Karlsruher Institut für Technologie

Karlsruher Institut für Technologie

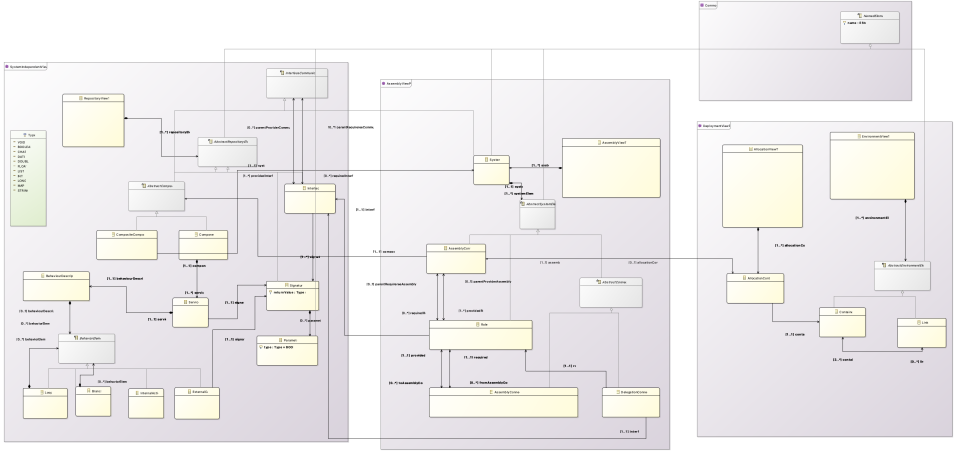
Abschlusspräsentation

Praktikum: Modellgetriebene Software-Entwicklung

Dirk Neumann, Patrick Mehl, Jakob Höfker | 12. Juli 2021

Inhaltsverzeichnis

Meta-Modellierung



Entwurfsentscheidungen für die Meta-Modellierung

1. Verwendung von 4 Packages
   - klare Trennung der View Points
   - Nachteil für spätere Entwicklung
2. Common Package für wiederverwendbare Elemente
3. ENUM zur Modellierung des `Type`
4. Verwendung von OCL
5.
6. Designentscheidung: weniger oder mehr Informationen am Anfang beschreiben beeinflusst wie man mit dem Meta-Modell weiterarbeiten kann

Xtext - textuelle Syntax

```
EnvironmentViewType {
    environmentElements {
        Container ApplicationServer,
        Container DatabaseServer,
        Link Network {
            containers (ApplicationServer, DatabaseServer)
        }
    }
}
AllocationViewType {
    allocationContexts {
        AllocationContext {
            container ApplicationServer
            assembly Sys.WebGUIAC
        },
        AllocationContext {
            container ApplicationServer
            assembly Sys.MediaStoreAC
        },
        AllocationContext {
            container DatabaseServer
            assembly Sys.PoolingAudioDBAC
        }
    }
}
```

- Ergebnis: `mediastore.simplepalladio`

Probleme beim Entwurf der textuellen Syntax

1. Import von verschiedenen Packages konnte nicht aufgelöst werden
   - Lösung: Verwendung von alternativer Importschreibweise anstatt Ns URI
   - `import "platform:/resource/SimplePalladio/..."`
2. Probleme mit Proxy-Ausflösung und zyklischen Abhängigkeiten
   - Lösung: Grammatik als Weaving-Modell der verschiedenen Modelle in einer Datei
   - `<datei>.simplepalladio`
3. OCL Constraints werden automatisch ausgewertet $\Rightarrow$ führte zu Fehlern, da OCL Constraints defekt waren

QVTo - Modelltransformation

Entwurfsentscheidungen und Probleme bei der Modelltransformation

1. Verwendung von intermediate-Objekt `TRole` zur Repräsentation von unterschiedlich existierenden Konzepten im Meta-Modell
   - Interface- und Communicator-Konzept ist in Palladio anders umgesetzt
   - Unterschiede: Modellierung in Metamodell als Property und in Palladio als Klasse
2. Ausnutzen der Vererbungshierarchie in Metamodell
   - Abstraktes Mapping: Mapping von abstraken Klassen mit Hilfe von polymorphen Disjunct-Mapping
   - Für Vererbungshierarchien auf Source-/Target-Seite $\rightarrow$ zusätzliche Verwendung für Mapping der Datentypen

Code-Generierung mit Xtend

```java
package DBCache;

import repository.AudioDB;

public class DBCacheImpl implements  AudioDB  {
    private AudioDB audioDB;

    public DBCacheImpl() {}

    public void setAudioDB(AudioDB audioDB) {
        Helper.assertNull(this.audioDB);
        this.audioDB = audioDB;
    }

    //Implementing addFile from interface AudioDB
    @Override
    public void addFile () {
        Helper.assertNotNull(this.audioDB);
        //TODO: implement
    }
    //Implementing queryDB from interface AudioDB
    @Override
    public void queryDB () {
        Helper.assertNotNull(this.audioDB);
        //TODO: implement
    }
}
```

- Beispielhafte Erzeugung des Quellcodes für die DBCacheImpl-Datei mit Attributen und Methoden-Templates
- Generierung von `DBCache/DBCacheImpl.java`

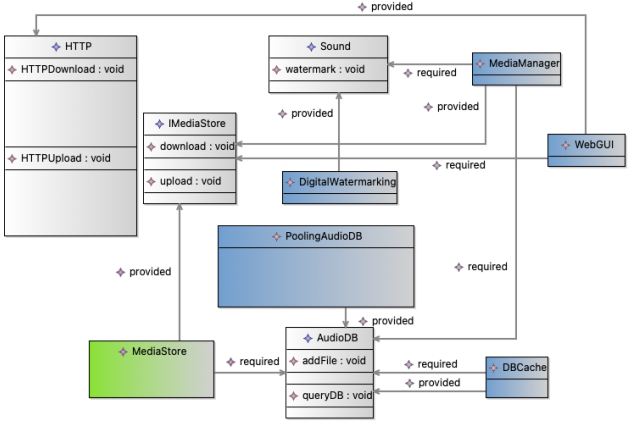
Entwurfsentscheidungen bei der Code-Generierung

1. Erzeugung der Klassen in siblings-Packages anstatt in respository-Package

Probleme bei der Code-Generierung

1. OCL Constraints konnten nicht aufgelöst werden
   - Deaktivierung der OCL-Validierungq

Sirius - grafischer Editor



- graues Element: Interface
- grünes Element: Composite Component
- blaues Element: Component
- required Edge: z.B. Component required Interface
- provided Edge: z.B. Component provided Interface

Entwurfsentscheidungen beim grafischen Editor

1. Generell: Erstellung der konkreten Syntax von der abstrakten Syntax (Meta-Modell)
2. Parameter werden als Liste innerhalb von Signaturblöcken dargestellt
3. Verwendung des grafischen Editors, um die Verwendung des Meta-Modells so intuiitiv wie möglich zu gestalten
   - Verwendung von üblichen Farben, Symbolen, Icons ermöglicht die Modellierung von

Probleme beim grafischen Editor

1. Kontextwechsel muss man wissen
   - Beispiel: Signaturen gehören bei Sirius zu Interfaces → wenn Interface gelöscht wird, dann ist Signatur weg, in Meta-Modell ist es weiterhin da
2. Parameterlisten
3. eigenständige Datentypen als Parameter

Key Learnings (1)

1. OCL wurde am Anfang wenig getestet und hat dann zu Problemen im weiteren Verlauf der Meilensteine geführt
2. Unterscheidung zwischen einfacher Modellierung und OCL hat unterschiedliche Vor- und Nachteile gegenüber von Modellierung über komplexeren Klassenstrukturen
   - kleinere Klassenstrukturen, jedoch OCL muss getestet werden
   - geringere Konsistenz-Haltung durch mehr Klassen und Attribute und weniger OCL
3. Unterschiedliche Form von Modellierungskonzepten
   - Beispiel: Modelltransformation (siehe Probleme Modelltransformation)
4. Informationsbeschaffung durch Literatur und Foren im Internet ist nur schwierig möglich → wenig vorhanden und wenn alte outdated Forumseinträge

Key Learnings (2)

1. Interpreter in Eclipse: OCL-Ausdruck eingeben und zeigt die Werte an
2. Vorsichtig sein mit bidirektionalen Referenzen:
   - nützlich bei OCL und Sirius → leichter um zwischen Kontexten zu wechseln (z.B. beim Löschen kann man in beide Richtungen gehen)
   - Nachteilig bei Xtext → jedoch Lösung durch Sandwiching