

258321 DKE Projekt

**Gruppe 2**

**Teammitglieder:**

k01607605, Aistleithner Andrea

k01256561, Dusanic Maja

k01356577, Teuchtmann Alexander

k01356229, Tomic Milos

MeilenStein Implementierung

Inhaltsverzeichnis

[1. Programmarchitektur 2](#_Toc7505)

[2. Funktionalitäten einzelner Klassen 2](#_Toc7506)

[3. Beschreibung Zusammenspiel der Klassen 2](#_Toc7507)

[4. Tests der geforderten Funktionalitäten 2](#_Toc7508)

[5. Abhängigkeiten Bibliotheken 2](#_Toc7509)

[6. Installationsanleitung 2](#_Toc7510)

[7. Limitierungen und Verbesserungsvorschläge im praktischen Einsatz 2](#_Toc7511)

[8. Abbildungsverzeichnis 3](#_Toc7512)

[9. Tabellenverzeichnis 3](#_Toc7513)

# Programmarchitektur

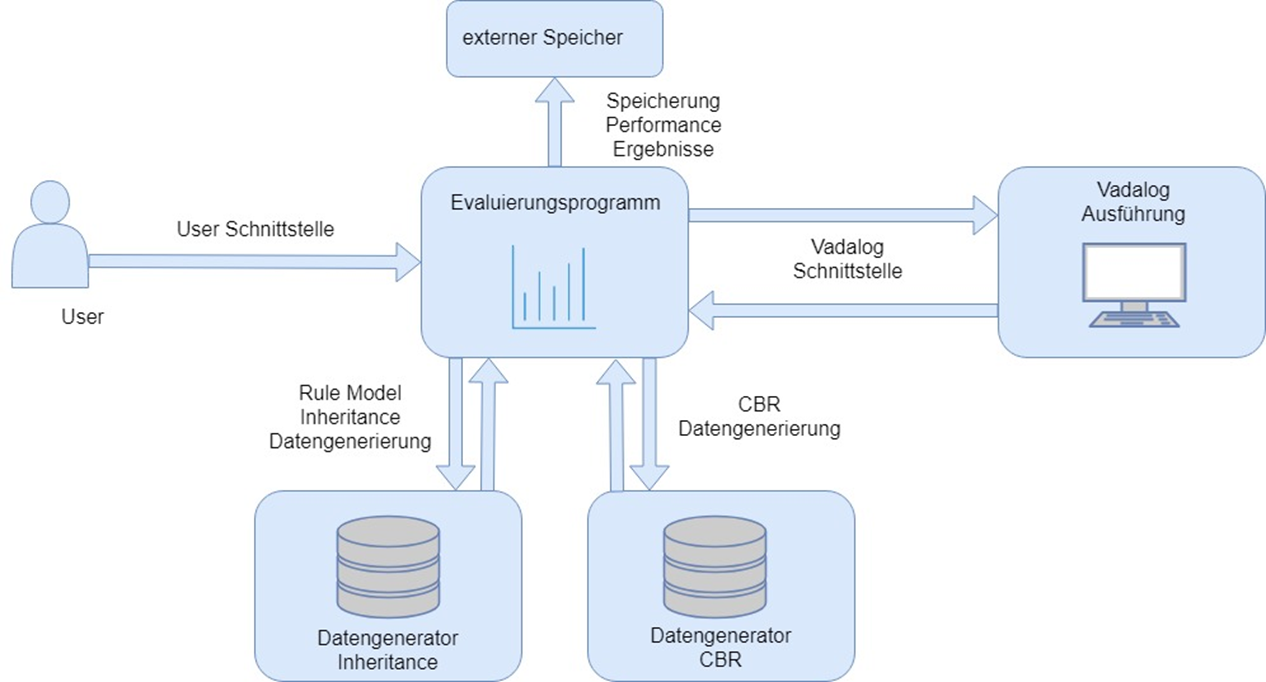


Abbildung 1: Programmarchitektur

In der oben abgebildeten Grafik, Abbildung 1, ist die Architektur des Programmes abgebildet. Diese Grafik wurde schon beim Konzeptuellen Entwurf erstellt und nurmehr leicht abgeändert, die essentiellen Komponenten blieben aber genauso erhalten.

## Komponenten

Das Programm setzt sich aus folgenden Komponenten zusammen:

**Das Evaluierungsprogramm**

Diese Komponente beinhaltet das ausführbare Hauptprogramm und ist der Mittelpunkt des Programmes, worüber die anderen Komponenten miteinander verbunden werden. Der User agiert über die User Schnittstelle mit dem Evaluierungsprogramm und übergibt die Input Werte zur Datengenerierung.

**Der Rule Model Inheritance Datengenerator**

Diese Komponente generiert den Rule Model Inheritance Code mithilfe der Input Werte des Users, welche über das Evaluierungsprogramm an den Datengenerator weitergegeben werden. Die fertig generierten Code Daten werden vom Generator zurück an die Komponente des Evaluierungsprogrammes übergeben.

**Der CBR Datengenerator**

Diese Komponente generiert den CBR Code mithilfe der Input Werte des Users, welche über das Evaluierungsprogramm an den Datengenerator weitergegeben werden. Die fertig generierten Code Daten werden vom Generator zurück an die Komponente des Evaluierungsprogrammes übergeben.

**Die Dummy Vadalog Ausführung**

Nachdem der fertig generierte Test Code vom Generator an das Evaluierungsprogramm zurückgegeben wurde, ruft die Komponente Evaluierungsprogramm die Dummy Vadalog Ausführung auf. Diese Komponente simuliert die Vadalog Ausführung und generiert Zufalls-Evaluierungswerte, die Execution Time, ob Fehler aufgetretenen sind und wie viel CPU Leistung bei der Ausführung verbraucht wurde. Diese Werte werden zurück an das Evaluierungsprogramm gegeben.

Der generierte Test Code wird in ein Text File geschrieben, welches generiert wird, da die Konsole schnell überläuft und lange Codes nicht vollständig ausgeben kann. Ebenso werden die Input Werte und die Evaluierungsergebnisse in dieses Text File geschrieben.

**Der externe Speicher, die Datenbank**

Nachdem die Testdaten generiert und die Ausführung durchgeführt werden, übergibt das Evaluierungsprogramm die zu speichernden Werte an die Datenbank. Hierbei werden die Input Werte, die Ergebnisse der Tests und das Datum und die Uhrzeit der Durchführung gespeichert. Der generierte Code wird nicht in der Datenbank gespeichert, sondern nur ausgegeben.

## Schnittstellen

Folgende Schnittstellen lassen die Komponenten des Programmes miteinander kommunizieren:

**Die User Schnittstelle**

Diese Schnittstelle lässt den User die Input Werte an das Evaluierungsprogramm übergeben.

**Die Rule Model Inheritance Schnittstelle**

Diese Schnittstelle erlaubt es, die Input Werte vom Evaluierungsprogramm an den Rule Model Inheritance Datengenerator zu übergeben und die generierte Test Codes wieder zurück an das Evaluierungsprogramm zu übergeben.

**Die CBR Schnittstelle**

Diese Schnittstelle erlaubt es, die Input Werte vom Evaluierungsprogramm an den CBR Datengenerator zu übergeben und die generierte Test Codes wieder zurück an das Evaluierungsprogramm zu übergeben.

**Die Vadalog Schnittstelle**

Die Vadalog Schnittstelle ermöglicht die Kommunikation zwischen Evaluierungsprogramm und der Dummy Vadalog Ausführung.

**Die Speicherung Schnittstelle**

Über die Speicherung Schnittstelle werden die zu speichernden Daten an die Datenbank übergeben.

## Umsetzung im Programm

Abbildung 2: Package Struktur im Programm

In Abbildung 2 ist die Struktur der Packages zu sehen und wie die geplanten Komponenten im Programm umgesetzt wurden.

Das Package **DataGeneratorCBR** setzt die Funktionalität der Komponente **CBR Generator** um.

Das Package **GeneratorRandomString** enthält die Funktionalität, dass ein zufallsgenerierter String aus zufällig zusammengestetzten Buchstaben generiert wird. Dies wird zur Unterstützung der Datengeneratoren eingesetzt.

Das Package **DataGeneratorRuleModelInheritance** setzt die Komponente **Rule Model Inheritance** **Generator** um.

Das Package **DB** erfüllt die Aufgabe der Datenspeicherung und setzt die Komponente **externer Speicher** um.

Das Package **EvaluationFramework** erfüllt die Aufgabe der Komponente des **Evaluierungs Frameworks.**

Das Package **Exceptions** dient zur Unterstützung, dass bei der Eingabe der Input Werte keine negativen Zahlen eingegeben werden können.

Das Package **Models** dient zur Unterstützung der Datenspeicherung. Darin befinden sich zwei Klassen, die ein CBR Objekt oder ein RMI Objekt erstellen können. Dies erleichtert die Einträge in die Datenbank.

In dem Package **Test** wurden während dem Implementieren die Datengeneratoren getestet.

Das Package **Vadalog** übernimmt die Aufgabe der Komponente **Vadalog Ausführung**.

Das Text File **out.txt** ist die Ausgabe des generierten Codes, der Input Werte und der Evaluierungsergebnisse.

# Funktionalitäten einzelner Klassen

Die einzelnen Packages, die in Abbildung 2 zu sehen sind, und deren dazugehörigen Klassen werden nun in diesem Abschnitt genauer beschrieben. Es werden die Funktionalität und das Zusammenspiel der Klassen beschrieben.

## DataGeneratorCBR

Abbildung 3: Package DataGeneratorCBR

In Abbildung 3 sind die Klassen des Packages DataGeneratorCBR zu sehen. Diese Klassen werden im folgenden genauer hinsichtlich ihrer Funktionalität und ihrem Zusammenspiel beschrieben.

### BusinessCase

### BusinessCaseClass

### Context

### ContextClass

### GeneratorCBR

### Module

### Parameter

### ParameterValue

## DataGeneratorRandomString

## DataGeneratorRMI

## DB

## EvaluationFramework

## Exceptions

## Models

## Test

## Vadalog

# Tests der geforderten Funktionalitäten

# Abhängigkeiten Bibliotheken

# Installationsanleitung

# Limitierungen und Verbesserungsvorschläge im praktischen Einsatz

# Abbildungsverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

# Tabellenverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**