Proposal zur Masterthesis

Featurebasierte Fehlervorhersage mittels Methoden des Machine Learnings

Stefan Hermann Strüder

25. September 2019

Universität Koblenz-Landau Fachbereich Informatik

Arbeitsgruppe Software Engineering – Prof. Dr. Jan Jürjens

betreut durch: Dr. Daniel Strüber

Stefan Hermann Strüder

stefanstrueder@uni-koblenz.de
Matrikelnummer: 214200670
Studiengang: Master Informatik

Proposal zur Masterthesis
Thema: Featurebasierte Fehlervorhersage mittels Methoden des Machine Learnings

Eingereicht: 25. September 2019

Betreuer: Dr. Daniel Strüber

Prof. Dr. Jan Jürjens Arbeitsgruppe Software Engineering
Fachbereich Informatik
Universität Koblenz-Landau
Universitätsstraße 1
56070 Koblenz

Inhaltsverzeichnis

Inhaltsverzeichnis III
1 Einleitung und Motivation 1
2 Ziele und Vorgehensweise 3
2.1 Zielsetzung und Forschungsfragen 3
2.2 Methodik 4
2.3 Vorläufige Ablaufplanung 6
3 Verwandte Literatur 9
Literaturverzeichnis 11
Anhang 12

1 Einleitung und Motivation

Softwarefehler stellen einen erheblichen Auslöser für finanzielle Schäden und Rufschädigungen von Unternehmen dar. Solche Fehler reichen von kleineren „Bugs“ bis hin zu schwerwiegenden Sicherheitslücken. Aus diesem Grund herrscht ein großes Interesse daran, einen Entwickler zu warnen, wenn er aktualisierten Softwarecode veröffentlicht, der möglicherweise einen Fehler beinhaltet.

Zu diesem Zweck haben Forscher im vergangenen Jahrzehnt verschiedene Techniken zur Fehlererkennung und Fehlervorhersage entwickelt, die zu einem Großteil auf Methoden und Techniken des *Machine Learnings* basieren. Diese verwenden in der Regel historische Daten von fehlerhaften und fehlerfreien Änderungen an Softwaresystemen in Kombination mit einer sorgfältig zusammengestellten Menge von *Features* (im Sinne von Charakteristika von Daten), um einen gegebenen Klassifikator zu anzulernen beziehungsweise zu trainieren. Dieser soll dann eine akkurate Vorhersage treffen, ob eine neu erfolgte Änderung an einer Software fehlerbehaftet oder frei von Fehlern ist.

Die Auswahl an Lernverfahren für Klassifikatoren ist groß. Studien zeigen, dass innerhalb dieser Verfahren sowohl Entscheidungsbaum-basierte (zum Beispiel J48, CART oder Random Forest) als auch bayessche Verfahren die meistgenutzten sind (Son et al., 2019). Alternative Lernmethoden sind Regression, k-Nearest-Neighbor oder neuronale Netze (Challagulla, Bastani, Yen, & Paul, 2008). Anzumerken ist allerdings, dass es keinen Konsens über die beste verfügbare Lernverfahren gibt, da jedes Verfahren unterschiedliche Stärken und Schwächen für bestimmte Anwendungsfälle aufweist.

Das Ziel dieser Arbeit ist die Entwicklung einer Vorhersagetechnik für Softwarefehler basierend auf *Software-Features*. Diese beschreiben Inkremente der Funktionalität eines Softwaresystems. Die auf diese Weise entwickelten Softwaresysteme heißen *Software-Produktlinien*. Diese bestehen aus einer Menge von ähnlichen Softwareprodukten und zeichnen sich dadurch aus, dass sie eine gemeinsame Menge von Features sowie eine gemeinsame Codebasis besitzen (Thüm, Apel, Kästner, Schaefer, & Saake, 2014). Durch das Vorhandensein verschiedener Features entlang der Softwareprodukte, kann eine breite Variabilität innerhalb einer Produktlinie erreicht werden.

Die nachfolgende Abbildung 1.1 zeigt den zentralen Prozess der Entwicklung einer Produktlinie. Aufgeteilt wird dieser in das *Domain Engineering* und das *Application Engineering.* Im Rahmen des Domain Engineerings wird ein sogenanntes Variabilitätsmodell (Variability Model) erzeugt, welches durch die Kombination der wählbaren Features beschrieben wird (Apel, Batory, Kästner, & Saake, 2013). Gängige Implementationstechniken für Features reichen von einfachen Lösungen durch Annotationen basierend auf Laufzeitparametern oder Präprozessor-Anweisungen bis hin zu verfeinerten Lösungen

basierend auf erweiterten Programmiermethoden, wie zum Beispiel Aspektorientierung. In Teilen dieser Implementierungstechniken wird jedes Feature wird als wiederverwendbares Domain Artifact modelliert und gekapselt, welches im Prozess des Application Engineerings in Form einer Konfiguration zusammen mit weiteren Features, im Hinblick auf die gewünschte Funktionalität der Software, ausgewählt werden kann. Ein Software Generator erzeugt dann die gewünschten Software Produkte basierend auf den bereits zuvor genannten Implementationstechniken für Features.

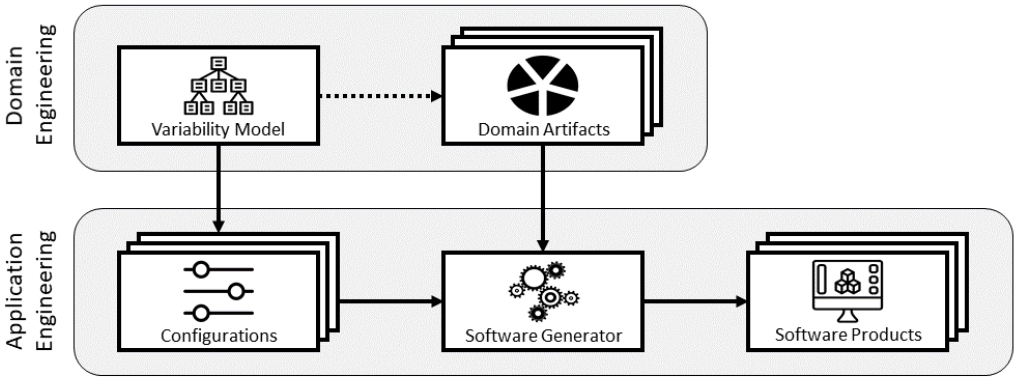


Abbildung 1.1: Generierung von Software-Produktlinien (nach Thüm et al., 2014)

Das Ziel dieser Arbeit ist es, eine auf Machine Learning gestützte Vorhersagetechnik unter Verwendung von Software-Features zu entwickeln. Diese Idee ist aufgrund mehrerer Gründe chancenreich:

1. Wenn ein bestimmtes Feature in der Vergangenheit mehr oder weniger fehleranfällig war, so ist eine Änderung, die das Feature aktualisiert, wahrscheinlich ebenfalls mehr oder weniger fehleranfällig.
2. Features, die mehr oder weniger fehleranfällig scheinen, könnten besondere Eigenschaften haben, die im Rahmen der Fehlervorhersage verwendet werden können.
3. Code, der viel Feature-spezifischen Code enthält (insbesondere die sogenannten Feature-Interaktionen), ist möglicherweise fehleranfälliger als sonstiger Code.

Dieses Ziel setzt sich aus mehreren Teilzielen zusammen. Dazu zählen die Erstellung eines Datensets zum Trainieren von Machine-Learning-Klassifikatoren sowie das Anlernen einer repräsentativen Auswahl an Klassifikatoren mit anschließender vergleichender Evaluation dieser. Ein genauer Überblick über die Forschungsziele befindet sich in Kapitel 2.1.

Sollte sich einer dieser Klassifikatoren in der Evaluation als besonders effektiv erweisen, so würde diese Arbeit den Stand der Technik hinsichtlich der Fehlervorhersage in Features vorantreiben und Organisationen erlauben, bessere Einblicke in die Fehleranfälligkeit von Änderungen in ihrer Codebasis zu erhalten.

2 Ziele und Vorgehensweise

Dieses Kapitel stellt die Zielsetzung der Arbeit mit den zugehörigen Forschungsfragen vor. Zudem werden die ausgewählte Methodik, namentlich das Prozessmodell CRISP-DM, sowie der vorläufige Zeitplan erläutert.

2.1 Zielsetzung und Forschungsfragen

Wie bereits in der Einleitung beschrieben, ist das übergeordnete Ziel dieser Arbeit die Entwicklung einer Vorhersagetechnik für Fehler in featurebasierter Software unter Zuhilfenahme von Methoden des Machine Learnings. Dazu ist vorgesehen, das Augenmerk auf Commits von Versionierungssystemen, wie beispielsweise Subversion oder Git, zu richten. Ein Commit bezeichnet dabei die zur Verfügungstellung einer aktualisierten Version einer Software. Als Datenbasis für das Trainieren der Klassifikatoren dienen dann fehlerhafte und fehlerfreie Commits von featurebasierter Software. Dies ermöglicht es, ausstehende defekte Commits vorherzusagen und das Risiko der Konsequenzen von Softwarefehlern zu senken.

Der Prozess der Entwicklung der Vorhersagetechnik ist in drei zu erreichende Forschungsziele eingeteilt. Jedem Forschungsziel werden Forschungsfragen zugeordnet, deren Aufklärung einen zusätzlichen Teil zur Erfüllung der Ziele beiträgt. Im Folgenden werden die Forschungsziele (RO – „research objective“) mit ihren zugehörigen Forschungsfragen (RQ – „research question“) vorgestellt.

RO1: Erstellung eines Datensets zum Trainieren von relevanten Machine-Learning-Klassifikatoren

RQ1a: Welche Daten kommen für die Erstellung des Datensets in Frage?

RQ1b: Wie weit müssen die Daten vorverarbeitet werden, um sie für das Training nutzbar zu machen?

RO2: Identifikation und Training einer Auswahl von relevanten Machine Learning Klassifikatoren basierend auf dem Datenset

RQ2: Welche Machine Learning Klassifikatoren kommen für die gegebene Aufgabe in Frage?

RO3: Evaluierung und Gegenüberstellung der Klassifikatoren sowie Vergleich zu modernen Vorhersagetechniken, die keine Features nutzen

RQ3a: Welche miteinander vergleichbaren Merkmale besitzen die Klassifikatoren?

RQ3b: Welche Metriken können für den Vergleich verwendet werden?

RQ3c: Welche Vor- und Nachteile besitzt ein Klassifikator?

RQ3d: Wie lassen sich die Klassifikatoren mit weiteren Vorhersagetechniken, die keine Features nutzen, vergleichen?

Zusätzlich zu den drei Forschungszielen gehören eine Vor- und Nachbereitung der Arbeit, sodass sich insgesamt fünf Arbeitsphasen ergeben. Diese werden in den weiteren Unterkapiteln näher erläutert. Als finale Vorhersagetechnik wird jener Klassifikator verwendet, der im Rahmen der Gegenüberstellung während der Evaluation als am effektivsten hervorgeht.

2.2 Methodik

Die für diese Arbeit gewählte Methodik basiert auf dem Prozessmodell *Cross-Industry Standard Process for Data Mining*, kurz *CRISP-DM*, nach Chapman et al. (2000). Es wird als Vorlage für die Arbeitsphasen für die Erreichung der Forschungsziele dieser Arbeit verwendet. Da der überwiegende praktische Teil dieser Arbeit durch Programmierung im Bereich des Machine Learning geschieht, bildet das CRISP-DM Prozessmodell ein passendes vordefiniertes Vorgehen. Dieses wurde speziell für die Erarbeitung von *Data Mining* Projekten entwickelt, eignet sich jedoch auch für die Verwendung im Rahmen des Machine Learnings, da sich die in beiden Bereichen verwendeten Methoden und Prozesse zu einem erheblichen Teil überlagern. Ein Überblick über die sechs Phasen des Prozessmodells ist in Abbildung 2.1 dargestellt. Zusätzlich umfasst die Abbildung die Zuordnung der Arbeitsphasen. Einen genauen Überblick über den konkreten Umfang der Arbeitsphasen bietet das im Anschluss folgende Unterkapitel 2.3.

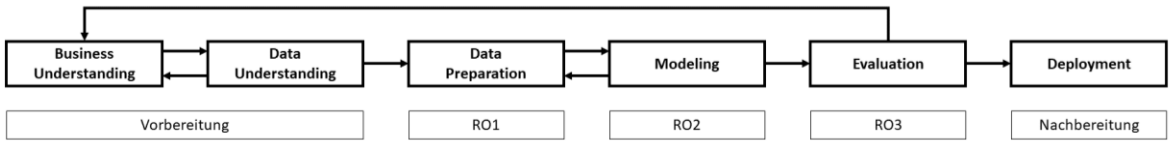


Abbildung 2.1: Phasen des CRISP-DM Prozessmodells (Chapman et al., 2000) mit Zuordnung der Arbeitsphasen

Die ersten beiden Phasen *Business Understanding* und *Data Understanding* widmen sich der Vorbereitung der Arbeit. Die initiale Phase umfasst dabei die allgemeine Einarbeitung in das zugrundeliegende Thema und der Formulierung der Forschungsziele. Anzumerken ist, dass diese Phase bereits vor der sechsmonatigen Bearbeitungszeit der Arbeit beginnt und somit

schon das Verfassen dieses Proposals als Teilaufgabe dieser Phase gezählt werden kann, da es auch eine grobe Einarbeitung in das Thema erfordert.

Die darauffolgende Phase *Data Understanding* dient der Suche und Einsicht von für den weiteren Verlauf der Phasen relevanten Daten und, falls vorhanden, vorgefertigten Datensets. Da Commits als Datenbasis zur Erlernung der Klassifikatoren betrachtet werden, wird der überwiegende Teil der Suche nach Daten auf dem Onlinedienst GitHub stattfinden, welchem das Versionierungssystemen Git zugrunde liegt. Für die weiteren Phasen ist es von besonderer Bedeutung, den Aufbau der Daten sorgfältig zu untersuchen.

Die dritte Phase *Data Preparation* kümmert sich um die Erstellung eines endgültigen Datensets und den dort hinführenden Prozessen. Diese Phase ist deckungsgleich mit den Anforderungen des ersten Forschungsziels.

Zur Anwendung kommt das im vorherigen Schritt erstellte Datenset in der Phase *Modeling*. In dieser werden die Data-Mining-Algorithmen und -Techniken gemäß den zugrundeliegenden Anforderungen auf das Datenset angewendet. Adaptiert an das Lernen der Machine Learning Klassifikatoren spiegelt dies die Arbeitsphase zur Erfüllung des zweiten Forschungsziels dar.

Die fünfte Phase umfasst die *Evaluation* der Resultate des zuvor erfolgten Schrittes und deckt somit die Erfüllung des dritten Forschungsziels ab.

Die Nachbereitung der Arbeit wird durch die Phase *Deployment* abgedeckt. Diese umfasst die Erstellung der finalen Ausarbeitung sowie der Abschlusspräsentation und der anschließenden Vorführung dieser im Rahmen des Kolloquiums.

Es ist zu erkennen, dass die Beschreibung der Arbeitsphasen weitestgehend auf einem theoretischen Level verfasst wurde. Es wird anhand der fünf CRISP-DM-Phasen gezeigt, *was* für den erfolgreichen Abschluss der Arbeit absolviert werden muss. Die Erörterung der Frage, *wie* die einzelnen zu erledigenden Aufgaben durchgeführt werden müssen, ist Teil der Vorbereitung der Arbeit. Im Rahmen der Phasen Business Understanding und Data Understanding wird nach einer eingehenden Recherche die genaue Methodik festgelegt (siehe Unterziele der ersten Phase im kommenden Abschnitt).

2.3 Vorläufige Ablaufplanung

Im Nachfolgenden wird die vorläufige Ablaufplanung der Arbeit aufgezeigt. Die Ordnung erfolgt gemäß der Aufteilung in die fünf zuvor beschriebenen Arbeitsphasen. Die geschätzte Dauer der verschiedenen Unterziele wird jeweils in Tagen, Wochen oder Monaten angegeben. Zur Verfügung stehen insgesamt sechs Monate Bearbeitungszeit.

Phase 1: Vorbereitung

Unterziele	**Dauer**	
Strukturierte Literaturrecherche • Techniken der featurebasierten Softwareprogrammierung • Techniken zur Fehlererkennung in Software • Klassifikation mittels Machine Learning • Klassifikationsmethoden • Auswahl der Programmiersprache • Tool- & Libraryauswahl • Evaluationsmetriken	2 Wochen	Business Understanding
Recherche zur Bildung eines Datensets • Merkmale / Aufbau eines Datensets • Suche nach Datenquellen • Suche nach vorgefertigten Datensets • Prüfung der Daten / Datensets auf Eignung • Analyse des Aufbaus der Daten / Datensets	1 Woche	Data Understanding
Total:	3 Wochen	

Phase 2: Forschungsziel 1 – Erstellung des Datensets (Data Preparation)

Unterziele	**Dauer**
finale Datenauswahl • Festlegung von Kriterien	1 Woche
Datenbereinigung • „Preprocessing“	1 Woche
finale Konstruktion des Datensets • Integration der Daten und des Feature-Aspekts • erneute abschließende Bereinigung sowie Formatierung • Teilung in Training-Set und Test-Set	1 Woche
Total:	3 Wochen

Phase 3: Forschungsziel 2 – Training der Machine Learning Klassifikatoren (Modeling)

Unterziele	**Dauer**
Auswahl geeigneter Klassifikatoren	1 Woche
Training der Klassifikatoren	3 Wochen
Total:	4 Wochen

Phase 4: Forschungsziel 3 – Evaluation und Vergleich der Machine Learning Klassifikatoren

Unterziele	**Dauer**
Evaluation der einzelnen Klassifikatoren • Festlegung der Metriken • Anwendung des Test-Sets • Berechnung der Metriken	2 Wochen
Vergleich der Klassifikatoren anhand der Metriken	2 Wochen
Vergleich mit weiteren Vorhersagetechniken, die nicht auf Features setzen	1 Woche
Total:	5 Wochen

Phase 5: Nachbereitung (Deployment)

Unterziele	**Dauer**
Besprechung der vorangegangenen Arbeit mit Betreuer • Umsetzung möglicher Verbesserungsvorschläge	1 Woche
Erstellung der Ausarbeitung*	7 Wochen
Erstellung der Abschlusspräsentation	1 Woche
Total:	9 Wochen

* Die Erstellung der Ausarbeitung ist ein laufender Prozess über den gesamten Verlauf der Bearbeitungszeit. Der hier erwähnte siebenwöchige Zeitraum dient unter Anderem zur Korrektur beziehungsweise Verbesserung hinsichtlich des Feedbacks des Betreuers und zur abschließenden Finalisierung.

Die nachfolgende Abbildung zeigt den zeitlichen Ablauf der Arbeit als Gantt-Chart inklusive konkreter Datumsangaben. Eine größere Version des Plans befindet sich im Anhang.

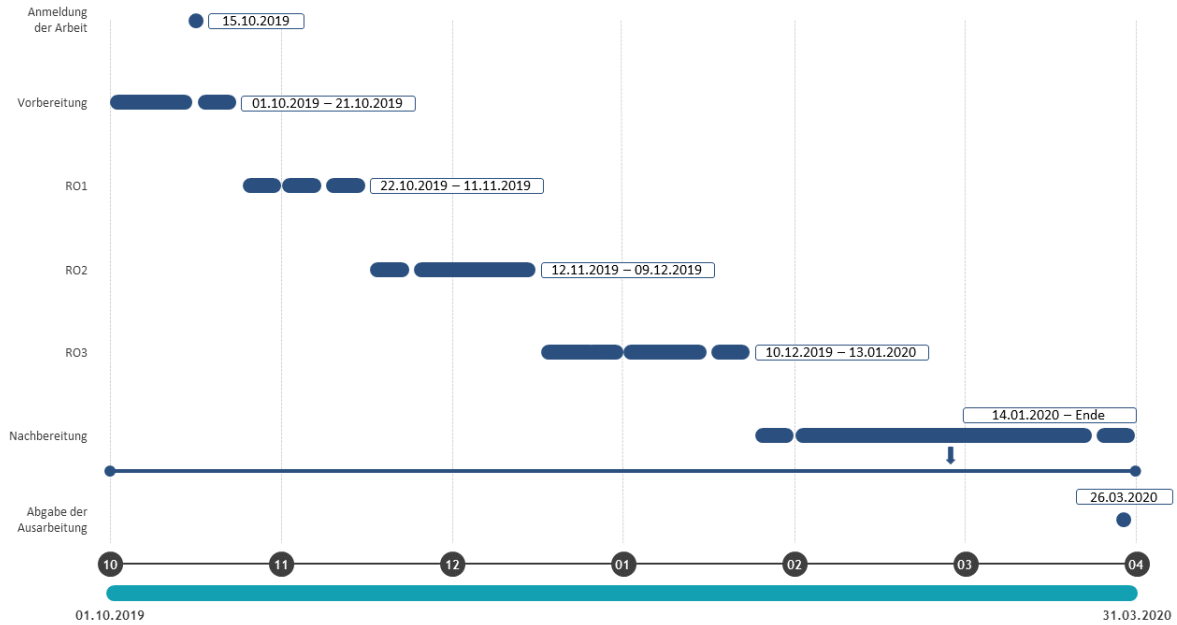


Abbildung 2.2: Zeitlicher Ablaufplan der Arbeit als Gantt-Chart

Der tatsächliche Ablauf kann während der sechsmonatigen Bearbeitungszeit der Arbeit abweichen. Sämtliche Abweichungen werden in der finalen Ausarbeitung kenntlich gemacht.

3 Verwandte Literatur

Dieses Kapitel dient einer kurzen Einführung von relevanter Literatur hinsichtlich des Themas der Arbeit. Eine detaillierte Auseinandersetzung mit vorhandener Literatur wird in der ersten Phase der Arbeit stattfinden.

Es existieren bereits eine Reihe von wissenschaftlichen Arbeiten zum Thema der Fehlervorhersage in Software. Die Erforschung der Fehlervorhersage in featurebasierter Software geschieht jedoch bisher in einem weitestgehend geringen Umfang.

Im Rahmen einer gemeinsamen wissenschaftlichen Arbeit der Universitäten Waterloo (Kanada) und Göteborg (Schweden) mit dem Titel *„Towards Predicting Feature Defects in Software Product Lines“* entstand im Jahr 2016 das bisher einzige Werk, dass sich mit dem zuletzt genannten Thema befasst (Queiroz, Berger, & Czarnecki, 2016). Die Fallstudie beschreibt einen ersten eingegrenzten Ansatz zur Erstellung eines Klassifikators zur Vorhersage von defekten Features und dient zur Richtungsweisung für zukünftige Arbeiten bezüglich des Themas. Es werden die Erstellung eines kleinen Datensets sowie das anschließende Trainieren von J48- (basierend auf Decision Trees), Naive-Bayes- und Random-Forest- Klassifikatoren mit anschließender Evaluation und einem Vergleich auf Basis verschiedener Metriken beschrieben. Diese Arbeit bildet die einführende Forschungsgrundlage für die in diesem Proposal vorgestellte Masterthesis unter der Abgrenzung, dass das Ziel der Thesis darin besteht, defekte Commits vorherzusagen.

Einen analytischen Einblick in das Thema der featurebasierten Programmierung von Software bietet das Paper *„What is a Feature? A Qualitative Study of Features in Industrial Software Product Lines“* aus dem Jahr 2015 (Berger et al., 2015). Es werden die Charakteristika von Features vorgestellt sowie aufgeführt, wie sich „gute“ von „schlechten“ Features unterscheiden. Die in diesem Paper vorgestellten Erkennungsmerkmale von Features werden insbesondere für die Modellierung des Datensets von Relevanz sein.

Ein Beispiel für die Fehlervorhersage von Software, die nicht auf Features aufgebaut ist, gibt das Paper *„Predicting Defects for Eclipse“* aus dem Jahr 2007 (Zimmermann, Premraj, & Zeller, 2007). Es beschreibt, wie ein Datenset aus der Fehlerdatenbank der Entwicklungsumgebung Eclipse erstellt und zum Anlernen eines Klassifikators verwendet wurde. Aus diesem Paper können Ansätze für die Programmierung der Vorhersagetechniken entnommen werden, sofern sie auf Features übertragbar sind.

Eine Einführung in das Thema der Metriken, die im Rahmen der Evaluation Gebrauch finden werden, bietet die wissenschaftliche Arbeit „How, and Why, Process Metrics Are Better“ (Rahman & Devanbu, 2013). Insbesondere die dort vorgestellten Metriken zur Performanz- und Stabilitätsmessung der Vorhersagetechniken, bilden eine solide Wissensgrundlage für die Anwendung im Rahmen der Evaluation.

Grundlegende Informationen zum Thema Software-Produktlinien konnten aus den Büchern „Software Product Line Engineering: Foundations, Principles and Techniques“ (Pohl, Böckle, & van der Linden, 2005) und „Feature-Oriented Software Product Lines“ (Apel et al., 2013) entnommen werden.

Zur Einführung in das Thema der Machine Learning gestützten Fehlervorhersage dienten die wissenschaftlichen Arbeiten „Empirical Assessment of Machine Learning based Software Defect Prediction Techniques“ (Challagulla et al., 2008) und „Empirical Study of Software Defect Prediction: A Systematic Mapping“ (Son et al., 2019).

Literaturverzeichnis

Apel, S., Batory, D., Kästner, C., & Saake, G. (2013). *Feature-Oriented Software Product Lines*. https://doi.org/10.1007/978-3-642-37521-7

Berger, T., Lettner, D., Rubin, J., Grünbacher, P., Silva, A., Becker, M., … Czarnecki, K. (2015). What is a feature? *Proceedings of the 19th International Conference on Software Product Line - SPLC '15*, *3*(1), 16–25. https://doi.org/10.1145/2791060.2791108

Challagulla, V. U. B., Bastani, F. B., Yen, I. L., & Paul, R. A. (2008). Empirical assessment of machine learning based software defect prediction techniques. *International Journal on Artificial Intelligence Tools*, *17*(2), 389–400. https://doi.org/10.1142/S0218213008003947

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0. *CRISP-DM Consortium*, 76. https://doi.org/10.1109/ICETET.2008.239

Pohl, K., Böckle, G., & van der Linden, F. (2005). *Software Product Line Engineering*. https://doi.org/10.1007/3-540-28901-1

Queiroz, R., Berger, T., & Czarnecki, K. (2016). Towards predicting feature defects in software product lines. *FOSD 2016 - Proceedings of the 7th International Workshop on Feature-Oriented Software Development, Co-Located with SPLASH 2016*, 58–62. https://doi.org/10.1145/3001867.3001874

Rahman, F., & Devanbu, P. (2013). How, and why, process metrics are better. *Proceedings - International Conference on Software Engineering*, 432–441. https://doi.org/10.1109/ICSE.2013.6606589

Son, L. H., Pritam, N., Khari, M., Kumar, R., Phuong, P. T. M., & Thong, P. H. (2019). Empirical study of software defect prediction: A systematic mapping. *Symmetry*, *11*(2). https://doi.org/10.3390/sym11020212

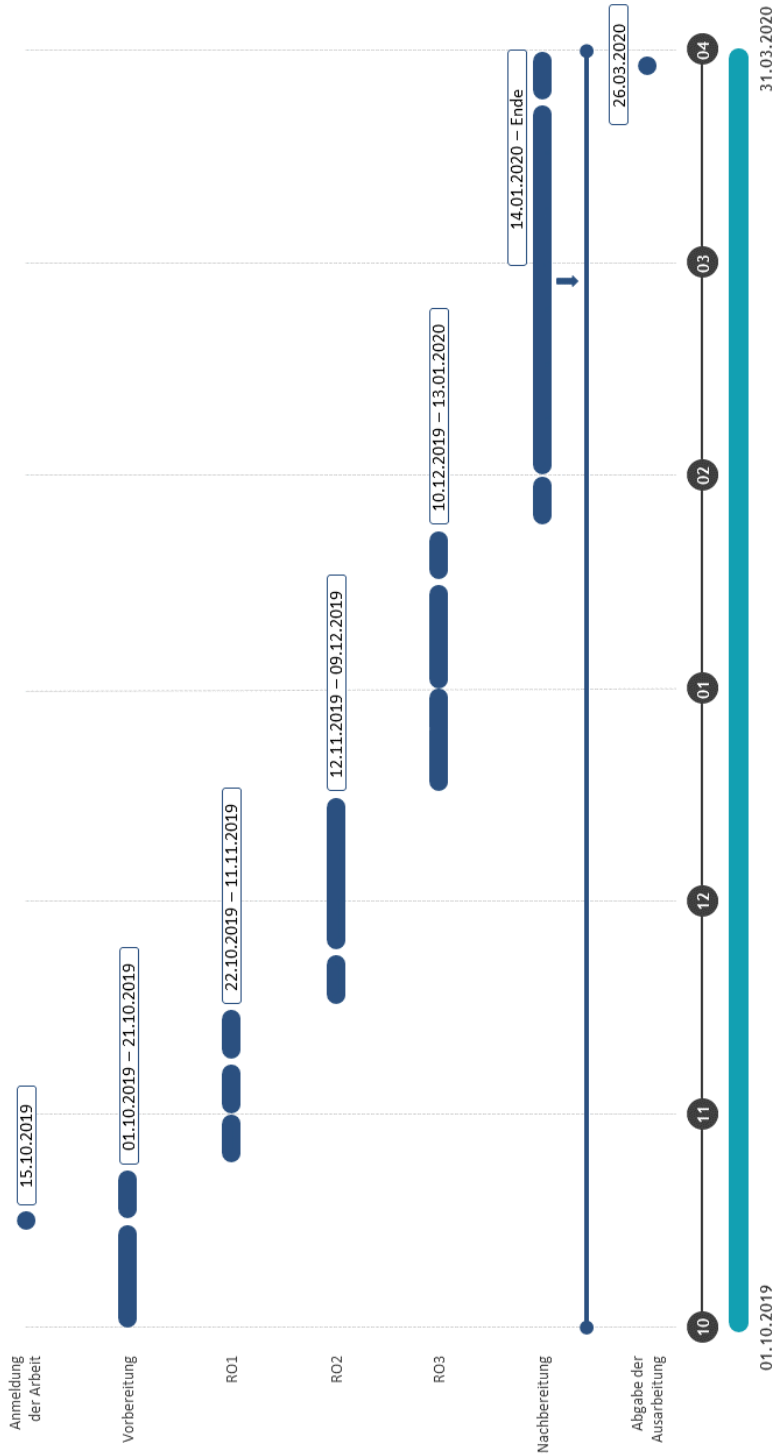
Thüm, T., Apel, S., Kästner, C., Schaefer, I., & Saake, G. (2014). A classification and survey of analysis strategies for software product lines. *ACM Computing Surveys*, *47*(1). https://doi.org/10.1145/2580950

Zimmermann, T., Premraj, R., & Zeller, A. (2007). Predicting defects for eclipse. *Proceedings - ICSE 2007 Workshops: Third International Workshop on Predictor Models in Software Engineering, PROMISE'07*. https://doi.org/10.1109/PROMISE.2007.10

Quellenverzeichnis der Icons von Abbildung 1.1:

- decision tree by Becris from the Noun Project
- Five Parts by Arthur Shlain from the Noun Project
- configuration by joe pictos from the Noun Project
- Gears by Lisa Oregioni from the Noun Project
- Software by Creaticca Creative Agency from the Noun Project

Anhang


15.10.2019
01.10.2019 – 21.10.2019
22.10.2019 – 11.11.2019
12.11.2019 – 09.12.2019
10.12.2019 – 13.01.2020
14.01.2020 – Ende
26.03.2020
Anmeldung der Arbeit
Vorbereitung
RO1
RO2
RO3
Nachbereitung
Abgabe der Ausarbeitung
10
11
12
01
02
03
04
01.10.2019
31.03.2020