

Abschlusspräsentation

Praktikum: Modellgetriebene Software-Entwicklung

Dirk Neumann, Patrick Mehl, Jakob Höfker | 5. Juli 2021

Inhaltsverzeichnis

1. M1: Meta-Modellierung
2. M2: Xtext - textuelle Syntax
3. M3: QVTo - Modelltransformation
4. M4: Xtend - Code-Generierung
5. M5: Sirius - grafischer Editor
6. Key Learnings

M1: Metamodell
○○

M2: Xtext
○○

M3: QVTo
○○

M4: Xtend
○○○

M5: Sirius
○○

Key Learnings
○



Entwurfsentscheidungen für die Meta-Modellierung

- Verwendung von 4 Packages
 - klare Trennung der View Points
 - Nachteil für spätere Entwicklung
- Common Package für wiederverwendbare Elemente
- ENUM zur Modellierung des Type
- Verwendung von OCL

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

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

- ## Key Learnings

Entwurfsentscheidungen und Probleme bei der Modelltransformation

- ① Verwendung von intermediate-Objekt TRo1e 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 abstrakten Klassen mit Hilfe von polymorphen Disjunct-Mapping
 - Für Vererbungshierarchien auf Source-/Target-Seite → zusätzliche Verwendung für Mapping der Datentypen

Code-Generierung mit Xtend

```
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

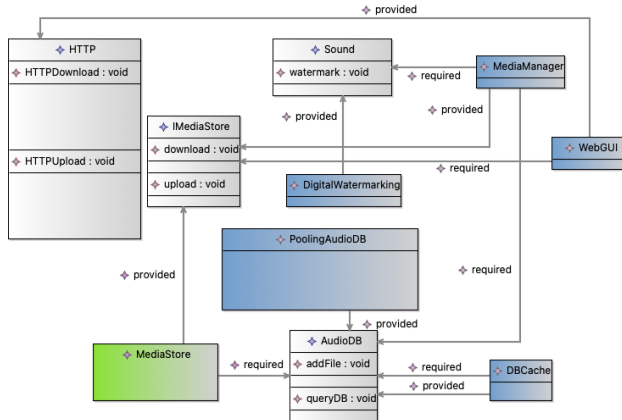
Entwurfsentscheidungen bei der Code-Generierung

- Erzeugung der Klassen in siblings-Packages anstatt in repository-Package

Probleme bei der Code-Generierung

- OCL Constraints konnten nicht aufgelöst werden
 - Deaktivierung der OCL-Validierung

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

M1: Metamodell
○○

M2: Xtext
○○

M3: QVTo
○○

M4: Xtend
○○○

M5: Sirius
●○

Key Learnings
○

Entwurfsentscheidungen beim grafischen Editor

- Parameter werden als Liste innerhalb von Signaturblöcken dargestellt

Key Learnings

- OCL wurde am Anfang wenig getestet und hat dann zu Problemen im weiteren Verlauf der Meilensteine geführt
- 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
- Unterschiedliche Form von Modellierungskonzepten
 - Beispiel: Modelltransformation (siehe Probleme Modelltransformation)