Model Transformations for DSL Processing

Stefan Kapferer

January 14, 2019

University of Applied Sciences of Eastern Switzerland (HSR FHO)

Table of contents

Introduction: Model Transformation

Models

Models are used in **all disciplines and phases** of the **software development lifecycle**:

- **Business Modeling**: Domain Models, Use Case Models, ...
- **Analysis & Design**: Architecture View Models, Context Maps ...
- **Implementation**: Class Models, Object Models, Data Models, ...
- **Testing**: Performance Simulation Models, ...
- **Operations & Maintenance**: Deployment Models, ...

Model Transformation

Goal: Transform a model into another model.

Models differ in:

- Level of abstraction
- Representation / Language
- Metamodel (Eclipse Ecore, UML, ...)

Examples:

- Change level of abstraction
  - Refinement of domain model towards fully-fledged class diagram
- Change representation or language (keep semantics)
  - Refactorings
  - Code migration into other language

DSL Example & Live Demo

ContextMapper DSL [2]

A Domain-specific Language for Context Mapping & Service Decomposition[1]

CONTEXT MAPPER

- Modeling Domain-driven Design (DDD) Context Maps
- Goal: Apply model transformations to realize architectural refactorings [5] towards service decomposition
  - Split bounded contexts

DSL Processing via Model Transformation:

- DSL Text $\xrightarrow{parsing}$ Abstract Syntax Tree (AST) $\rightarrow$ Model
- Model $\xrightarrow{transformation}$ Model
- Model $\rightarrow$ Abstract Syntax Tree (AST) $\xrightarrow{unparsing}$ DSL Text

[1] `https://contextmapper.github.io/`

ContextMapper DSL Example Context Map

Example Context Map:

```
ContextMap {
  /* Add Bounded Contexts to Context Map */
  contains CustomerManagement
  contains CustomerSelfService
  contains PolicyManagement
  contains DebtCollection

  /* Define Bounded Context Relationships: */

  CustomerSelfService -> CustomerManagement : Customer-Supplier

  PolicyManagement -> CustomerManagement : Upstream-Downstream {
    implementationTechnology = "RESTful HTTP"
    upstream implements OPEN_HOST_SERVICE, PUBLISHED_LANGUAGE
    downstream implements CONFORMIST
  }

  PolicyManagement <-> DebtCollection : Shared-Kernel {
    implementationTechnology = "Shared Java Library"
  }
}
```

ContextMapper DSL Example: Input

DSL snippet modeling a bounded context:

```
/* Example Bounded Context in CML */
BoundedContext CustomerManagement {
  Aggregate Customers {
    Entity Customer {
      String firstName
      String familyName
      Account customerBankAccount
    }
    Entity Account {
      String iban
      String bankName
    }
  }
  Aggregate CustomerSelfService {
    Entity Account {
      String username
      String password
      Customer owner
    }
  }
}
```

Live Demo: «Split Bounded Context by Duplicate Entity Name»

ContextMapper DSL Example: Output

```
/* Example Bounded Context in CML */
BoundedContext CustomerManagement {
  Aggregate Customers {
    Entity Customer{
      String firstName
      String familyName
      Account customerBankAccount
    }
    Entity Account {
      String iban
      String bankName
    }
  }
}
BoundedContext SplitBoundedContext {
  Aggregate CustomerSelfService {
    Entity Account {
      String username
      String password
      Customer owner
    }
  }
}
```

Model Transformations with Henshin

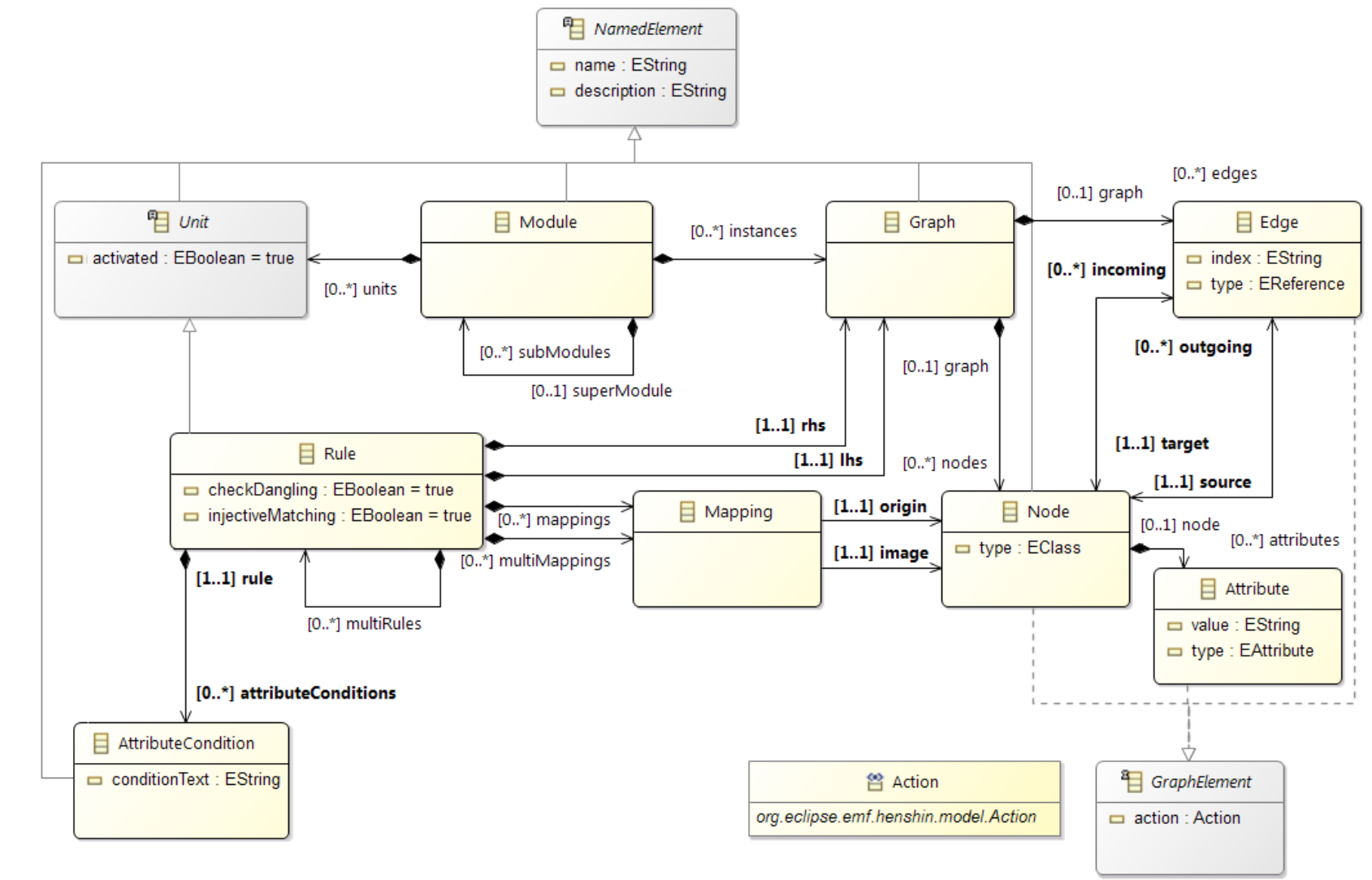
Henshin Transformation Tool

Henshin [1] is an EMF [4] based transformation tool.

- *Henshin* means transformation in Japanese
- Supports in-place model transformations
- Endogenous & Exogenous
- Horizontal & Vertical

- Based on Algebraic Graph Transformation [3]

The Henshin Transformation Meta-Model[2]

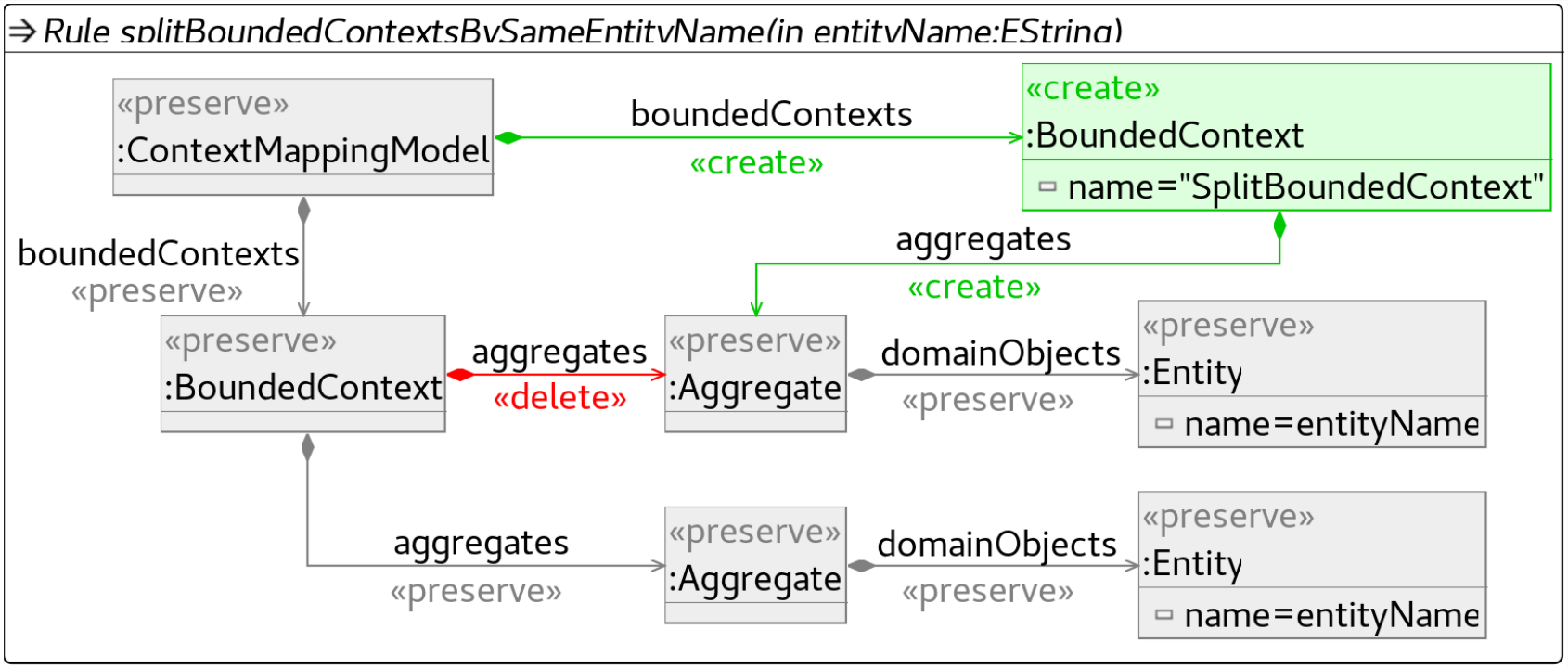


[2] Copied from
https://wiki.eclipse.org/Henshin/Transformation_Meta-Model

Example Transformation Model

Transformation model for example seen in the live demo:

- LHS graph: «preserve» + «delete»
- RHS graph: «preserve» + «create»



Algebraic Graph Transformation

From String Grammars ...

The classical string grammar you know:

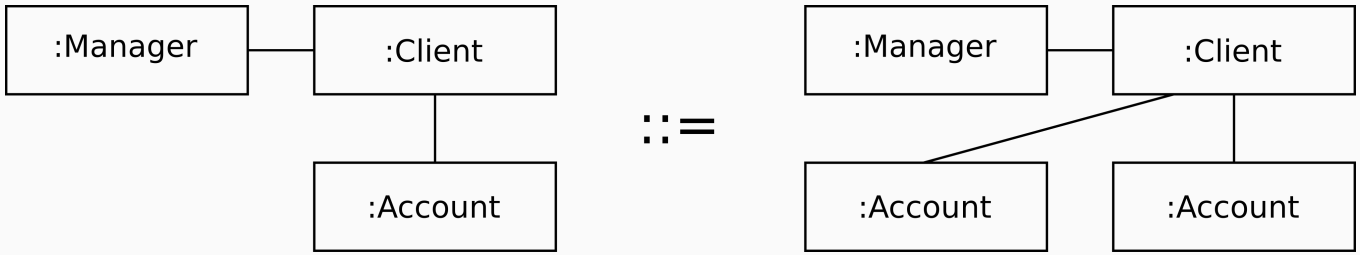
```
DecimalNumeral —> 0 | NonZeroDigit Digits
Digits         —> ε | Digit | Digits Digit
Digit          —> 0 | NonZeroDigit
NonZeroDigit   —> 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

- Example: *DecimalNumeral* grammar of the Java Language Specification in Backus-Naur Form (BNF).
- String grammar consists of a **set of production rules**.

… to Graph Grammars

Similar principle with graphs:

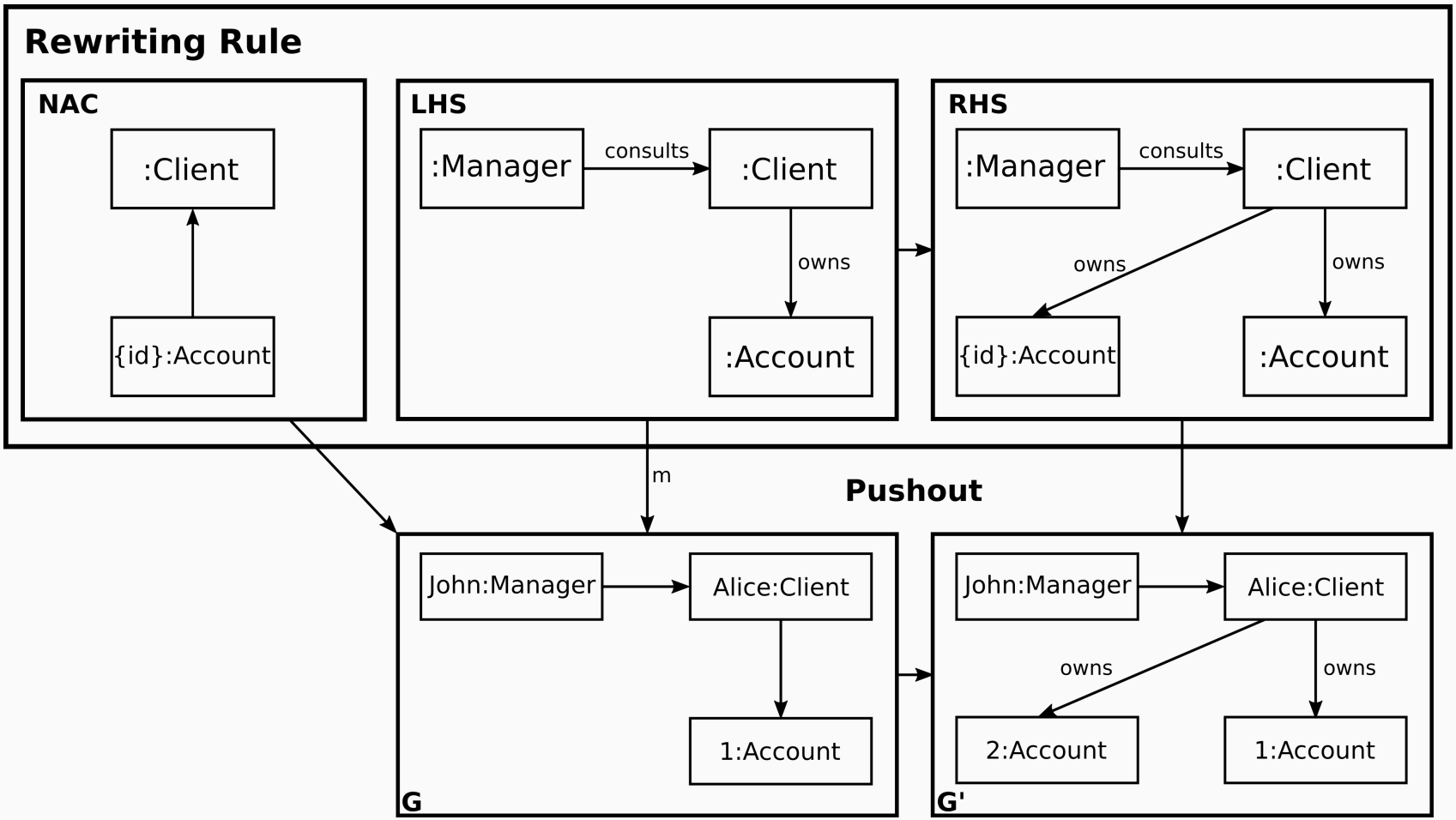
- Left-hand side (LHS) of the rule has to be matched in graph on which the rule is applied to.
- Right-hand side (RHS) describes the changes which will be applied



Example: Banking scenario «create new account for a customer».

Pushout Operation

The so-called **pushout** is an operator transforming a graph *G* into another graph *G'* given a production *p* (graph grammar rule) and two graph morphisms.



Summary & Conclusions

Seminar Question #1: Model Transformation vs. Program Transformation

- Other **software engineering disciplines**
  - Business Modeling, Analysis & Design vs. Implementation
- Other **level of abstraction**

- **Model Transformations**
  - model-to-model
  - model-to-code
  - code-to-model
- **Program Transformations**
  - code-to-code

Note: Refactoring (code-to-code) can be implemented as model transformation too.

Seminar Question #2:
Henshin Maturity & Differences to other Tools

- Henshin **Conclusion**:
  - + Expressive transformation specification
  - + Mature transformation engine
  - + Solid foundations
  - - Improvable Tooling

- Main **differences** to other transformation approaches:
  - Not only theoretic research project
    - Good compromise between scientific foundations & feasible tool implementation
  - Declarative transformation specification
    - Other tools often use imperative approaches

Seminar Question #3:
Model Transformation for DSL Processing

- DSL as a customized **model representation**
- DSL text can be transformed into other model representations
- Example:
  - DSL to EMF model («**Parsing**»)
  - EMF model to DSL («**Unparsing**»)

- **DSL processing** approach:
  - DSL Text $\xrightarrow{parsing}$ Abstract Syntax Tree (AST) $\rightarrow$ Model
  - Model $\xrightarrow{transformation}$ Model
  - Model $\rightarrow$ Abstract Syntax Tree (AST) $\xrightarrow{unparsing}$ DSL Text

Questions & Discussion

Questions?

Discussion

References i

T. Arendt, E. Biermann, S. Jurack, C. Krause, and G. Taentzer.
Henshin: Advanced concepts and tools for in-place emf model transformations.
In *Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems: Part I*, MODELS'10, pages 121–135, Berlin, Heidelberg, 2010. Springer-Verlag.

ContextMapper DSL.
A Domain-specific Language for Context Mapping & Service Decomposition.
`https://contextmapper.github.io/`.
[Online; Accessed: 2018-12-09].

References ii


H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer.
Fundamentals of Algebraic Graph Transformation.
Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2006.


D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro.
EMF: Eclipse Modeling Framework.
Eclipse Series. Pearson Education, 2008.


O. Zimmermann.
Architectural refactoring for the cloud: a decision-centric view on cloud migration.
Computing, 99(2):129–145, 2017.