

CORE – CREATING ROLE DIAGRAMS AND MORE

Henri Mühle

Fakultät für Mathematik
Universität Wien

23.02.2011

① INTRODUCTION

② GRAPHICAL MODELING FRAMEWORK

③ CORE – CONTEXTUAL ROLE EDITOR

1 INTRODUCTION

2 GRAPHICAL MODELING FRAMEWORK

3 CORE – CONTEXTUAL ROLE EDITOR

INTRODUCTION

- ▶ since my diploma thesis, I have accomplished some work on representing role-oriented software models in terms of formal contexts
- ▶ when working at the Department of Computer Science, I was supposed to work with the Eclipse Graphical Modeling Framework (GMF)
- ▶ this gave rise to the idea to develop a modeling tool to create role models
- ▶ the intended tool should transform contexts into role models and vice versa

INTRODUCTION

- ▶ since my diploma thesis, I have accomplished some work on representing role-oriented software models in terms of formal contexts
- ▶ when working at the Department of Computer Science, I was supposed to work with the Eclipse Graphical Modeling Framework (GMF)
- ▶ this gave rise to the idea to develop a modeling tool to create role models
- ▶ the intended tool should transform contexts into role models and vice versa
... and maybe act as a model checker resp. design advisor

① INTRODUCTION

② GRAPHICAL MODELING FRAMEWORK

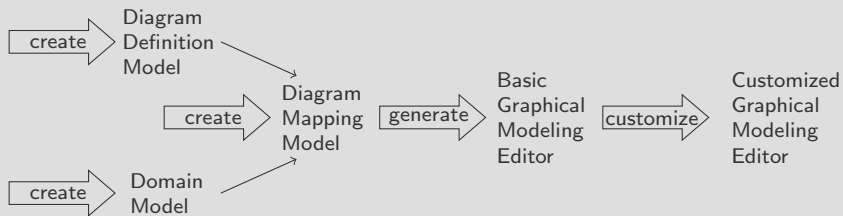
③ CORE – CONTEXTUAL ROLE EDITOR

GRAPHICAL MODELING FRAMEWORK

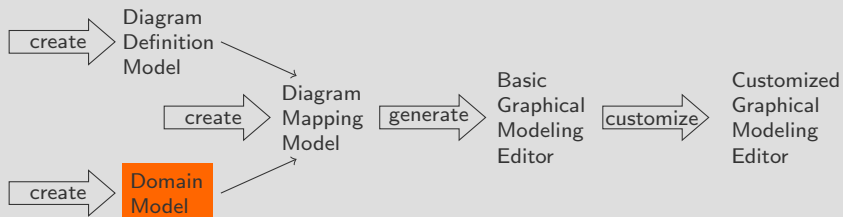
- ▶ GMF provides a set of **generative** components to develop graphical editors using Eclipse Modeling Framework (EMF) and Graphical Editing Framework (GEF)
- ▶ EMF:
 - ▶ code generation facility for building tools based on a structured data model
 - ▶ requires a model specification in XMI
 - ▶ provides tools to generate Java classes and adapter classes, that allow for viewing and editing of the model
- ▶ GEF:
 - ▶ provides technology to realize graphical editors
 - ▶ integrates these editors into Eclipse workbench

(wiki.eclipse.org)

GMF WORKFLOW



GMF WORKFLOW



Domain Model: define,

- ▶ available diagram elements
- ▶ available relations between diagram elements
- ▶ properties of diagram elements

GMF WORKFLOW

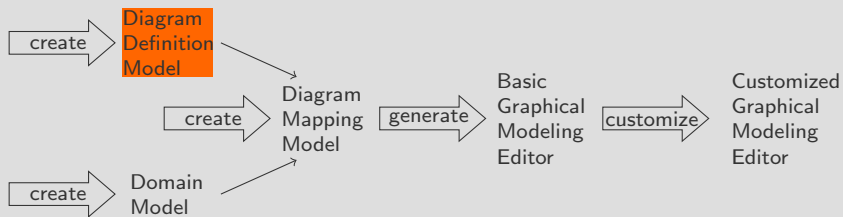


Diagram Definition Model:

- ▶ Tooling Model → available tools to create diagram elements
- ▶ Graphical Model → graphical representation of diagram elements

GMF WORKFLOW

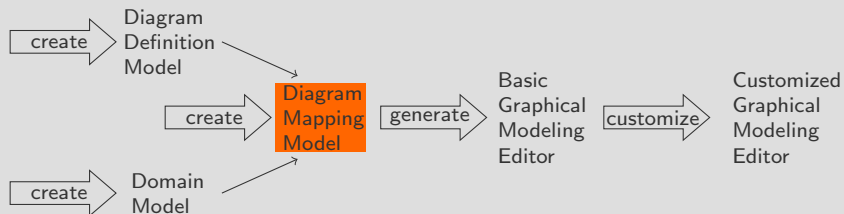
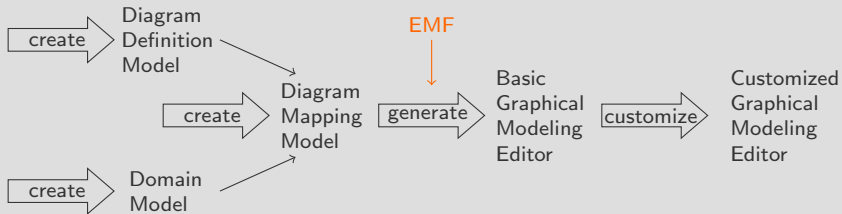


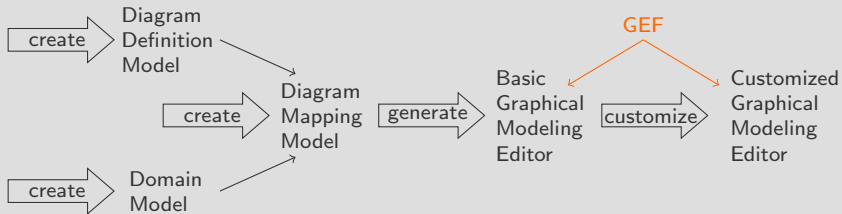
Diagram Mapping Model:

- ▶ map creation tools to graphical elements
- ▶ map diagram elements to both

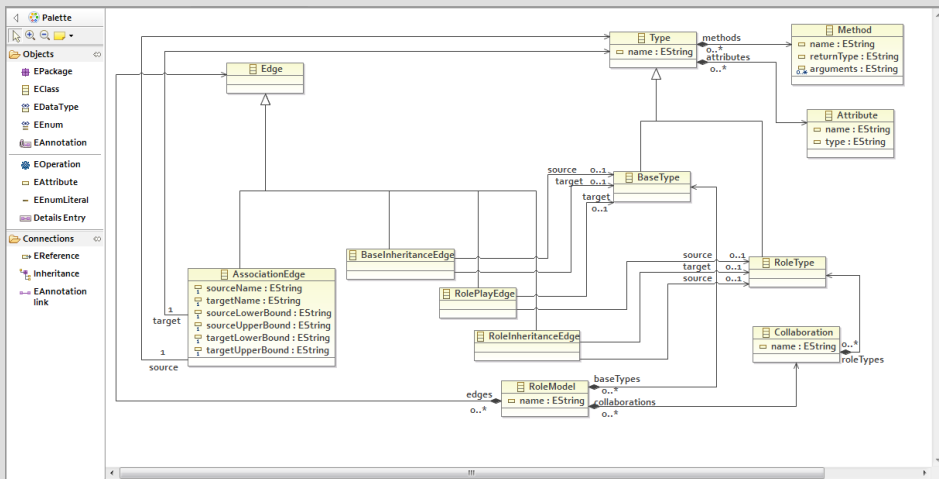
GMF WORKFLOW



GMF WORKFLOW



EXAMPLE: DOMAIN MODEL OF CORE



EXAMPLE: GMF-BASED DIAGRAM EDITOR OF CORE

The screenshot displays the Eclipse IDE interface for a GMF-based diagram editor. The main workspace shows a UML diagram titled "Lecture_gen.rd.diag". The diagram consists of several classes and their relationships:

- Lecture Package:** Contains two classes:
 - Lecturer:** Methods: `material: void`, `writeNeatly(): void`, `explainClearly(): void`.
 - Participant:** Methods: `material: void`, `grade: void`, `writeNeatly(): void`, `chatterQuietly(): void`.
- Professor Class:** Methods: `name: void`, `faculty: void`, `write(): void`, `explain(): void`.
- AssistantProfessor Class:** Inherits from **Professor**. Method: `evaluation: void`.
- Student Class:** Methods: `name: void`, `studID: void`, `write(): void`, `chatter(): void`.

Relationships:

- Lecturer** is associated with **Professor** via a relationship labeled `<<played by>>`.
- Participant** is associated with **Student** via a relationship labeled `<<played by>>`.
- AssistantProfessor** inherits from **Professor**.

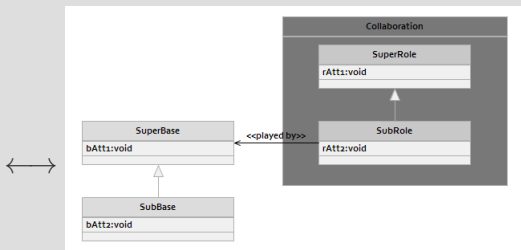
The IDE interface includes a Package Explorer on the left showing the project structure under `net.core.editor.update`. The right side features a Palette with categories like Types, Edges, and Fields. The bottom status bar indicates the current package is `net.core.editor.update`.

OUTPUT OF THE DIAGRAM EDITOR

```

<?xml version="1.0" encoding="UTF-8"?>
<roles:RoleModel
  xmlns:roles="http://net.core.editor/1.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmi:version="2.0">
  <baseTypes name="SuperBase">
    <attributes name="bAtt1" type="void" />
  </baseTypes>
  <baseTypes name="SubBase">
    <attributes name="bAtt2" type="void" />
  </baseTypes>
  <edges xsi:type="roles:BaseInheritanceEdge"
    source="//@baseTypes.1"
    target="//@baseTypes.0" />
  <collaborations name="coll">
    <roleTypes name="SuperRole">
      <attributes name="rAtt1" type="void" />
    </roleTypes>
    <roleTypes name="SubRole">
      <attributes name="rAtt2" type="void" />
    </roleTypes>
  </collaborations>
  <edges xsi:type="roles:RoleInheritanceEdge"
    source="//@collaborations.0/@roleTypes.1"
    target="//@collaborations.0/@roleTypes.0" />
  <edges xsi:type="roles:RolePlayEdge"
    source="//@collaborations.0/@roleTypes.1"
    target="//@baseTypes.0" />
</roles:RoleModel>

```



XML description

Role Model

① INTRODUCTION

② GRAPHICAL MODELING FRAMEWORK

③ CORE – CONTEXTUAL ROLE EDITOR

SOME BASICS

- ▶ CORE = **C**ontextual **R**ole **E**ditor
- ▶ it basically consists of two parts:
 1. the graphical diagram editor as a plugin for eclipse (based on EMF, GMF, GEF)
 2. a command-line tool for converting diagrams into contexts and vice versa
- ▶ this enables us to do the following:
 - ▶ generate readable UML-like diagrams from formal contexts
 - ▶ create an FCA-description of a role model in the sense of [MW10]

SOME FEATURES

- ▶ due to [GMM11] valid role play relations form bonds between the contraordinal scales of base and role types
 - ▶ a role play relation $P \subseteq B \times R$ is called **valid**, if for fixed $b \in B, r \in R$ and $\forall b' \leq b, r \leq r'$ holds $bPr \Rightarrow b'Pr'$
- ▶ thus, CORE acts as a design advisor, since it enumerates the possible role play relations, given the base and role types
- ▶ additionally, we can count the number of (proper) mergings of base and role types as well as generate the context of (proper) mergings
 - ▶ but yet, there is no direct application (other than the previous) of these mergings in the role description framework

SOME OPEN TASKS

- ▶ adding some relational features to enable typed attributes (i. e. association edges in the type diagram)
 - ▶ adding a code generator
 - ▶ adding a model checker (naïve: diff between input XML and generated XML)
-
- ▶ CORE can be found at
<http://homepage.univie.ac.at/henri.muehle/core.php>

– Demo –

Thank you.

BIBLIOGRAPHY

- [GMM11] Bernhard Ganter, Christian Meschke, and Henri Mühle.
Merging Ordered Sets.
*Proceedings of the 9th International Conference on
Formal Concept Analysis*, pages 183–203, 2011.
- [MW10] Henri Mühle and Christian Wende.
Describing Role Models in Terms of Formal Concept
Analysis.
*Proceedings of the 8th International Conference Formal
Concept Analysis*, pages 241–255, 2010.