



## Seminararbeit im Studiengang IMIT (BSc)

# Codemetriken für Softwareprodutktlinien

Version 1.0 vom 25. November 2019
(Vor Abgabe entfernen)

**Tjark Harjes**

301249

harjes@uni-hildesheim.de

**Betreuer:**
Prof. Dr. Klaus Schmid, SSE
<Zweitpruefer>

Arbeitsgruppe Software Systems Engineering • Institut für Informatik
Universität Hildesheim • Universitätsplatz 1 • D-31134 Hildesheim

# Eigenständigkeitserklärung

**Erklärung über das selbstständige Verfassen von "Codemetriken für Softwareprodutktlinien"**

Ich versichere hiermit, dass ich die vorstehende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der obigen Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich in jedem Fall durch die Angabe der Quelle bzw. der Herkunft, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet und anderen elektronischen Text- und Datensammlungen und dergleichen. Die eingereichte Arbeit ist nicht anderweitig als Prüfungsleistung verwendet worden oder in deutscher oder einer anderen Sprache als Veröffentlichung erschienen. Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschung behandelt werden.

Hildesheim, den 25. November 2019

---

Tjark Harjes

## Kurzfassung

Eine kurze Zusammenfassung der Arbeit, die Interesse beim Leser wecken soll.

## Abstract

Gerne zusätzlich oder alternativ in Englisch.

# Inhaltsverzeichnis

# Abbildungsverzeichnis



# Tabellenverzeichnis

# Quellcode-Verzeichnis

# 1 Einleitung

## 1.1 Motivation

## 1.2 Ziel der Arbeit

## 1.3 Aufbau der Arbeit

# 2 Grundlagen von Code Metriken

**Kapitel**.

## 2.1 Erhebung von Code-Metriken

Das hier ist ein Abschnitt

## 2.2 (Un)abhängigkeit von Code-Metriken voneinander

## 2.3 Anwendungsfallabhängige Festlegung von Code-Metriken

# 3 Angreifbarkeit von Softwareproduktlinien

Die Angreifbarkeit von Software und deren Produktlinien beschreibt welche Sicherheitslücken und Probleme aus dem geschriebenen Programmcode hervorgehen.

Der Zusammenhang von Schwachstellen im Code kann durch unterschiedliche Eigenschaften des Codes bestimmt werden. Sie sind durch geeignete Metriken zu evaluieren und können Aufschluss über den Status der Angreifbarkeit des Codes geben.

Eine Eigenschaft, die im Zusammenhang mit Schwachstellen im Code steht, ist die Komplexität des programmierten Quellcodes [Ape16].

## 3.1 Code-Metriken zur Evaluierung der Konfigurationskomplexität

Um die Komplexität von Quellcode kann laut einer Studie von Ferreira et al. durch mehrere Code-Metriken untersucht werden, die sich auf die Häufigkeit des Vorkommens von Konfigurationsanweisungen beziehen:

- Interne ifdef-Anweisungen
- Interne Konfigurationsoptionen
- Externe Konfigurationsoptionen

### 3.1.1 Menge an internen Ifdef-Anweisungen

Die referenzierte Studie nutzt die Menge Ifdef-Anweisungen, um die Konfigurationskomplexität zu bestimmen. Ifdef-Anweisungen sind Codeblöcke, die im Linux-Kernel, ähnlich wie if-Statements, eingesetzt werden. Sie dienen der Anpassung der Software fÃ¼r verschiedene Produktlinien.

Umso mehr verschiedene Features gleichzeitig programmiert werden, desto schwieriger ist es für den Entwickler den Programmfluss zu verstehen.

### 3.1.2 Menge an internen Konfigurationsoptionen

Die Menge an verschiedenen Konfigurationsoptionen in einer Funktion könnte einen Einfluss auf die Komplexität der des Codes haben. Je nachdem wie viele Optionen zur Verfügung stehen, muss ein Entwickler diese auch während der Veränderung jener Funktion bedenken.

### 3.1.3 Menge an externe Konfigurationsoptionen

Die Komplexität von Quellcode in Softwareproduktlinien kann auch anhand der Häufigkeit des Vorkommens von externen Konfigurationsoptionen untersucht werden(vgl. [Ape16]). Externe Konfigurationsoptionen bezeichnen die Wahrscheinlichkeit inwiefern eine ganze Funktion Teil einer Konfiguration ist. Hierbei sollte eine hÃ¶here Anzahl an, zur Aktivierung benötigten, Features die Häufigkeit der Verwendung in Produktlinien vermindern.

## 3.2 Code-Metriken zur Evaluierung von Angreifbarkeit

# 4 Lesbarkeit von Softwareproduktlinien

## 4.1 Code-Metriken zur Evaluierung der Lesbarkeit von Code

# 5 Wartbarkeit von Softwareproduktlinien

## 5.1 Code-Metriken zur Evaluierung der Wartbarkeit von Code

# 6 Bezug zwischen Angreifbarkeit, Wartbarkeit und Lesbarkeit

In diesem Kapitel wird kurz aufgezeigt, welche Gliederungsebenen es gibt. Hier befinden wir uns auf der Ebene eines **Kapitels**.

## 6.1 Einfluss von Lesbarkeit auf Wartbarkeit

### 6.1.1 Einfluss der Code-Metriken aufeinander

### 6.1.2 Negatives Zusammenspiel zwischen einzelnen Code-Metriken

## 6.2 Verbindung zwischen Wartbarkeit/Lesbarkeit und Angreifbarkeit

### 6.2.1 Einfluss der Code-Metriken aufeinander

### 6.2.2 Negatives Zusammenspiel zwischen einzelnen Code-Metriken

# 7 Schlussbemerkungen

FließtextFließtextFließtextFließtext

## 7.1 Fazit

FließtextFließtextFließtextFließtext

## 7.2 Ausblick

FließtextFließtextFließtextFließtext

# A Anhang

## A.1 Beispiele

Zitat ohne Seitenangabe: [BKPS04]

Zitat mit Seitenangabe: [BKPS04, S.1]

Referenz eines Glossareintrags: Computer

Eine normale Liste:

- Ein Punkt
- Ein anderer Punkt

Eine nummerierte Liste:

1. Erstens
2. Zweitens
3. ...

Eine Tabelle:

Tabelle A.1: Eine Tabelle

| Spalte 1 | Spalte 2 | Spalte 3 |
|---|---|---|
| 1 | 2 | 3 |

**Fetter Text**

*Kursiver Text*

Eine Referenz: Siehe Tabelle A.1: Eine Tabelle auf Seite 8.

Ein Bild:


Abbildung A.1: Das Logo der SUH

Eine abgesetzte Formel:

$$\sum_{n=0}^{3} n = 6 \tag{A.1}$$

Eine Formel im Fließtext: $2^2 = 4$

Ein Listing aus einer Datei:

```
1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Hello World!");
4     }
5 }
```

Listing A.1: HelloWorld

Ein 'on-the-fly' erstelltes Listing:

```
1 (Sun Sep 13 23:02:20 2009): ODBC Driver <system32>\wbemdr32.dll not present
2 (Sun Sep 13 23:02:20 2009): Successfully verified WBEM OBDC adapter (incompatible version
      removed if it was detected).
3 (Sun Sep 13 23:02:20 2009): Wbemupgd.dll Registration completed.
```

Listing A.2: Beispiel eines Log-Eintrags

# Glossar

**Computer** ist ein elektronisches Gerät, das Daten verarbeitet.

# Literaturverzeichnis

[Ape16] Apel, Gabriel Ferreira; Momin Malik;Christian Kästner;Jürgen P. (Hrsg.): *Do ifdefs Influence the Occurrence of Vulnerabilities? An Empirical Study of the Linux Kernel.* 2016

[BKPS04] Böckle, Günther ; Knauber, Peter ; Pohl, Klaus ; Schmid, Klaus: *Software-Produktlinien.* Dpunkt-Verlag, 2004