UNIVERSITÄT KOBLENZ · LANDAU

Fachbereich 4: Informatik

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering

Featurebasierte Fehlererkennung mittels Methoden des Machine Learnings

Masterarbeit

zur Erlangung des Grades Master of Science (M.Sc.)
im Studiengang Informatik

vorgelegt von

Stefan Hermann Strüder

Erstgutachter:	Prof. Dr. Jan Jürjens Institut für Softwaretechnik
Zweitgutachterin:	Dr. Daniel Strüber Chalmers University of Technology, Göteborg (Schweden)

Koblenz, im Januar 2020

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Kurzfassung

Dies ist die Kurzfassung in Deutsch.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Abstract

This is the abstract in English.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Anmerkung

Diese Masterarbeit entstand in Teilen in Zusammenarbeit mit der Forschungsgruppe der Division of Software Engineering unter der Leitung von Thorsten Berger am Department of Computer Science and Engeneering der Chalmers Universität of Technology in Göteborg, Schweden.

Mein besonderer Dank gilt Thorsten Berger für die Ermöglichung und Finanzierung dieser Zusammenarbeit. Ebenfalls gilt mein Dank dem gesamten Team der Forschungsgruppe für die Unterstützung bei Problemen und Fragen zu meiner Arbeit. Ein weiterer Dank gilt Daniel Strüber für seine Initiative zur Ermöglichung der Zusammenarbeit.

Comment

This master thesis was partly written in cooperation with the research group of the Division of Software Engineering headed by Thorsten Berger at the Department of Computer Science and Engineering of Chalmers University of Technology in Gothenburg, Sweden.

My special thanks goes to Thorsten Berger for facilitating and financing this cooperation. I would also like to thank the entire team of the research group for their support in case of problems and questions concerning my work. A further thank you goes to Daniel Strüber for his initiative to make this cooperation possible.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung und Motivation

ÜBERARBEITEN!

Ausblick: Dieses Kapitel dient zur allgemeinen Einführung in diese Masterarbeit. Dazu werden neben einer Einleitung und Motivation in das zugrundeliegende Thema, die grundlegenden Strukturen der Arbeit erläutert. Dazu gehören die Forschungsziele und Forschungsfragen, das verwendete Forschungsdesign, eine Übersicht der angedachten und tatsächlichen Zeitplanung sowie eine Erläuterung des Aufbaus der weiterführenden Teile dieser Arbeit.

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 [4]. Alternative Lernmethoden sind Regression, k-Nearest-Neighbor oder neuronale Netze [2]. 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

[5]. 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 [1]. 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.

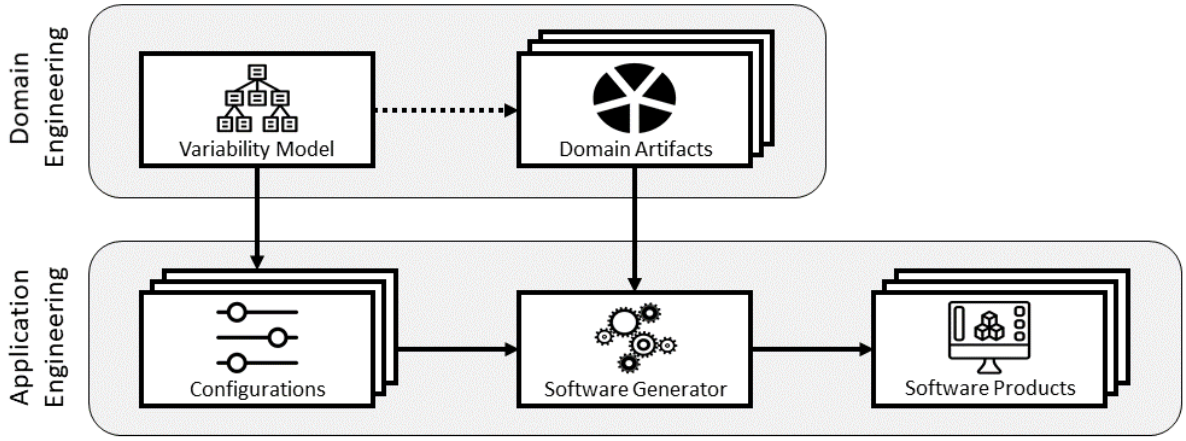

Abbildung 1.1: Generierung von Software-Produktlinien nach [5]

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 im nächsten Abschnitt.

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.

1.1 Forschungsziele und Forschungsfragen

ÜBERARBEITEN!

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.

1.2 Forschungsdesign

ÜBERARBEITEN!

Die für diese Arbeit gewählte Methodik basiert auf dem Prozessmodell Cross-Industry Standard Process for Data Mining, kurz CRISP-DM, nach [3]. 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.

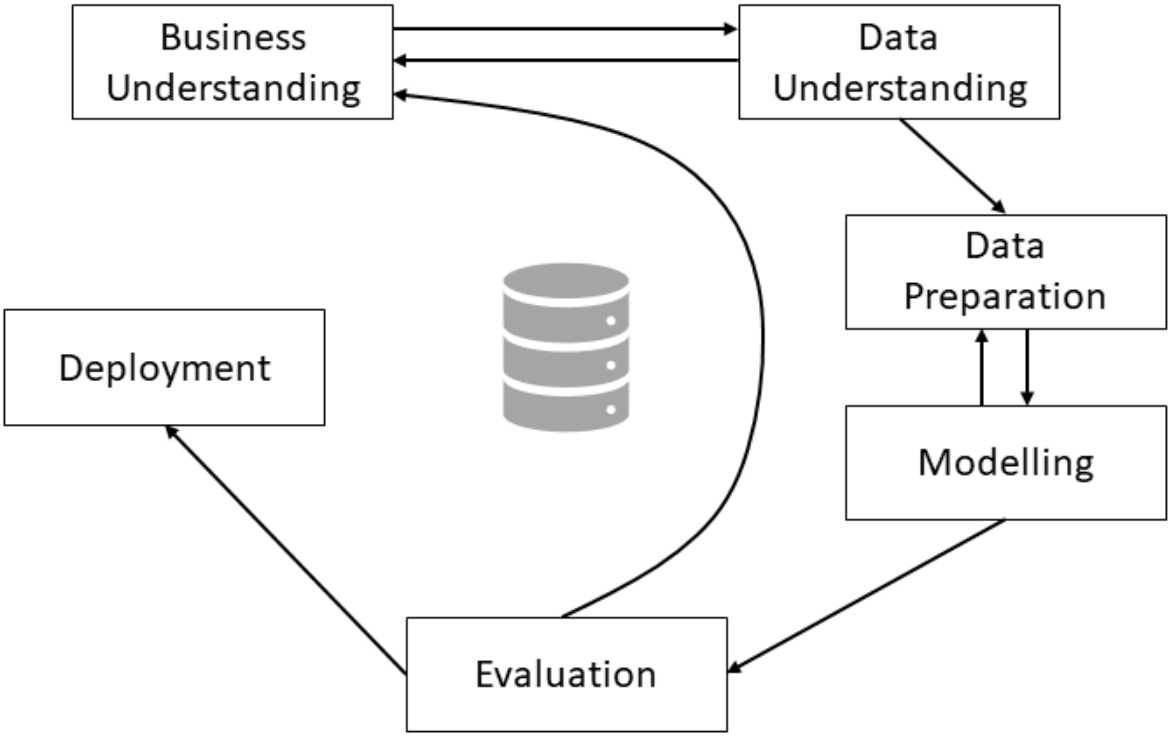

Abbildung 1.2: CRISP-DM Prozessmodells nach [3]

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.

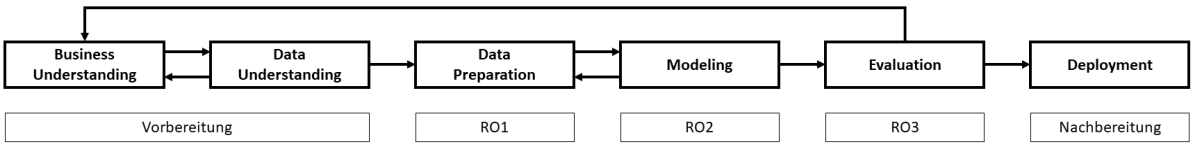

Abbildung 1.3: Phasen des CRISP-DM Prozessmodells nach [3] mit Zuordnung der Arbeitsphasen

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 zu-

grundeliegende 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).

1.3 Zeitplanung

ANPASSEN AN TATSÄCHLICHEN ABLAUF

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- und 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 / der 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 Bewertungsmetriken - Anwendung des Test-Sets - Berechnung der Bewertungsmetriken	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.

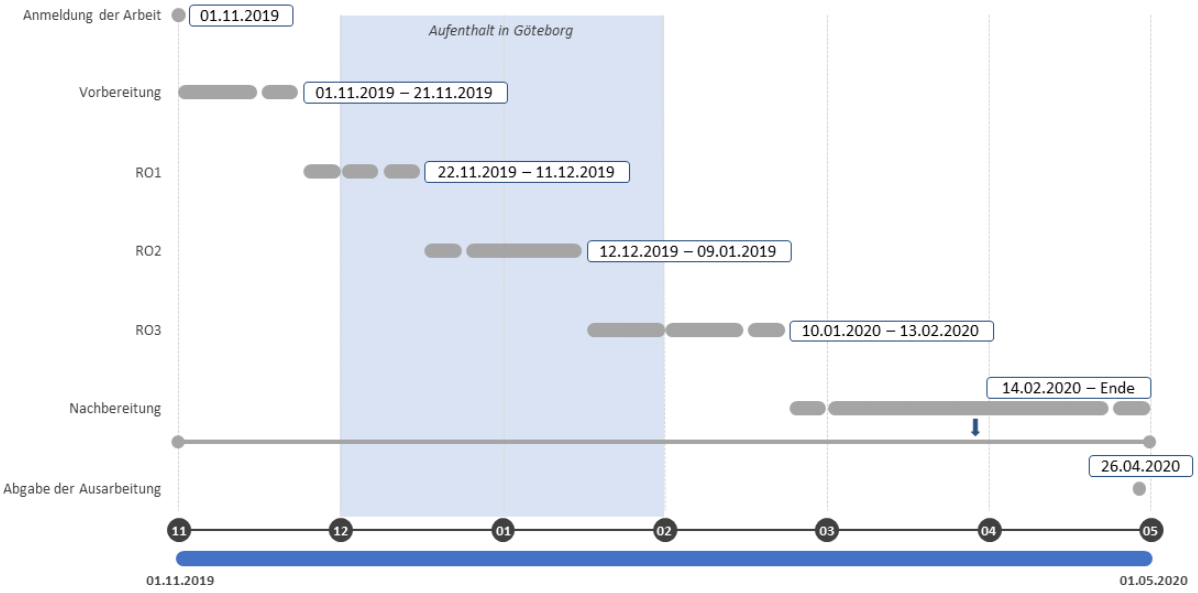

Abbildung 1.4: Zeitlicher Ablaufplan der Arbeit als Gantt-Chart

1.4 Aufbau der Arbeit

Kapitel 2

Hintergrund

Ausblick: Zum besseren Verständnis der weiteren Verlaufs dieser Arbeit, dient dieses Kapitel zur Einführung in die zugrundeliegenden Themen. Dazu wird zunächst die featurebasierte Softwareentwicklung erläutert, ehe dann der Themenbereich des Machine Learnings vorgestellt wird. Dazu werden die Klassifikation und die Fehlervorhersage mittels Machine Learning erläutert. Unterstützt werden die Abschnitte von Grafiken zum besseren Verständnis der Zusammenhänge.

2.1 Featurebasierte Softwareentwicklung

2.2 Machine-Learning-Klassifikation

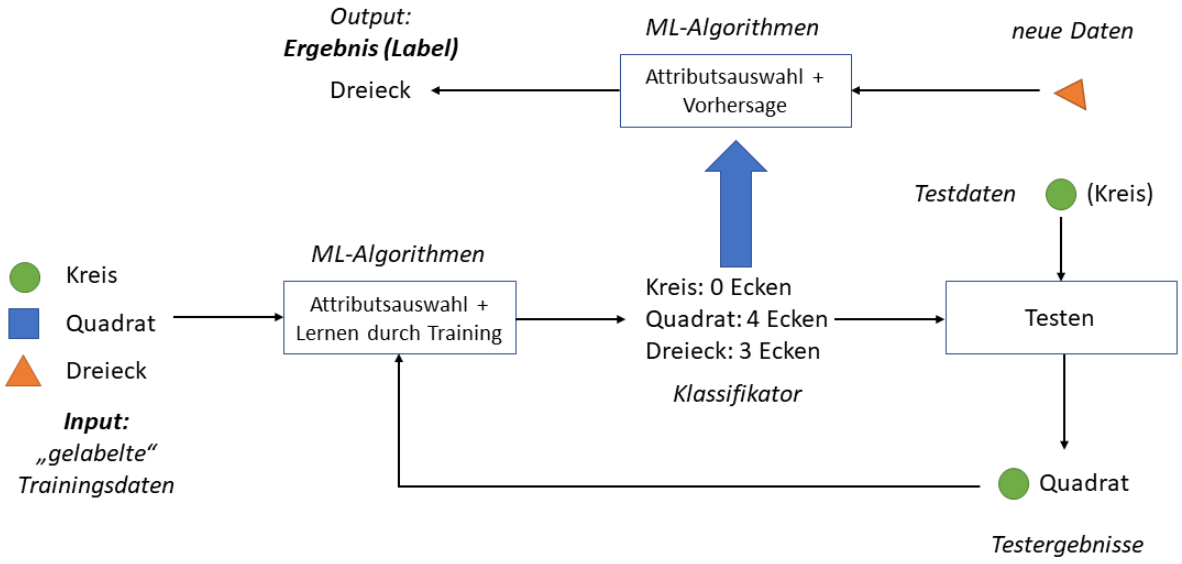

Abbildung 2.1: Allgemeiner Prozess des überwachten Machine Learnings

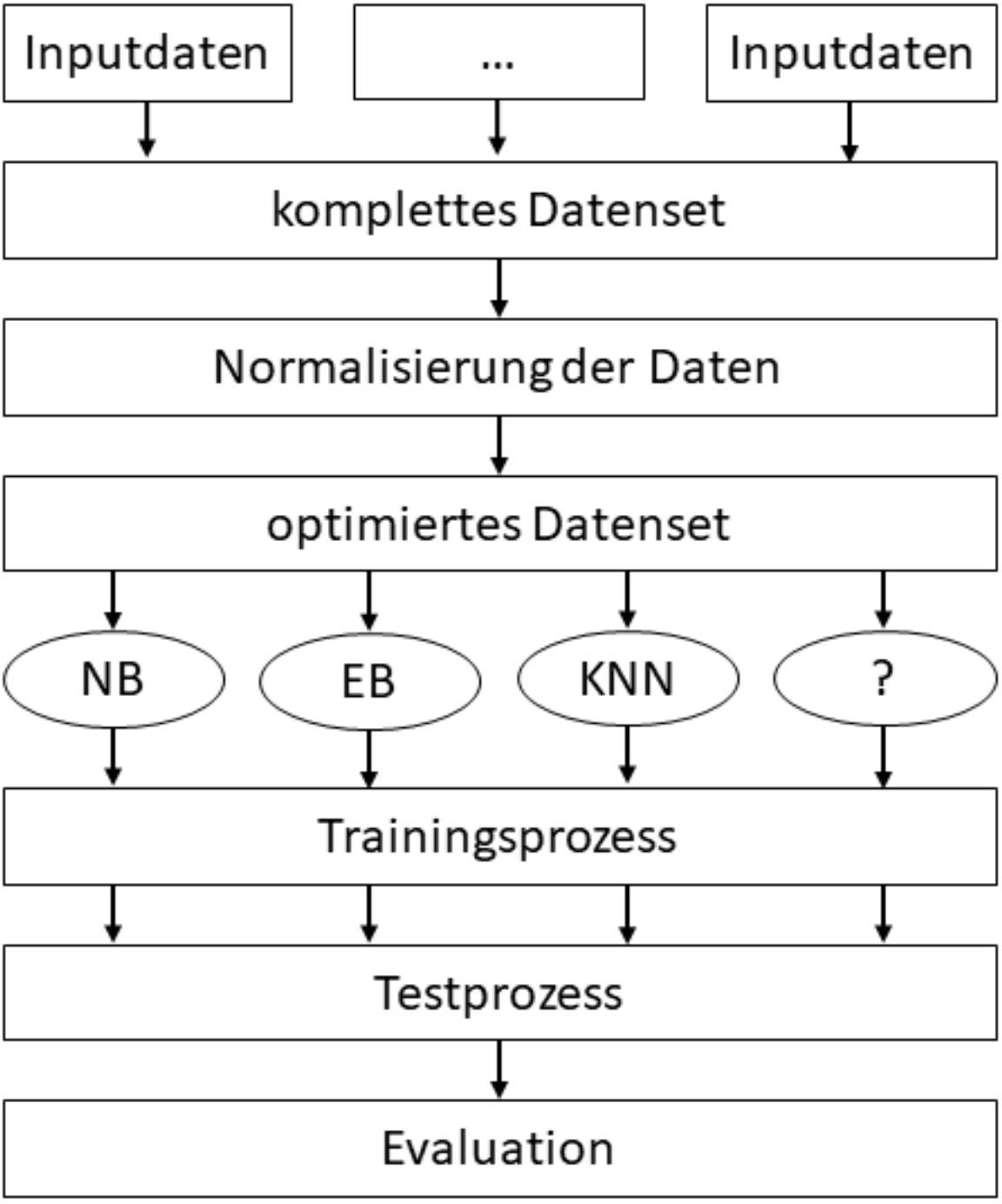

Abbildung 2.2: Angewendeter Prozess zur Durchführung der Klassfikation nach **[Ceylan2006]**

2.3 Fehlervorhersage mittels Machine Learning

Kapitel 3

Erstellung eines featurebasierten Datensets

Ausblick: Dieses Kapitel widmet sich der schrittweisen Erläuterung des Prozesses zur Erstellung des featurebasierten Datensets, welches zur Anlernung der Machine-Learning-Klassifikatoren dient. Dazu wird zunächst die Datenauswahl näher beleuchtet. Darauf folgt eine Darlegung der Konstruktion des Datensets sowie der Auswahl und Berechnung der Metriken, welche als Attribute (Features) im Rahmen der Anlernung der Klassifikatoren dienen.

3.1 Datenauswahl

Tabelle 3.1: Übersicht der zur Erstellung des Datensets verwendeten Tools mit zugehörigen Werten

	Zweck	Datenquelle	#Releases	#Commits	#Korrektiv	#Fehlereinführend	#Features
Blender	3D-Modellierungstool	GitHub-Mirror	11	19119	8258	1418	∼3000
Busybox	UNIX-Toolkit	Git-Repository	14	4984	1408	142	702
Emacs	Texteditor	GitHub-Mirror	7	12805	6959	685	863
GIMP	Bildbearbeitung	GitLab-Repository	14	7240	1703	272	1620
Gnumeric	Tabellenkalkulation	GitLab-Repository	8	6025	1591	136	725
gnuplot	Plotting-Tool	GitHub-Mirror	5	6619	880	1323	625
Irssi	IRC-Client	GitHub-Repository	7	253	77	1	17
libxml2	XML-Parser	GitLab-Repository	10	732	409	37	225
lighttpd	Webserver	Git-Repository	6	2597	1202	555	323
MPSolve	Polynomlöser	GitHub-Repository	8	668	158	69	130
Parrot	Virtuelle Maschine	GitHub-Repository	7	16245	3437	824	559
Vim	Texteditor	GitHub-Repository	7	9849	1033	2571	1227
xfig	Grafikeditor	Sourceforge-Repository	7	18	0	0	205

Tabelle 3.3: Übersicht des Schemas der SZZ-Tabellen des Datensets

Spaltenname	Beschreibung	Spaltenname	Beschreibung
name	Name des Softwareprojekts	filename	einem Commit zugehörige Dateien
commit_hash	Commit-Hashes fehlerbehebender Commits	bug_introducing	Auflistung der fehlereinführenden Commit-Hashes
filepath	einem Commit zugehörige Dateipfade		

3.2 Konstruktion des Datensets

Tabelle 3.2: Übersicht des Schemas der Haupttabellen des Datensets

Spaltenname	Beschreibung	Spaltenname	Beschreibung
name	Name des Softwareprojekts	lines_added	Anzahl der hinzugefügten Zeilen zur geänderten Datei
release_number	zugehörige Release-Version basierend auf vergebenen Tags	lines_removed	Anzahl der entfernten Zeilen von der geänderten Datei
commit_hash	eindeutiger Bezeichner eines Commits	change_type	Art der Änderung
commit_author	Autor eines Commits	diff	Diff der geänderten Datei
commit_msg	Nachricht eines Commits	corrective	Indikator, ob Commit fehlerbehebend war
filename	Name der geänderten Datei	bug_introducing	Indikator, ob Commit fehlereinführend war
nloc	„Lines of code“ der geänderten Datei	feature	Namen der zugehörigen Features der geänderten Datei
cycomplexity	Zyklomatische Komplexität der geänderten Datei		

3.3 Metriken

Ergänzen!!!!

Tabelle 3.4: Übersicht des Schemas der Metrics-Tabellen des Datensets

Spaltenname	Beschreibung	Spaltenname	Beschreibung
name	Name des Softwareprojekts	comm	Anzahl der Commits, die in einem Release dem betreffenden Feature gewidmet sind
release_number	zugehörige Release-Version basierend auf vergebenen Tags	adev	Anzahl der Entwickler, die das betreffende Feature in einem Release bearbeitet haben
feature	betreffendes Feature	ddev	kummulierte Anzahl der Entwickler, die das betreffende Feature in einem Release bearbeitet haben
...	...	..	...

Kapitel 4

Training von Machine Learning Klassifikatoren (TBD)

Ausblick: TBD.

4.1 Auswahl der Klassifikatoren (TBD)

4.2 Trainingsprozess (TBD)

Kapitel 5

Evaluation

Ausblick: Dieses Kapitel dient der Evaluation der im vorangegangenen Kapitel erläuterten Klassifikationen. Dies geschieht durch verschiedene Vergleichsmetriken, welche in diesem Kapitel vorgestellt werden. Ebenfalls umfasst dieses Kapitel einen Vergleich der Klassifikatoren zu nicht-featurebasierten Methoden und eine Erläuterung der Herausforderungen und Limitationen, die mit der Erarbeitung der vorangegangenen Kapitel einhergingen.

5.1 Herausforderungen und Limitationen

5.2 Vergleich der Klassifikatoren

5.2.1 Vergleichsmetriken

5.2.2 Ergebnisse (TBD)

5.3 Vergleich zu nicht-featurebasierten Methoden

Kapitel 6

Fazit

Ausblick: Das abschließende Kapitel dieser Arbeit dient zur Zusammenfassung der Ergebnisse der vorangegangenen Kapitel sowie zur Erläuterung der daraus gewonnenen Erkenntnisse. Ebenfalls wird ein Ausblick auf eine mögliche Weiterführung dieser Arbeit gegeben.

6.1 Zusammenfassung und Erkenntnisse

6.2 Ausblick

Literatur

[1] Sven Apel u. a. *Feature-Oriented Software Product Lines*. Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-37521-7.

[2] Venkata Udaya B. Challagulla u. a. „Empirical assessment of machine learning based software defect prediction techniques“. In: *International Journal on Artificial Intelligence Tools* 17.2 (2008), S. 389–400. ISSN: 02182130. DOI: 10.1142/S0218213008003947.

[3] Pete Chapman u. a. „CRISP-DM 1.0“. In: *CRISP-DM Consortium* (2000), S. 76. ISSN: 0957-4174. DOI: 10.1109/ICETET.2008.239.

[4] Le Son u. a. „Empirical Study of Software Defect Prediction: A Systematic Mapping“. In: *Symmetry* 11.2 (Feb. 2019), S. 212. DOI: 10.3390/sym11020212.

[5] Thomas Thüm u. a. „A Classification and Survey of Analysis Strategies for Software Product Lines“. In: *ACM Computing Surveys* 47.1 (Juni 2014), S. 1–45. DOI: 10.1145/2580950.

Anhang A

Test 1

Lorem ipsum

Anhang B

Test 2

Lorem ipsum