GME-MOF: The MOF-Based GME Metamodelling Environment

Matt Emerson
10/6/2004

Advisor:
Dr. Janos Sztipanovits
Motivation

- The core of MIC is the development of DSMLs.
- GME provides MetaGME, a metamodelling language for DSLM specification.
- The OMG has adopted MOF as the standard metamodelling language.

- What is the technical merit of the MOF standard vs. MetaGME?
- How can GME adapt to meet this standard with minimum pain?
Outline

- Overview of GME and MetaGME
- Overview of MOF
- Evaluation of MOF vs. MetaGME
- GME-MOF: The MOF-based GME metamodeling environment
GME

- Meta-programmable modeling tool
- Platform for the development of many model-based embedded systems tools
- Includes a metamodeling language which predates MOF: **MetaGME**
GME Modeling Hierarchy

Meta-Modeling Language
Meta-Meta-Model

Meta-Modeling Language
Meta-Model

specify

Domain Modeling Language
Model

specify

specify

Computer-Based System

Example: FSM Meta-Model

Example: FSM Model

stateMachine

state

transition

start

A

B

C

stop

metaTo

metaFrom
MetaGME

- Based on UML Class Diagrams and OCL
- Class Stereotypes imply the abstract syntax of elements in the metamodel:
  
  - `<<Model>>` : Compositional containers
  - `<<Atom>>` : Primitive objects
  - `<<Reference>>` : Pointers to model objects
  - `<<Set>>` : Aggregate containers
  - `<<Connection>>` : Analogous to UML Association Classes
  - `<<Aspect>>` : Logical visual partitions of a system
MOF

- OMG Standard metamodeling language
- Simplification of UML Class Diagrams
- Used as definition language for many OMG specifications:
  - UML
  - CWM
  - OCL
  - XMI
- One use case is the specification of DSMLs
MOF vs. MetaGME: Technical Merit

- MOF Operations, Parameters, and Exceptions
  - Specify interface to operational semantics
  - Aids creation of model interpreters

```cpp
// ***************************************************************
// CLASS ElevatorImpl
// ***************************************************************
class ElevatorImpl : public BON::ModelImpl
{
public:
  // attribute getters
  virtual int getCurrentFloor();

  // MOF-Specified Interface
  virtual void ascendRequest(int floor);
  virtual void descendRequest(int floor);
};
```
Superior facilities for metamodel reuse

- Package Importation as well as Generalization
  - MOF Package Importation allows type re-use for any purpose but instantiation
  - MOF Package Generalization is for metamodel extension
  - MetaGME Relies on GME Library Import (similar to MOF Package Generalization)

- Package Merge in MOF 2.0
  - Intelligent algorithm for merging metamodels
  - Identifies metamodel “join points” as metamodel elements with identical names
  - MetaGME requires case-by-case identification of join points
MOF vs. MetaGME: Technical Merit

- MOF lacks an easy facility for specifying modeling language concrete Syntax
  - No way to specify how models are visualized
  - Aids in tool independence
  - Inconvenient when working in GME
- No multi-aspect modeling
  - Cannot logically partition systems concerns at the system modeling level
GME-MOF: Design Forces

- Tight correspondence between MetaGME stereotypes and GME configuration file entities
- Existing tools support, including:
  - Constraint Checker
  - Metamodel Interpreter
- Existing user base and modeling languages
- Kinship of MOF and MetaGME
  - Both based on UML Class Diagrams
- Leverages existing metamodelling language
- Leverages existing metamodel translation tool
- MOF provides an MDA-style interface for GME Metamodelling
### MOF-to-MetaGME Translation

<table>
<thead>
<tr>
<th>MOF Concept</th>
<th>MetaGME Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Folder</td>
</tr>
<tr>
<td>Class</td>
<td>Model, Atom, Set, Reference</td>
</tr>
<tr>
<td>Aggregate Association</td>
<td>Composition</td>
</tr>
<tr>
<td>Non-Aggregate Association</td>
<td>Connection, SetMembership, ReferTo</td>
</tr>
<tr>
<td>String-typed Attribute</td>
<td>FieldAttribute</td>
</tr>
</tbody>
</table>

- Model-to-model translation
- Design using the Graph Rewriting and Transformation Language (GReAT)
- Guided by MOF Tags
- Not isomorphic
Specifying the Translation

Mapping MOF Primitive-type Attributes to MetaGME FieldAttributes

- Match the pattern visualized in black.
- Create the pattern visualized in blue.

OMG MIC Workshop
How to meaningfully represent in GME:
- Singleton Classes
- Derived Attributes
- Derived Associations
- Attributes of arbitrary types
- MOF-style metamodel importation
Translation Example: UML Class Diagrams

Input (MOF)

Output (MetaGME)

ClassDiagram

ClassBase
- IsAbstract: Boolean
- ContainsClassBase
  - Constraint
    - ConstraintDescription: String
    - ConstraintEqn: String

Attributes: String
Stereotype: String

RefersTo

HasConstraint

Composition

Constraint

ConstraintEqn: field
ConstraintDescription: field

HasConstraint

ClassDiagram

ClassBase
- IsAbstract: field

ClassCopy

Class

Composition

Constraint

ConstraintEqn: field
ConstraintDescription: field

ClassCopy

Class

Stereotype: field
Attributes: field

OMG MIC Workshop

Matt Emerson
GME-MOF: An MDA Interface for GME

- Leverages existing metamodeling language
- Leverages existing metamodel translation tool
- MOF provides an MDA-style interface for GME Metamodeling
- Solution illustrates the versatility of metamodeling and meta-programmable tools
GME-MOF: The MOF-Based GME Metamodelling Environment

Matt Emerson
10/6/2004

Advisor:
Dr. Janos Sztipanovits