

258321 DKE Projekt

**Gruppe 2**

**Teammitglieder:**

k01607605, Aistleithner Andrea

k01256561, Dusanic Maja

k01356577, Teuchtmann Alexander

k01356229, Tomic Milos

KONzeptueller Entwurf

Inhaltsverzeichnis

[1. Systemkomponenten 2](#_Toc530668658)

[1.1. Evaluierungsprogramm 2](#_Toc530668659)

[1.2. Datengenerator Rule Model Inheritance 3](#_Toc530668660)

[1.3. Datengenerator CBR 3](#_Toc530668661)

[1.4. Externer Speicher 3](#_Toc530668662)

[2. Schnittstellen 4](#_Toc530668663)

[2.1. User Schnittstelle 5](#_Toc530668664)

[2.2. Rule Model Inheritance Datengenerierung Schnittstelle 5](#_Toc530668665)

[2.3. CBR Datengenerierung Schnittstelle 5](#_Toc530668666)

[2.4. Vadalog Schnittstelle 5](#_Toc530668667)

[2.5. Speicherung Schnittstelle 5](#_Toc530668668)

[3. Umsetzung 6](#_Toc530668669)

[4. Abbildungsverzeichnis 7](#_Toc530668670)

# Systemkomponenten

Das System des Evaluierungsframeworks wird in Abbildung 1 graphisch dargestellt. Die Komponenten im grün umrahmten Bereich, sind die zu implementierenden Komponenten.



Abbildung 1: Systemkomponenten

Im folgenden Abschnitt werden die in der Abbildung 1 dargestellten Komponenten genauer beschrieben. Ebenso wird die geplante Implementierung der Komponenten erläutert und schriftlich dargestellt.

## Evaluierungsprogramm

Die Komponente des Evaluierungsprogramms steht im Mittelpunkt des Systems und agiert als User Interface mit dem Benutzer. Die grafische Darstellung vom User Interface wird als unwichtig eingestuft, weil dies nicht zum Ziel dieses Projekts gehört.

Das Eingeben der Daten vom Benutzer wird über die Konsole erfolgen. Der Benutzer agiert mit der Komponente des Evaluierungsprogrammes über die User Schnittstelle, welche im Abschnitt 2.1 genauer erläutert wird.

Die vom Benutzer eingegeben Daten, über die gewünschten Testfälle, werden im Evaluierungsprogramm erfasst und über eine der Datengeneratoren Schnittstellen an den jeweils betroffenen Datengenerator weitergegeben.

Die vom Datengenerator erzeugten Daten werden zurück an das Evaluierungsprogramm übermittelt, über die betroffene Datengenerator Schnittstelle.

Anschließend übergibt das Evaluierungsprogramm den Code (bzw. die Testfälle), der vom Datengenerator erzeugte wurde, über die Vadalog Schnittstelle, welche im Abschnitt 2.4 genauer beschrieben wird, an die von der Universität zur Verfügung gestellte Vadalog Ausführung, damit diese den Code (bzw. die Testfälle) ausführt.

Dabei wird ein Zeitobjekt erzeugt, welche den Startzeitpunkt des Tests festhalten soll. Nach Abschluss der Durchführung des Testfalles, wird von der Vadalog Schnittstelle ein Ergebnis zurückgegeben. Hier wird wieder ein Zeitobjekt erzeugt, um den Endzeitpunkt des Tests festzuhalten.

Durch die Differenz des Startzeitpunktes und des Endzeitpunktes kann ermittelt werden, wie lange die Durchführung des Tests dauerte.

Die vom Benutzer eingegeben Daten und auch die Durchführungszeit des Testfalles wird vom Evaluierungsprogramm an die Schnittstelle zur Datenspeicherung übergeben. Diese Schnittstelle wird in Abschnitt 2.4 genauer erläutert.

## Datengenerator Rule Model Inheritance

Wünscht der Benutzer Testfälle betreffend der Rule Model Inheritance, erhält der Datengenerator vom Evaluierungsprogramm über seine zugehörige Schnittstelle die Informationen, wie viele Testdaten er generieren soll. Diese geforderten Testdaten werden vom Datengenerator generiert.

Die Testdaten werden vom Generator erzeugt und der fertige Test Code wird über die Schnittstelle wieder zurück an das Evaluierungsprogramm übergeben.

## Datengenerator CBR

Wünscht der Benutzer Testfälle betreffend CBR, erhält der Datengenerator vom Evaluierungsprogramm über seine zugehörige Schnittstelle die Informationen, wie viele Testdaten er generieren soll. Diese geforderten Testdaten werden vom Datengenerator generiert.

Die Testdaten werden vom Generator erzeugt und der fertige Test Code wird über die Schnittstelle wieder zurück an das Evaluierungsprogramm übergeben.

## Externer Speicher

Im externen Speicher werden die Ergebnisse der Performanceevaluierung gespeichert. Zusätzlich werden die vom Benutzer geforderten Testdaten hier persistiert, um die Tests nachvollziehen zu können und gegebenenfalls die Testergebnisse für spätere Evaluierungsarbeiten wiederverwenden zu können. Somit besteht auch die Möglichkeit, nicht nur ein Testergebnis mehrerer Durchläufe zu analysieren, sondern auch zeitlich unabhängige Tests untereinander vergleichen zu können.

Bei dem externen Speicher handelt es sich um eine mySQL Datenbank (Version 5,7), welche von einem externen Hoster zur Verfügung gestellt wird, und somit grundsätzlich dauerhaft verfügbar (99,9% Verfügbarkeit) ist. Dadurch ist es möglich, mithilfe des Evaluierungsprogramms die gewünschten Daten extern zu speichern, um sie persistent zu erhalten, wenn das Programm geschlossen wird. Durch die Datenbank ist ebenso der Mehrbenutzerbetrieb, sowie eine programmunabhängige Speicherung der Daten möglich. Die Einrichtung und Bearbeitung der Datenbank und Tabellen erfolgt über das phpMyAdmin Interface (Version 4.8.3) des Hosters.

Adresse: <https://e42776-phpmyadmin.services.easyname.eu>

Server: e42776-mysql.services.easyname.eu

Benutzer: u48005db23

Datenbank: u48005db23

Passwort: prdke2018

Die Daten werden über die Speicherung Schnittstelle, genauer behandelt in Abschnitt 2.5, vom Evaluierungsprogramm an den Speicher übergeben.

# Schnittstellen

Um es dem Programm zu ermöglichen, mit den einzelnen Komponenten zu kommunizieren, sind folgende Schnittstellen notwendig und zu implementieren. Die benötigten Schnittstellen sind in der folgenden Grafik, in Abbildung 2, orange-rot umrahmt.



Abbildung 2: Schnittstellen

Um eine funktionierende Kommunikation zwischen den Systemkomponenten zu garantieren, werden die in Abbildung 2 aufgezeigten Systemschnittstellen implementiert. Genauer beschrieben werden die notwendigen Schnittstellen im nachfolgenden Abschnitt.

## User Schnittstelle

Die User Schnittstelle ist die Verbindung zwischen dem Benutzer des Programmes und des Evaluierungsprogramm, dem Mittelpunkt des Systems. Die Eingabe vom Benutzer erfolgt über die Konsole, wobei die grafische Darstellung vom User Interface nicht zum Ziel gehört.

Der User wählt aus, welche Art von Testfälle er durchführen möchte.

Die Kommunikation zwischen User und Programm erfolgt über die Konsole. Die Konsole gibt dem User die Auswahl, welche Testfälle dieser durchführen möchte:

1 CBR

2 Rule Model Inheritance

Der User gibt die Zahl, als Integer Wert, ein, welche Art von Testfälle dieser durchführen möchte.

Basierend auf der Auswahl des Nutzers, verläuft das Programm wie in den folgenden zwei Abschnitten beschrieben.

### CBR

Anzahl Parameter

Contexte Prozentsatz vom Parameter

Parameter Values sollen sich ungefähr mit Anzahl der Parameter decken

Anzahl Business Cases

### Rule Model Inheritance

Hat der User als gewünschte Art der Testfälle 2 Rule Model Inheritance gewählt, wird dieser erneut nach einer Auswahl gefragt, welche der Rule Model Inheritance durchgeführt werden soll.

1 abstractionOnly

2 conformanceOnly

3 dynamicBehavioralDetectionOnly

4 inheritanceOnly

5 staticBehaviouralDetectionOnly

6 structuralDetectionOnly

Der User gibt die Zahl, als Integer Wert, ein, welche Art von Testfälle dieser durchführen möchte.

Basierend auf der Auswahl des Users, verläuft das Programm wie in den folgenden Abschnitten beschrieben. Die vom User getätigten Eingaben in den folgenden Abschnitten werden anschließend an das Evaluierungsprogramm übergeben.

#### abstractionOnly

Um diese Art der Testfälle durchzuführen wird als Input ein Module gefordert, für das alle Vererbungsbeziehungen aufgelöst wurden – Resolved Module.

Hierfür wird eine bestimmte Anzahl an Regeln und Fakten gefordert, welche der User als Zahl, Integer, eingeben soll.

Rules: (Integer Eingabe User)

Facts: (Integer Eingabe User)

#### conformanceOnly

Um diese Art der Testfälle durchzuführen werden als Input erkannte/gemeldete Änderungsvorgänge und Modifikationsbeschränkungen gefordert – detected/reported modification and operations modification restrictions.

Hierfür wird eine bestimmte Anzahl an Regeln und Fakten gefordert, welche der User als Zahl, Integer, eingeben soll.

Rules: (Integer Eingabe User)

Facts: (Integer Eingabe User)

#### dynamicBehavioralDetectionOnly

Um diese Art der Testfälle durchzuführen werden als Input Meta-Repräsentationen von Resultset-Paaren (Eltern- und Child-Resultsets) und Vererbungsbeziehungen von Modulen, für die Ergebnismengen angegeben sind gefordert – Meta-representations of resultset pairs (parent and child resultsets) and inheritance relations of modules for which resultsets are given.

Hierfür wird eine bestimmte Anzahl an Regeln und Fakten gefordert, welche der User als Zahl, Integer, eingeben soll.

Rules: (Integer Eingabe User)

Facts: (Integer Eingabe User)

#### inheritanceOnly

Um diese Art der Testfälle durchzuführen werden als Input ein Modul, das abgeleitet werden soll (toDerive) eine Modul-Metadarstellungen des relevanten Vererbungsbaums gefordert – module to be derived (toDerive) and module meta-representations of relevant inheritance tree.

Hierfür wird eine bestimmte Anzahl an Regeln und Fakten gefordert, welche der User als Zahl, Integer, eingeben soll.

Rules: (Integer Eingabe User)

Facts: (Integer Eingabe User)

#### staticBehaviouralDetectionOnly

Um diese Art von Testfälle durchzuführen wird als Input eine Meta-Repräsentationen von Modulen im Vererbungs-Teilbaum gefordert – Meta-representations of modules in inheritance subtree.

Hierfür wird eine bestimmte Anzahl an Regeln und Fakten gefordert, welche der User als Zahl, Integer, eingeben soll.

Rules: (Integer Eingabe User)

Facts: (Integer Eingabe User)

#### structuralDetectionOnly

Um diese Art von Testfälle durchzuführen wird als Input eine Metadarstellungen des relevanten Modulvererbungsbaums gefordert – Meta-representations of relevant module inheritance tree.

Hierfür wird eine bestimmte Anzahl an Regeln und Fakten gefordert, welche der User als Zahl, Integer, eingeben soll.

Rules: (Integer Eingabe User)

Facts: (Integer Eingabe User)

## Rule Model Inheritance Datengenerierung Schnittstelle

Diese Schnittstelle dient als Verbindung zwischen dem Rule Model Inheritance Datengenerator und des Evaluierungsprogramms. Die vom Generator generierten Daten werden über diese Schnittstelle an das Evaluierungsprogramm übergeben.

## CBR Datengenerierung Schnittstelle

Die Schnittstelle für die CBR Datengenerierung ist die Verbindung zwischen dem CBR Datengenerator und dem Evaluierungsprogramm.

## Vadalog Schnittstelle

Die Vadalog Schnittstelle ist die Verbindung zur Ausführung des generierten Codes. Diese Schnittstelle, wird von der Universität zur Verfügung gestellt.

Der generierte Code wird zur Ausführung übergeben, nach Beendigung der Durchführung des Tests erhält das Evaluierungsprogramm über diese Schnittstelle eine Information darüber und kann somit den Endzeitpunkt des Testfalles feststellen und damit die Durchführungsdauer des Testfalles ermitteln.

Bei der Implementierung unseres Programms wird ein Dummy anstatt der Vadalog Schnittstelle implementiert.

Im Programm wird eine Dummy Methode callVadalog() implementiert. Diese gibt die Performancewerte der Code Ausführung zurück.

Im speziellen werden folgende Werte zurückgegeben:

* Die Ausführungsdauer in Sekunden
  + Wird durch einen Random Zahlengenerator erzeugt
  + Datentyp: Double
* Die Anzahl der Fehler
  + Werden durch einen Random Zahlengenerator erzeugt
  + Datentyp: Integer
* Die durchschnittliche Anzahl der Fehler pro Sekunde
  + Die Gesamtanzahl der Fehler wird durch die Ausführungszeit dividiert
  + Datentyp: Double
* Die durchschnittliche Auslastung des Prozessors in Prozent
  + Wird durch einen Random Zahlengenerator erzeugt
  + Datentyp: Double

## Speicherung Schnittstelle

Mithilfe der Speicherung Schnittstelle werden die Performance Ergebnisse vom Evaluierungsprogramm an den externen Speicher übergeben.

Diese Verbindung wird durch eine JDBC Verbindung (Java Database Connectivity) ermöglicht. Die zu speichernden Daten werden mithilfe von JDBC in der SQL Datenbank des externen Speichers persistent erhalten.

# Umsetzung

Die Umsetzung der Anforderungen erfolgt mit der Programmiersprache Java JDK 8 (oder durch eine neuere Version). Als Entwicklungsumgebung wird Eclipse Oxygen herangezogen. Zur Unterstützung der Umsetzung der Datengeneratoren und des Evaluierungsframeworks werden keine Frameworks benutzt.

Die Datenspeicherung wird mithilfe einer mySQL Datenbank ermöglicht und die Verbindung zwischen dem Java Programm und des Evaluierungsframeworks wird mithilfe von JDBC umgesetzt, um die zu speichernden Daten an die folgende Datenbank zu übertragen:

Adresse: <https://e42776-phpmyadmin.services.easyname.eu>

Server: e42776-mysql.services.easyname.eu

Benutzer: u48005db23

Datenbank: u48005db23

Passwort: prdke2018

# Abbildungsverzeichnis

[Abbildung 1: Systemkomponenten 2](file:///C:\Users\Anwender\Documents\GitHub\258321_DKE_PR\Dokumente\Konzeptueller%20Entwurf\Konzeptueller_Entwurf.docx#_Toc530668646)

[Abbildung 2: Schnittstellen 4](file:///C:\Users\Anwender\Documents\GitHub\258321_DKE_PR\Dokumente\Konzeptueller%20Entwurf\Konzeptueller_Entwurf.docx#_Toc530668647)