Model-Driven Engineering Übungsbeispiel

Hochschule Bremen / KSS

Methoden zur Entwicklung komplexer Softwaresysteme

SS 2013

Übungsbeispiel

Organisation

- Das Beispiel dient zur Vorbereitung auf die Klausur und soll als asynchroner Anteil der LV bearbeitet werden.
- Die Übung kann in Gruppen bis zu 3 Personen durchgeführt werden, wobei die Bewertung pro Gruppe erfolgt.
- Die Abgabe der Übung erfolgt via AULIS.
- Die Abgabefrist Endet mit dem Tag der schriftlichen Klausur.
- Mit der abgegeben Übung können bis zu 10 Punkte erreicht werden, die zum Ergebnis der Klausur (max. 100 Punkte) hinzugerechnet werden (mehr als gesamt 100 Punkte werden nicht gewertet ;-)

Übungsbeispiel

Aufgabenstellung

Es soll ein Generator Framework entwickelt werden, welches das “Metamodel-Interface” Projekt erweitert.
Es sollen Code Generatoren sowie unterschiedliche konkrete Syntax-Varianten implementiert werden.
Zeigen Sie mit JUnit Testfällen, wie das Framework arbeitet.

1. **Interface Metamodel** (Ausgangspunkt)
   Das Metamodel für einfache Java Interfaces (nur primitive Datentypen und keine Exceptions) finden Sie im Projekt „Metamodel-Interface“
   Auch ein Beispiel-Modell (Stack.java) ist dort zu finden.

Übungsbeispiel

Aufgabenstellung

2. **Code Generatoren (4 Punkte)**
   Implementieren Sie Code-Generatoren, die für ein Interface Modell folgende Artefakte erzeugen:
   - Java Interface
   - Abstrakte Decorator Basis-Klasse
   - Konkreter Decorator der alle Methodenaufrufe überwacht (Ausgabe der Methoden, Parameter, Rückgabewerte und Laufzeiten in ms)

3. **Expression Builder (2 Punkte)**
   Implementieren Sie einen Expression Builder mit dem Sie das Interface Modell über ein „fluent API" erstellen können (siehe internal DSL).

Übungsbeispiel

Aufgabenstellung

4. **Annotations (2 Punkte)**
   Implementieren Sie eine Mapper Klasse die aus einem Java Interface mit der Annotation @Decorator ein Interface Modell via Reflection aufbaut.

5. **Parser (2 Punkte)**
   Implementieren Sie einen Parser mit dem Sie das Interface Modell aus einem Java Source-File einlesen. Hinweis: Sie benötigen dazu ein Grammar File mit Actions.