FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Diplomarbeitspräsentation

ATL4pros: Introducing Native UML Profile Support into the ATLAS Transformation Language

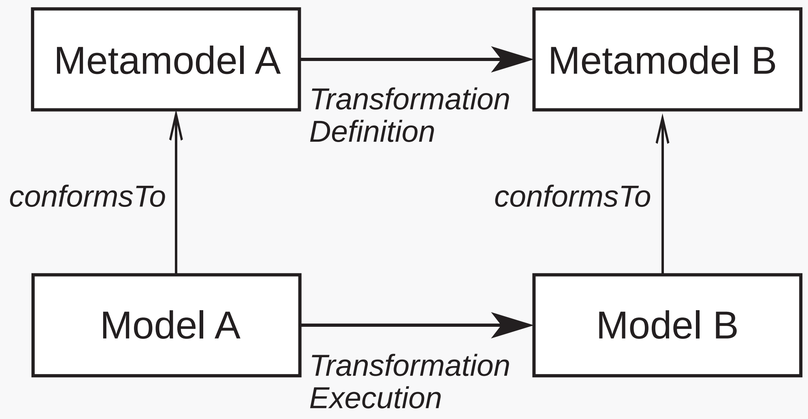
Andrea Randak

Masterstudium:
Wirtschaftsinformatik

Technische Universität Wien
Institut für Softwaretechnik und Interaktive Systeme
Arbeitsbereich: Business Informatics Group
Betreuerin: O.Univ.-Prof. Dipl.-Ing. Mag. Dr. Gerti Kappel

Introduction

- In Model Driven Engineering (MDE), models are the key artifacts of the entire engineering process
- UML is the most popular modeling language today
- UML profiles may be used to tailor UML models for specific domains and platforms
- Model transformation is crucial for transforming source to target models
  - **ATLAS Transformation Language (ATL)** as most prominent representative



Model-to-Model transformation

Problem Statement

- Metamodels are well supported in ATL
- Using UML profiles demands for a complex workaround
  - Verbose transformation code
  - Calls to underlying Java UML2 API
  - API knowledge required
  - Complex imperative code statements
- Readability diminished, hardly maintainable

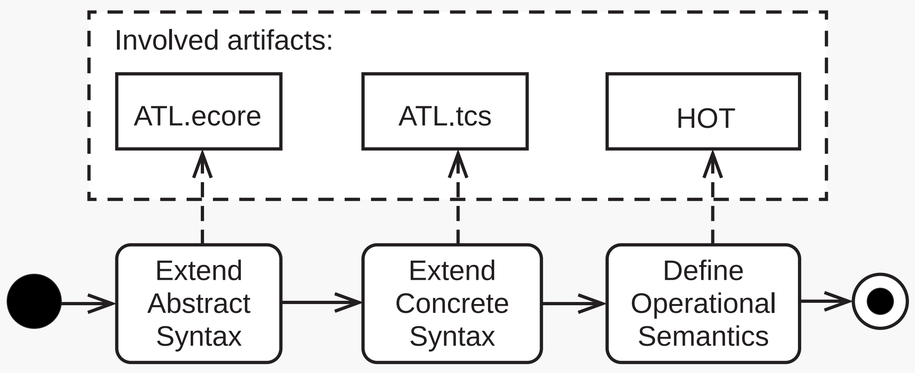
Aim

Extending the ATLAS Transformation Language for a native UML profile support

- Concise transformation code due to new keyword **apply**
- No further knowledge about Java UML2 API required
- Hide imperative statements from transformation engineer
- Enhance readability and maintainability

Extension Process at a Glance

- **Abstract Syntax extension**
  Insertion of new classes into the ATL metamodel, i.e., the abstract syntax, for integration of new keyword
- **Concrete Syntax extension**
  Extension of the Textual Concrete Syntax (TCS) of ATL for reflecting all modifications of the abstract syntax
- **Operational Semantics definition**
  Definition of a Higher-Order Transformation (HOT) for translating ATL4pros to an executable version in standard ATL



Extension process and involved artifacts

Result - ATL4pros

```
helper def: stereo : uml!Stereotype = OclUndefined;

rule SubjectArea_2_Package  {
    from
        s : DSL!SubjectArea
    to
        t : UML!Package (
            name <- s.name
        )
    do {
            t.applyProfile(profile!Profile.allInstances().asSequence().first());

            thisModule.stereo <- profile!Stereotype.allInstances()
                -> any( e | e.name = 'SubjectArea');

            t.applyStereotype(thisModule.stereo);

            if(not s.phase.oclIsUndefined()){
                t.setValue(thisModule.stereo, 'phase',s.phase);
            }

    }
}
```

ATL

```
rule SubjectArea_2_Package  {
    from
        s : DSL!SubjectArea
    to
        t : UML!Package (
            name <- s.name
        ) apply PRO!SubjectArea (
                phase <- s.phase
        )
}
```

ATL4pros

Conclusion

- **ATL4pros** eases the use of UML profiles in a model-to-model transformation
- New keyword was successfully integrated into the abstract and the concrete syntax of ATL
- Case study showed that all imperative code could be refactored to declarative code
- Extension process as guideline for future extensions of ATL

Kontakt: andrea@randak.at