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

Zitat Test (da Silva, 2014)

2 VORSTELLUNG USE

UML based Specifiation Environment (USE) wird zur Spezifikation von Informationssystemen verwendet und wurde an der Universität Bremen entwickelt. Neben dem Einsatz für Fallstudien, wird USE vor allem in der Lehre an Hochschule wie z. B. MIT, Cambridge, University of Edinburgh und University of Lisbon eingesetzt. USE 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. Weitere Integritätsausdrücke für ein Modell können durch die OCL definiert werden. (Martin Gogolla, 2007) Die OCL wird im späteren Kapitel (TODO) vorgestellt.

Die Abbildung 1 veranschaulicht den Workflow für eine USE-Spezifikation. Ein Entwickler spezifiziert ein USE-Modell, welches ein System beschreibt und nutzt dafür UML- und OCL-Ausdrücke. Mithilfe von USE ist es ihm möglich die bestimmten Anforderungen an sein System auf Erfüllung mit dem Modell zu validieren.

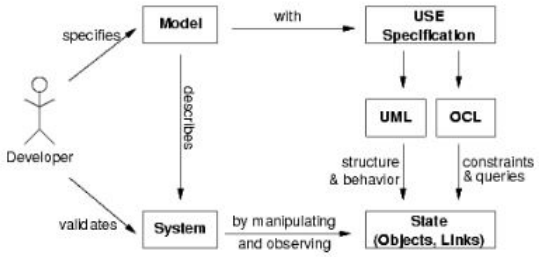

Abbildung 1: Workflow einer USE-Spezifikation (Database Systems Group, 2007)

2.1 Spezifikation

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. Die nachfolgenden Code-Ausschnitte verwenden lediglich UML.

```
model IceCream

class Station
   attributes
      name          : String
      target        : Integer
   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. Zum gesamten USE-Modell gehören weiterhin noch die Klasse *Entry* und die Assoziation *Station_Entry*.

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

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

Zur Vervollständigung des Modells gehört außerdem eine aus der Arbeit (da Silva, 2014) entnommene Klasse *CalendarDate*.

2.2 Tool

Um eine Spezifikation auf nicht-formale Anforderungen zu validieren, kann ein Modell mithilfe des USE-Tools animiert werden. Direkt nach dem Import eines Modells erhält man vom Tool ein Feedback über die Validität der UML- und OCL-Definitionen. Neben der Validierung bietet das Tool weitere Möglichkeiten, wie z. B. die Visualisierung eines Klassen-, Sequenz- oder Objektdiagramms. In der Abbildung 2 finden sich die im Kapitel 2.1 definierten Klassen und Assoziationen als Klassendiagramm wieder.

3 BESCHREIBUNG DER ANWENDUNG

Das Beispiel wurde aus dem Artikel (Fowler, 2006) entnommen. Es handelt sich um ein fiktives Programm der Regierung zur Kontrolle der Eispartikel in der Luft. Wenn die Konzentration zu niedrig ist, bedeutet das, dass die Bevölkerung zu wenig Eiscreme isst, was eine Menge an Risiken für die Umwelt und die öffentliche Ordnung darstellt. Um die Eispartikel in der Luft zu überwachen, hat der Staat Kontrollstationen im gesamten Land verteilt

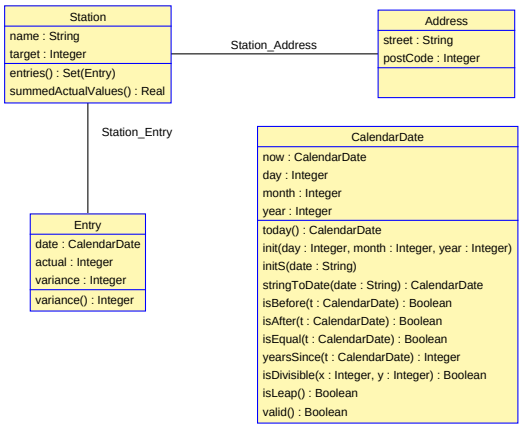

Abbildung 2: Klassendiagramm für das Beispiel

aufgestellt. Für jede Station gibt es einen festgelegten Zielwert der Eispartikel. Der aktuelle Wert weicht in der Regel vom Zielwert ab.

Die Anwendung ermöglicht es neue Stationen mit Zielwerten aufzunehmen und alte Stationen zu löschen. Außerdem gibt es die Möglichkeit eine Adresse zu einer Station anzugeben. Eine Adresse ist im Nachhinein auch wieder entfernbar. Die Erfassung von beliebig vielen Einträgen zu einer Station ist ebenfalls möglich. Auch Einträge lassen sich im Nachhinein wieder entfernen. Zudem wird für jeden Eintrag, nach Eingabe des aktuellen Wertes die Abweichung zum Zielwert angezeigt.

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*.

Fowler, M. (2006). Gui architectures.

Martin Gogolla, Fabian Buttner, M. R. (2007). Use: A uml-based specification environment for validating uml and ocl.

ANHANG

If any, the appendix should appear directly after the references without numbering, and not on a new page. To do so please use the following command: *\section*{APPENDIX}*