

# MODELLGETRIEBENE ENTWICKLUNG EINER MOBILEN APPLIKATION MIT JUSE4ANDROID

Jano Espenhahn, Tobias Franz and Franziska Krebs  
Fachhochschule Brandenburg, Fachbereich Informatik und Medien  
{espenhah, franzt, krebsf}@fh-brandenburg.de

Keywords: MDA, UML, USE, OCL, Android

Abriss: ein deutsches Abstract

Abstract: ein englisches Abstract

## 1 EINLEITUNG

### 1.1 Motivation

Zitat Test (?)

### 1.2 Ziel

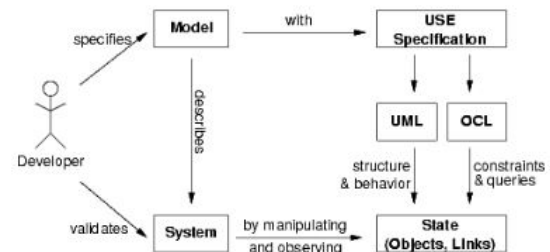
### 1.3 Aufgabenstellung

### 1.4 Abgrenzung

### 1.5 Ergebnis

## 2 VORSTELLUNG USE

UML based Specifiatiion Environment (USE) wird zur Spezifikation von Informationssystemen verwendet und wurde an der Universität Bremen entwickelt. Es basiert auf einer Teilmenge der Unified Modeling Language (UML) und der Object Constraint Language (OCL). Eine USE-Spezifikation besteht aus einer textuellen Beschreibung eines Modells, bei der Eigenschaften aus UML-Diagramm verwendet werden. Um eine Spezifikation auf nicht-formale Anforderungen zu validieren, kann ein Modell mithilfe des USE-Tools animiert werden. Weitere Integritätsausdrücke für ein Modell können durch die OCL definiert werden. (?) Die OCL wird im späteren



Workflow einer USE-Spezifikation (?)

Kapitel (TODO) vorgestellt. Die nachfolgende Abbildung veranschaulicht den Workflow für eine USE-Spezifikation.

Ein Entwickler spezifiziert ein USE-Modell, das ein System beschreibt und nutzt dabei UML- und OCL-Ausdrücke. Mithilfe von USE ist es ihm möglich zu validieren, ob die bestimmten Anforderungen an sein System mit dem Modell erfüllt sind.

### 2.1 Syntax

Die textuelle Beschreibung eines Modells mit USE beginnt immer mit der Definition eines Modell-Namens. In diesem Fall ist das *IceCream*. Im Anschluss folgen Klassendefinitionen mit ihren jeweiligen Attributen und Methoden. Im Beispiel hat die Klasse *Station* das Attribut *name* und die Operation *entries* ohne Übergabeparameter. Das folgende Beispiel basiert lediglich auf UML. OCL-Ausdrücke werden später vorgestellt.

model IceCream

## ANHANG

```
class Station
  attributes
    name      : String
  operations
    entries() : Set(Entry) = self.records->asSet
end
```

Klassen können untereinander in Abhängigkeit stehen. Für diese Abhängigkeiten sind Assoziationen vorgesehen. Um eine Assoziation auszudrücken, wird zuerst eine weitere Klasse *Address* eingeführt.

```
class Address
  attributes
    street : String
    postCode : Integer
end
```

Für das dem Artikel zugrunde liegende Beispiel kann eine Station entweder eine oder keine Adresse haben.

```
association Station_Address between
  Station[ 1 ]
  Address[ 0..1 ] role place
end
```

*Station\_Address* ist dabei der Name der Assoziation und das Attribut *place* nimmt in der Klasse *Station* die Rolle für die Adresse ein. Um das gesamte Modell zu vervollständigen, fehlen noch die Klasse *Entry* und die Assoziation *Station\_AddressStation\_Entry*.

```
class Entry
  attributes
    date      : CalendarDate
    target     : Integer
    actual     : Integer
    variance   : Integer
  operations
    variance() : Integer = actual - target
end

association Station_Entry between
  Station[ 1 ]
  Entry[ * ] role records
end
```

## 2.2 Tool

## 3 VORSTELLUNG OCL

## 4 JUSE4ANDROID

## REFERENCES

- da Silva, L. (2014). Model-driven generative programming for bis mobile applications. Master's thesis, ISCTE IUL University of Lisbon.
- Database Systems Group, B. U. (2007). *USE - A UML based Specification Environment*.
- Martin Gogolla, Fabian Buttner, M. R. (2007). Use: A uml-based specification environment for validating uml and ocl.