

258321 DKE Projekt

**Gruppe 2**

**Teammitglieder:**

k01607605, Aistleithner Andrea

k01256561, Dusanic Maja

k01356577, Teuchtmann Alexander

k01356229, Tomic Milos

MeilenStein Implementierung

Inhaltsverzeichnis

[1. Programmarchitektur 2](#_Toc19406)

[1.1. Komponenten 2](#_Toc19407)

[1.2. Schnittstellen 3](#_Toc19408)

[1.3. Umsetzung im Programm 4](#_Toc19409)

[2. Funktionalitäten einzelner Klassen 4](#_Toc19410)

[2.1. DataGeneratorCBR 5](#_Toc19411)

[2.1.1. BusinessCase 5](#_Toc19412)

[2.1.2. BusinessCaseClass 5](#_Toc19413)

[2.1.3. Context 6](#_Toc19414)

[2.1.4. ContextClass 6](#_Toc19415)

[2.1.5. GeneratorCBR 6](#_Toc19416)

[2.1.6. Module 7](#_Toc19417)

[2.1.7. Parameter 8](#_Toc19418)

[2.1.8. ParameterValue 8](#_Toc19419)

[2.2. DataGeneratorRandomString 8](#_Toc19420)

[2.3. DataGeneratorRMI 8](#_Toc19421)

[2.4. DB 8](#_Toc19422)

[2.5. EvaluationFramework 8](#_Toc19423)

[2.6. Exceptions 8](#_Toc19424)

[2.7. Models 9](#_Toc19425)

[2.8. Test 9](#_Toc19426)

[2.9. Vadalog 9](#_Toc19427)

[3. Tests der geforderten Funktionalitäten 9](#_Toc19428)

[4. Abhängigkeiten Bibliotheken 9](#_Toc19429)

[5. Installationsanleitung 9](#_Toc19430)

[6. Limitierungen und Verbesserungsvorschläge im praktischen Einsatz 9](#_Toc19431)

[7. Abbildungsverzeichnis 10](#_Toc19432)

[8. Tabellenverzeichnis 10](#_Toc19433)

# Programmarchitektur

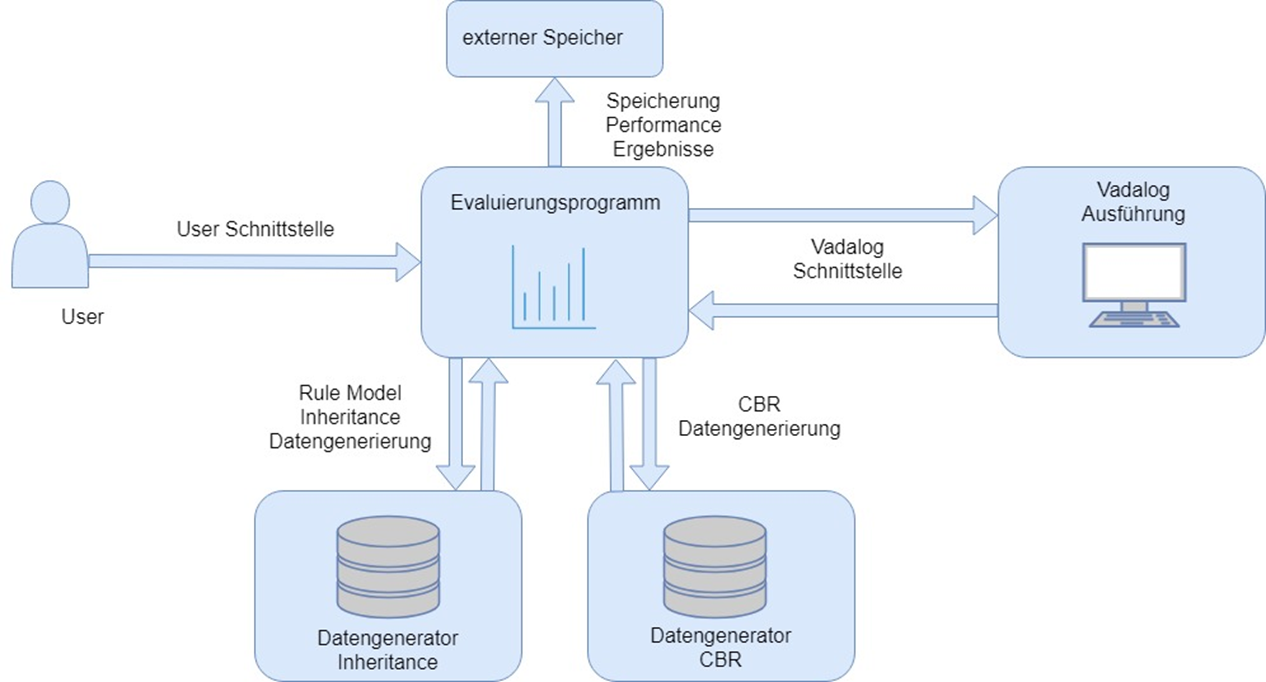


Abbildung 1: Programmarchitektur

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

Die Klasse **BusinessCase.java** speichert die Informationen über den BusinessCase, welche für die Generierung des CBR Codes notwendig sind.

Die Klasse hat folgende Attribute:

String name;

**private** List<String> descProp = **new** ArrayList<String>();

**private** List<String> parameterValues = **new** ArrayList<String>();

In dieser Klasse gibt es zwei ArrayLists mit dem Typ String. Eine der beiden Listen speichert die describing Properties, welche zu dem Business Case gehören. Die andere Liste speichert die zugehörigen ParameterValues, die zu dem Business Case gehören.

Ansonsten finden sich in der Klasse noch die Get und Set Methoden für die Attribute sowie Methoden um neue Werte zu den Listen hinzuzufügen.

### BusinessCaseClass

Die Klasse **BusinessCaseClass.java** speichert die Informationen über die BusinessCaseClass, welche für die Generierung des CBR Codes notwendig sind.

Die Klasse hat folgende Attribute:

**private** String name;

**private** List<BusinessCase> businessCases = **new** ArrayList<BusinessCase>();

In der Klasse gibt es ein Attribut name, vom Typ String, welches den Namen der BusinessCaseClass speichert.

Ebenso existiert eine Array Liste vom Typ BusinessCase, in welcher die dazugehörigen BusinessCase Objekte zu dem BusinessCaseClass Objekt gespeichert werden.

Ansonsten finden sich in der Klasse noch die Get und Set Methoden für das Attribut name sowie eine Methode um neue Werte zu der Liste hinzuzufügen.

### Context

Die Klasse **Context.java** speichert die Informationen über den Context, welche für die Generierung des CBR Codes notwendig sind.

Die Klasse hat folgende Attribute:

**private** String ctx;

**private** String name;

**private** List<ParameterValue> parameterValues = **new** ArrayList<ParameterValue>();

**private** Module module;

**private** String contextClass;

Die Klasse verfügt über einen String ctx, welcher die ID des Context repräsentiert. Der String name speichert den Namen des Context Objekts.

Ebenso verfügt die Klasse über eine Array Liste vom Typ parameterValues, welche die zugehörigen ParameterValues zu dem Context gespeichert werden.

Zusätzlich verfügt die Klasse über ein Objekt Module, welches das zugehörige Module zu dem Context speichert. Ebenso ist ein Attribut contextClass in der Klasse vom Typ String, welches die zugehörige contextClass des Objektes speichert.

Ansonsten finden sich in der Klasse noch die Get und Set Methoden für die Attribute sowie eine Methode um neue Werte zu der Liste hinzuzufügen.

### ContextClass

Die Klasse **ContextClass.java** speichert die Informationen über die ContextClass, welche für die Generierung des CBR Codes notwendig sind.

Die Klasse hat folgende Attribute:

**private** String name;

**private** List<Parameter> parameters = **new** ArrayList<Parameter>();

**private** List<Context> contexts = **new** ArrayList<Context>();

Die Klasse verfügt über ein Attribut name vom Typ String, welches den Namen des ContextClass Objektes speichern soll.

Ebenso verfügt die Klasse über zwei Array Listen. Die eine Liste ist vom Typ Parameter und speichert alle zugehörigen Parameter zu dem ContextClass Objekt.

Die zweite Liste ist vom Typ Context und speichert alle zugehörigen Context Objekte zu dem jeweiligen ContextClass Objekt.

Ansonsten finden sich in der Klasse noch die Get und Set Methoden für die Attribute sowie Methoden um neue Werte zu den Listen hinzuzufügen.

### GeneratorCBR

Die Klasse **GeneratorCBR.java** ist die wichtigste Klasse aus diesem Package. Sie verfügt über die Funktionalität, welche den CBR Code generiert.

Die Klasse verfügt über folgende Attribute:

**private** **static** String *CBRCode*;

**private** **static** ContextClass *cc*;

**private** **static** BusinessCaseClass *bcc*;

Das Attribut CBRCode, welches vom Typ String ist, ist die Variable, in welche der genertierte Code dann gespeichert wird.

Das Attribut cc, vom Typ ContextClass, speichert das ContextClass Objekt, welches für die Generierung des CBR Codes benötigt wird.

Das Attribut bcc, vom Typ BusinessCaseClass, speichert das BusinessCaseClass Objekt, welches für die Generierugn des CBR Codes benötigt wird.

Die Klasse beinhaltet eine Hauptmethode zur Generierung des Codes. Diese Hauptmethode wird aus der ausführbaren Hauptklasse des Programms, EvaluationFrameworkApp.java, aufgerufen, um den CBR Code mit den Input Werten zu generieren.

Diese Hauptmethode sieht folgendermaßen aus:

**public** **static** String generateCBRCode(**int** parameters, **int** paramValues, **int** businessCases) {

*CBRCode* = "";

*CBRCode* += *generateContextClass*();

*CBRCode* += *generateBusinessCaseClass*();

*CBRCode* += *generateParameters*(parameters);

*CBRCode* += *generateParameterValues*(paramValues);

*CBRCode* += *generateParameterValuesHierarchies*();

*CBRCode* += *generateContexts*(paramValues);

*CBRCode* += *generateDetermineParameterValues*();

*CBRCode* += *generateBusinessCases*(businessCases);

*CBRCode* += *generateStaticCode*();

**return** *CBRCode*;

}

Die Hauptmethode des CBR Datengenerators besteht aus Hilfsmethoden, welche den jeweiligen Teil des Codes generieren. Die Aufteilung der Hilfsmethoden weicht von der geplanten Umsetzung des konzeptuellen Entwurfs leicht ab, da während der Implementierung erkannt wurde, dass eine Abänderung sinnvoll und hilfreich ist.

Die Hilfsmethoden generieren den entsprechenden Code Teil und erzeugen dabei die Objekte, welche in den anderen Klassen dieses Packages beschrieben sind. Diese Objekte werden benötigt um den CBR Code generieren zu können.

Die Hauptmethode übergibt einen String, in welchem der gesamte CBR Code gespeichert ist.

Da die Klasse EvaluationFrameworkApp.java diese Methode aufruft, befindet sich der generierte CBR Code dann in dieser Klasse.

### Module

Die Klasse **Module.java** speichert die Informationen über das Module, welche für die Generierung des CBR Codes notwendig sind.

Die Klasse hat folgendes Attribut:

**private** String name;

Das Attribut name vom Typ String beschreibt den Namen des Module Objekts.

Ansonsten finden sich in der Klasse noch die Get und Set Methoden für das Attribut.

### Parameter

Die Klasse **Parameter.java** speichert die Informationen über das Module, welche für die Generierung des CBR Codes notwendig sind.

Die Klasse hat folgende Attribute:

**private** String name;

**private** List<ParameterValue> parameterValues = **new** ArrayList<ParameterValue>();

**private** String descProp;

In der Klasse befindet sich an Attribut name, vom Typ String, welches den Namen des Parameter Objekt speichert.

Ebenso befindet sich eine Array List vom Typ ParameterValue, welche die zugehörigen parameterValues zu dem Parameter Objekt speichert.

Zusätlich gibt es noch ein Attribut descProp, vom Typ String, welches die describing Property des Parameter Objekt speichert.

Ansonsten finden sich in der Klasse noch die Get und Set Methoden für die Attribute sowie eine Methode um neue Werte zu der Liste hinzuzufügen.

### ParameterValue

Die Klasse **ParameterValue.java** speichert die Informationen über das Module, welche für die Generierung des CBR Codes notwendig sind.

Die Klasse hat folgendes Attribut:

**private** String name;

Das Attribut name vom Typ String beschreibt den Namen des ParameterValue Objekts.

Ansonsten finden sich in der Klasse noch die Get und Set Methoden für das Attribut.

## DataGeneratorRandomString



Abbildung : Package DataGeneratorRandomString

In dem Package DataGeneratorRandomString befindet sich nur eine Klasse, die **GeneratroRandomString.java**. Diese Klasse dient zur Unterstützung der CBR und RMI Datengeneratoren, dies ist in Abbildung 4 zu sehen.

In dieser Klasse befinden sich 2 Methoden, welche zufällig generierte Strings erzeugen.

Folgende 2 Methoden befinden sich in der Klasse:

**public** **static** String getRandomString(**int** length){…}

**public** **static** String getRandomBigChar(**int** length){…}

Die Methode getRandomString erzeugt einen String aus zufällig aneinander gereihten Kleinbuchstaben. Die Länge des zu generierenden Strings wird als Parameter der Methode mitgegeben.

Die Methode getRandomBigChar erzeugt einen String aus zufällig aneinander gereihten Großbuchstaben. Die Länge des zu generierenden Strings wird als Parameter der Methode mitgegeben.

Die beiden Methoden werden von den Datengeneratoren aufgerufen, um diese bei der Erzeugung der CBR und RMI Codes zu unterstützen.

## DataGeneratorRMI

## DB

## EvaluationFramework

### CBR Code Generierung

### RMI Code Generierung

## Exceptions

## Models

### CBR

### RMI

## Test

### CBR Tests

### RMI Tests

## Vadalog

# Tests der geforderten Funktionalitäten

# Abhängigkeiten Bibliotheken

# Installationsanleitung

# Limitierungen und Verbesserungsvorschläge im praktischen Einsatz

# Abbildungsverzeichnis

[Abbildung 1: Programmarchitektur 2](file:///C:\Users\Anwender\Documents\GitHub\258321_DKE_PR\Dokumente\Meilenstein%20Implementierung\Meilenstein_Implementierung.docx#_Toc11114)

[Abbildung 2: Package Struktur im Programm 4](file:///C:\Users\Anwender\Documents\GitHub\258321_DKE_PR\Dokumente\Meilenstein%20Implementierung\Meilenstein_Implementierung.docx#_Toc11115)

[Abbildung 3: Package DataGeneratorCBR 5](file:///C:\Users\Anwender\Documents\GitHub\258321_DKE_PR\Dokumente\Meilenstein%20Implementierung\Meilenstein_Implementierung.docx#_Toc11116)

# Tabellenverzeichnis

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