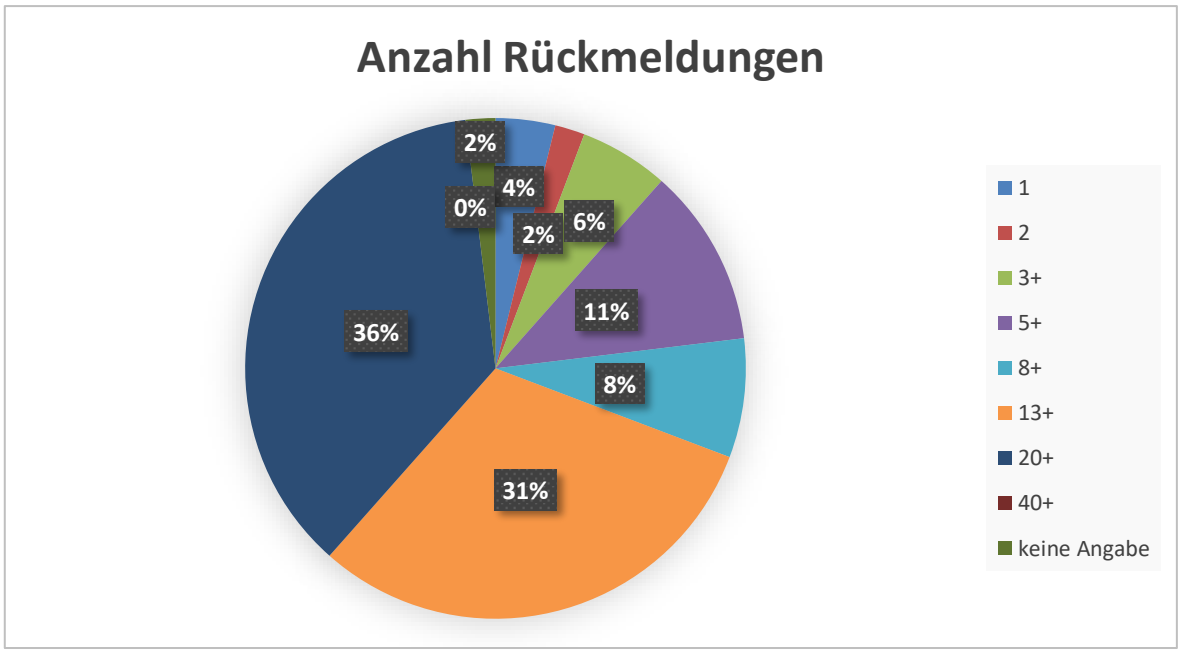


Abbildung 10: Verteilung der Erfahrungsjahre der einzelnen Befragten in den Gesamtrückmeldungen

Der Großteil (67%) aller Rückmeldungen stammt von Personen mit mehr als 13 Jahren (16 Fragebögen) oder mehr als 20 Jahren (19 Fragebögen) Erfahrung in der Softwareentwicklung. Leider konnte keine Person mit mehr als 40 Jahren Erfahrung in der Softwareentwicklung für den Fragebogen gewonnen werden, diese sind aber aufgrund der jungen Branche sehr rar. Die restlichen 17 Rückmeldungen teilen sich demnach auf

Personen zwischen einem und zwölf Jahren an Erfahrung, sowie einer nicht erhaltenen Auskunft auf.

Diese beiden Gruppen werden nachfolgend ausgewertet, um den Unterschied der Meinungen zwischen den „Jungen“ (Personen zwischen einem und zwölf Jahre an Erfahrung) und den „Alten“ (Personen ab 13 Jahre an Erfahrung in der Branche) zu analysieren.

Für die befragten Einsteiger in der Softwareentwicklung ist im Unterschied zum Gesamtergebnis, die Verwendung von einfachen Lintern, unkomplizierte Inbetriebnahme des Werkzeuges für den Bereitstellungsprozess, die Durchführung von Penetrationstests und die Überwachung der Performance wichtig, um gute Entwicklung zu gewährleisten.

Die Verwendbarkeit des Tools für den Continuous Integration Prozess auch von nicht Technikern, die Benachrichtigungsfunktionalität von auslaufenden Zertifikaten und auch die Tatsache, ob ein Werkzeug von einer „Open-Source“ Community entwickelt und gewartet wird, ist nach Meinung der „Jungen“ optional.

Für erfahrene Personen in der Softwareentwicklung gibt es zum Gesamtergebnis nur einen Unterschied. Ihrer Meinung nach ist die Protokollierung lange in die Vergangenheit nicht notwendig.

Ein sehr interessanter Fakt ist, dass die einzelne Auswertung der Gruppe der Architekten und auch der Erfahrenen, mit der Gesamtauswertung nahezu übereinstimmt. Dies stärkt die in der Softwareentwicklung gelebte These, dass im Zweifelsfall, wenn innerhalb des Teams keine gemeinsame Entscheidung getroffen werden kann, die wichtige und verantwortungsvolle Meinung von Architekten und routinierten Arbeitskräften für die weitere Vorgangsweise ausschlaggebend ist.

4.1.4 Schlussfolgerung/Empfehlung

Es sollte die Entwicklung einer Webapplikation mit einem ändernden Team niemals ohne passender Entwicklungsumgebung mit Unterstützung für die Programmiersprache, Debugger und Code Vervollständigung durchgeführt werden. Ein verteiltes Source Code Verwaltungssystem, wie GIT, muss genauso wie die strenge Einhaltung des Code-Review Prozesses, um die Qualität zu verbessern und das Wissen innerhalb des Teams zu verteilen, auf alle Fälle immer verwendet werden. Automatisierte Tests, zumindest der Backend Schnittstellen, Verfügbarkeitsüberwachung der Applikation, sowie einzelner APIs

und zeitgerechte Warnung bevor ein Zertifikat ausläuft, sind ebenfalls von Beginn an in einem Projekt immer zu berücksichtigen.

Bei weiteren Werkzeugen in der Entwicklungsumgebung, wie der Möglichkeit der integrierten Source-Code Verwaltung, effiziente Quellcode Indizierung und ein möglichst geringer initialer Konfigurationsaufwand sollte je nach Projektbedingung und Meinung des Teams bewertet und gegebenenfalls genutzt werden. Das selbe gilt auch für statische Source-Code Analysetools, automatische Source-Code Formatierung, einfache Installation, Nutzung und umfangreicheren Funktionsumfang für das Tool des Bereitstellungsprozesses. Die Durchführung von automatischen Oberflächenfunktionstests, Kompatibilitätstests, Lasttests und Penetrationstests ist genauso wie die detaillierte und historische Protokollierung und Performance Messung ebenfalls von den spezifischen Projekt Voraussetzungen abhängig. Gerade die Nutzung der zuletzt erwähnten Hilfsmittel bringen einen immensen Arbeitsaufwand mit. Wenn daher zum Beispiel an einer firmeninternen Webapplikation gearbeitet wird, bei der mit Sicherheit feststeht, dass sie nur in einem bestimmten Browser auf einer konkreten Bildschirmauflösung genutzt werden wird, dann macht die Arbeit für Kompatibilitätstest überhaupt keinen Sinn, da es sich in dem Fall nicht um einen realen Anwendungsfall handelt.

Ein Tool für die direkte Einbettung der Schnittstelle für die Software Infrastruktur Steuerung, sollte genauso wie das Hilfsmittel zur gemeinsamen remote Programmierung bevor der getätigten Investition gründlich überlegt werden. Dasselbe gilt für die Anschaffungen eines umfangreichen Werkzeuges für den Bereitstellungsprozess, welches auch von nicht Technikern genutzt werden kann. Werkzeuge dieser Kategorien klingen zwar in der Theorie oft sehr interessant, werden aber in der Praxis von erfahrenen Fachleuten nur wenig geschätzt. Diese verursachen meistens nur Kosten und Aufwand und haben wenn überhaupt nur einen sehr geringen Nutzen.

4.1.5 Kritik

Mit nur 52 Rückmeldungen sollte das Ergebnis dieser Umfrage etwas kritisch betrachtet werden. Normalerweise gilt bei einer Umfrage, je mehr Personen von der ausgewählten Zielgruppe daran teilnehmen, desto eher kann das Ergebnis angenommen werden und als richtig erachtet werden.

Ein weiterer Kritikpunkt ist, dass die Aufstellung von einem allgemeinen Leitbild, welche Tools genutzt und Prozesse eingehalten werden sollen, um die wirtschaftliche

Entwicklung einer Webapplikation mit einem flexiblen und skalierbaren Team zu ermöglichen, nur eine Richtschnur sein kann. In der Regel muss immer auf ein Projekt, sowie dessen Anforderungen eingegangen werden, um die richtigen Mittel zu finden um das gewünschte Ziel zu erreichen.

Zu guter Letzt, muss noch angemerkt werden, dass es sich bei der Ausarbeitung um eine Momentaufnahme handelt. In einem Jahr können schon ganz neue Möglichkeiten an Werkzeugen am Markt verfügbar sein. Es handelt sich eben um eine Materie, die sich aufgrund der starken Nachfrage und dem technologischen Fortschritt unentwegt weiterentwickelt.

5 Zusammenfassende Bemerkung

5.1 Zusammenfassung und Fazit

Werkzeuge sind in der modernen Softwareentwicklung unumgänglich. Das ist Fakt. Ohne Entwicklungsumgebung und ohne Versionskontrolle zu arbeiten, wäre wie der Versuch einen Nagel anstatt mit einem Hammer, mit der eigenen Hand hineinzuschlagen. Es würde sehr schmerzlich sein und wäre auf Dauer nicht durchführbar. Dieses Beispiel ist vergleichbar mit einer falschen oder nicht vorhandenen Werkzeugnutzung in der Softwareentwicklung.

Eine Nicht Nutzung sinnvoller Werkzeuge hat schon ab der Stunde Null in einem Projekt negative Folgen, das viel größere Problem bei der Entwicklung von Applikationen ist jedoch, dass der Einsatz so mancher Tools zu jedem späteren Zeitpunkt, oft nur mit hohem Aufwand, nämlich großen Änderungen in der bereits erstellten Software, wieder wettgemacht werden kann.

Weitere wichtige Werkzeuge sind die verschiedenen Varianten der Testautomatisierung, die Durchführung von Code-Reviews um die Code-Qualität und die Wissensverteilung zu verbessern, sowie Protokollierung und Monitoring des Betriebs. Wichtig ist aber auch, dass Hilfsmittel projektbezogen und rational eingesetzt werden. Sonst kann leicht passieren, dass einerseits Tools unnötig eingesetzt werden, wie zum Beispiel die Durchführung von Lasttests, bei einer Webapplikation wo sicher ist, dass diese nie von einer hohen Anzahl an Benutzern auszugehen ist. Andererseits darf den Werkzeugen nicht mehr Bedeutung geschenkt werden als dem eigentlichen Geschäftswert in einem Projekt.

Abschließend kann gesagt werden: Das geeignete Tool für den jeweiligen Arbeitsschritt zu verwenden, bedeutet eine enorme Ersparnis hinsichtlich Arbeitsaufwand, Testdurchführungen, Sicherheitsprüfungen und ist zusätzlich eine Absicherung des Projektes für zukünftige personalunabhängige Anforderungen beziehungsweise Anpassungen.

5.2 Ausblick und weiterer Forschungsbedarf

Ein Ausblick in die Zukunft, wie die Erarbeitung in der Praxis angewandt wird, beziehungsweise wofür das Ergebnis dieser Arbeit als Grundstein dient.

Die Arbeit ist eine Analyse, welche Werkzeuge den technischen Softwareentwicklungsprozess unterstützen. Es stellt einen allgemeinen Leitfaden dar. Ob der Einsatz von Werkzeugen oder die Einhaltung bestimmter Prozesse sinnvoll ist, hängt von der jeweiligen Ausgangssituation ab. Die in dieser Arbeit durchgeführte Analyse kann als Grundlage für weitere Untersuchungen verwendet werden, um in der Zukunft unterschiedliche Ausgangssituationen und Veränderungen der Werkzeuge zu bewerten.

Index

Literatur

[Einführung-Softwaretechnik-2021] Broy Manfred, Kuhrmann Marco: Einführung in die Softwaretechnik,
Heidelberg, Springer Verlag, 2021

[AgileSoftwareentwicklung-2015] Eckstein, Jutta: Agile Softwareentwicklung in großen Projekten,
Heidelberg, dpunkt.verlag, 2015

[SoftwareentwicklungKompUndV-2020] Brandt-Pook Hans, Kollmeier Rainer: Softwareentwicklung kompakt und verständlich,
Wiesbaden, Springer Verlag, 2020

[AgilityKompakt-2009] Hruschka Peter, Rupp Chris, Starke Gernot: Agility kompakt
Heidelberg, Spektrum Akademischer Verlag, 2009

[lerneProgrammieren-com] Mertens Robert: Lerne Programmieren – Starte deine Zukunft, https://lerneprogrammieren.com/, verfügbar am 18.05.2021, 13:44

[BestPracticeSoftwareEng-2010] Schatten Alexander, Demolsky Markus, Winkler Dietmar, Biffl Stefan, Gostischa-Franta Erika, Österreicher Thomas: Best Practice in Software-Engineering,
Heidelberg, Spektrum Akademischer Verlag, 2010

[CleanCode-2009] Martin Robert C.: Clean Code - Refactoring, Patterns, Testen und Technik für sauberen Code
Heidelberg, mitp-Verlag, 2009

[QualitätssicherungDurchSoftwaretests-2013] Kleuker Stephan: Qualitätssicherung durch Softwaretests, Wiesbaden, Springer Verlag, 2013

[LehrbuchDerSoftwaretechnik-2011] Balzert Helmut: Lehrbuch der Softwaretechnik – Entwurf, Implementierung, Installation und Betrieb, Heidelberg, Spektrum Akademischer Verlag, 2011

[AgileInDerUnternehmenspraxis-2017] Hanschke Inge: Agile in der Unternehmenspraxis – Fallstricke erkennen und vermeiden, Potenziale heben, Wiesbaden, Springer Verlag, 2017

[BestWebDevelopmentFrameworks-2021] Sakovich Natallia: Sam Solutions – 10 Best Web Development Frameworks in 2021, https://www.sam-solutions.com/blog/web-development-frameworks/, verfügbar am 21.05.2021, 18:32

[WebseitenProgrammierung-2012] Pomaska Günter: Webseiten Programmierung – Sprachen, Werkzeuge, Entwicklung, Wiesbaden, Springer Verlag, 2012

[DerWegInDieCloud-2020] Lindner Dominic Niebler Paul, Wenzel Markus: Der Weg in die Cloud – Ein Leitfaden für Unternehmer und Entscheider, Wiesbaden, Springer Verlag, 2020

[10BestIdeSoftware-2021] McKinnon Jenni: WebsiteSetup – 10 Best IDE Software, https://websitesetup.org/best-ide-software/, verfügbar am 01.06.2021, 16:20

[StaticAnalysisJavaS Hughes Karl: LogRocket – Static Analysis in JavaScript – 11 tools

cript-2020] to help you catch errors before users do, https://blog.logrocket.com/static-analysis-in-javascript-11-tools-to-help-you-catch-errors-before-users-do/, verfügbar am 02.06.2021, 19:05

[BestCI/CDTools-2020] Dhabekar Praful: Medium – Top 7 Best CI/CD Tools you should get your hands on in 2020, https://medium.com/devops-dudes/top-7-best-ci-cd-tools-you-should-get-your-hands-on-in-2020-832c29db936a, verfügbar am 03.06.2021, 14:55

[BestTestingToolsWeb-2021] Cherednichenko Sveta: mobindustry – 16 Best Automation Testing Tools for Web Applications in 2021, https://www.mobindustry.net/15-best-automation-testing-tools-for-web-applications/, verfügbar am 03.06.2021, 17:08

[Selenium-IDE-2013] Neff Thomas: Phase2 – Selenium IDE: Automated Testing Made Easy, https://www.phase2technology.com/blog/selenium-ide-automated-testing-made-easy, verfügbar am 03.06.2021, 20:33

[WebsiteMonitoringTools-2021] Osman Maddy: WebsiteSetup – Website Monitoring Tools, https://websitesetup.org/website-monitoring-services/, verfügbar am 05.06.2021, 15:38

[ForschungsmethodenUndEvaluation-2016] Döring Nicola, Börtz Jürgen: Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften, Berlin, Heidelberg, Springer Verlag, 2016

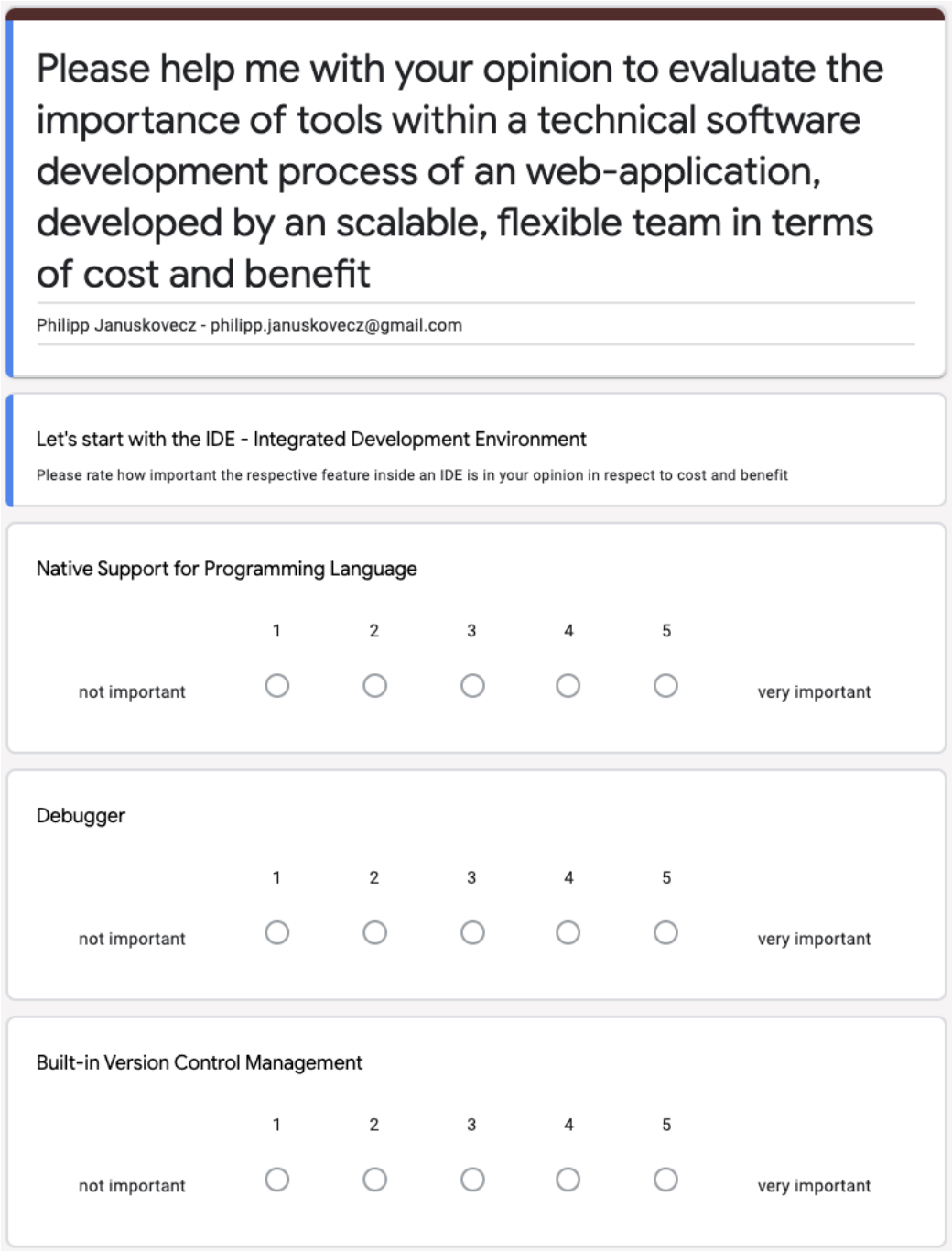
[Fragebögen-2016] Hollenberg Stefan: Fragebögen – Fundierte Konstruktion, sachgerechte Anwendung und aussagekräftige Auswertung,

Wiesbaden, Springer Verlag, 2016

[12BestCodeReview Tools-2020] Shaumik Daityari: Kinsta Blog – 12 Best Code-Review Tools für Entwickler, https://websitesetup.org/website-monitoring-services/, verfügbar am 18.07.2021, 17:48

Anlagen

Anlagen, Teil 1



Please help me with your opinion to evaluate the importance of tools within a technical software development process of an web-application, developed by an scalable, flexible team in terms of cost and benefit

Philipp Januskovecz - philipp.januskovecz@gmail.com

Let's start with the IDE - Integrated Development Environment

Please rate how important the respective feature inside an IDE is in your opinion in respect to cost and benefit

Native Support for Programming Language

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Debugger

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Built-in Version Control Management

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Automatic Code Completion

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Built-in Infrastructure Integration

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Quick and Reliable Source Code Indexing

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Out-of-the-box Experience (without configuration effort)

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Expandability by Plug-Ins

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Live Work Collaboration (pair programming support for several remote developers)

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Let's Continue with the Source Code Management

Please rate how important the respective aspect regarding Source Code Management is in your opinion in respect to cost and benefit

General use of distributed source code management system in every project (eg. GIT)

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Next point is all about Code Quality

Please rate how important the respective adjustment regarding Code Quality is in your opinion in respect to cost and benefit

Linter for Static Code Analysis

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Auto Formatting of the Code

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Regular extensive execution of Static Code Analysis (e.g SonarCloud at every build)

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Code Review - to improve Code Quality

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Code Review - to improve Knowledge Distribution

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Let's Continue with the Continuous-Integration Process Tool

Please rate how important the respective aspects of the Build, Deployment and Release Process tool are in your opinion in respect to cost and benefit

Easy installation and operation of the tool

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Many features (e.g schedules, notifications, ...)

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Expandability by Plug-Ins

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Usable by non-developers

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Comfortable integration between the rest of the used tools (e.g IDE, project management software, communication tool and infrastructure)

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Next point is all about Testing

Please rate how important the different test strategies are in your opinion in respect to cost and benefit

Automated UI Tests

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Automated API Tests (Integration Tests)

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Compatibility test (e.g. different browsers/screen sizes)

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Load Test

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Penetration Test

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Creation/execution and maintenance of the tests should be possible with as little technical knowledge as possible

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Last but not least - Operation and Maintenance

Please rate how important the respective aspects of Operation and Maintenance are in your opinion in respect to cost and benefit

Monitoring the Availability of the application (alerting in the event of failure)

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Monitoring the Error Rate of the APIs

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Monitoring the Performance of the application

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Notification (e.g expiring certificates)

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Logging as detailed as possible

	1	2	3	4	5	
not important	○	○	○	○	○	very important

Logging as far back as possible

	1	2	3	4	5	
not important	○	○	○	○	○	very important

In General (all segments from IDE to Operation) -Tool supported and maintained by a company

	1	2	3	4	5	
not important	○	○	○	○	○	very important

In General (all segments from IDE to Operation) -Tool is Open-Source

	1	2	3	4	5	
not important	○	○	○	○	○	very important

I have one more request, please tell me your current role and the number of your years of experience in software development.

Beschreibung (optional)

My current Role is

1. Project Manager/Scrum Master
2. Requirements Engineer
3. Developer
4. Tester
5. Designer
6. Operation Engineer
7. Architect

My years of experience in software development are

1. 1
2. 2
3. 3+
4. 5+
5. 8+
6. 13+
7. 20+
8. 40+

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Ternitz, den 26.07.2021

Philipp Januskovecz