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
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
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)

data 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

1. **Domain Model** → **create**
2. **Diagram Definition Model** → **create**
3. **Diagram Mapping Model** → **generate**
4. **Basic Graphical Modeling Editor** → **customize**
5. **Customized Graphical Modeling Editor**
**GMF Workflow**

- **Domain Model**: define,
  - available diagram elements
  - available relations between diagram elements
  - properties of diagram elements

Diagram Definition Model → Diagram Mapping Model → Basic Graphical Modeling Editor → Customized Graphical Modeling Editor

- create Domain Model
- create Definition Model
- create Mapping Model
- generate Basic Graphical Modeling Editor
- customize Customized Graphical Modeling Editor
GMF Workflow

Create Domain Model → Create Diagram Definition Model

Create Diagram Mapping Model → Generate Basic Graphical Modeling Editor

Customize Graphical Modeling Editor

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**

- **Create Domain Model**
- **Create Diagram Definition Model**
- **Create Diagram Mapping Model**
- **Generate Basic Graphical Modeling Editor**
- **Customize Graphical Modeling Editor**

**EMF**
**GMF Workflow**

- **Create Domain Model**
- **Create Diagram Definition Model**
- **Create Diagram Mapping Model**
- **Generate Basic Graphical Modeling Editor**
- **Customize Graphical Modeling Editor**

**GEF**
Example: Domain Model of Core
EXAMPLE: GMF-BASED DIAGRAM EDITOR OF CORE
Output of the Diagram Editor

<?xml version="1.0" encoding="UTF-8"?>
<roles:RoleModel
 xmlns:roles="http://net.core.editor/1.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:version="2.0">
 <baseTypes>
  <baseTypes name="SuperBase">
   <attributes name="aAtt1" 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="coll1">
  <roleTypes name="SuperRole">
   <attributes name="cAtt1" type="void"/>
  </roleTypes>
  <roleTypes name="SubRole">
   <attributes name="cAtt2" 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>

XMI description

Role Model

Henri Mühle
Some Basics

Core = Contextual Role Editor

- 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
