Better Metadata Management through Better Metamodels

Issues to consider when developing a metamodel, and why you might care

GK Khalsa
khalsa@objectrad.com
Objectrad, Temecula, CA

Discussion

• The role of metamodeling in application development & metadata management
• The MOF for metamodels
• Implications of MOF on metadata design
• Techniques for better metamodels
• Issues for MOF, UML, XMI, JSR-40 etc.
The Role of the Metamodel

«insert typical 4-layer metamodel diagram here»

Why We Metamodel

- Define UML
- Provide XMI – XML representation of metadata
- Define other metadata types
- Manage metadata uniformly, build links & transformations among various islands
- Extend UML and other metadata definitions
MOF Metamodeling

- MOF defines the language for metamodels (meta-metamodel)
- MOF defines APIs for defining and managing metamodels
- MOF defines a metamodel to API mapping
  - defines APIs for defining and managing metadata (from any metamodel)

What Should a Metamodel Look Like?

- MOF is close to a subset of UML
- Transformations from UML to MOF exist
  - including implementations
  - transformations are not published
- So, meta-modeling is just like modeling, right?
Logical vs. Physical Metamodel

MOF Metamodel to Interface Mapping
Metamodeling Strategy

- Should a modeler
  - define logical model & refine to more physical
  - just define the best model for an implementation
  - define the best logical model, & ignore mapping issues
Featureless Subtypes

Forces:
Each of a number of subtypes have no distinct features

MOF Mapping for Subtypes

«USE»
MOF Interface Mapping
«Create»
12 Interfaces
Alternative Modeling

Technique:

Remove the Subtypes
Make the base type non-abstract
Add an attribute to the base type (enum or string) to represent the specific subtype

Mapping the Alternative
Composition for Attribution

Forces:

Composition defined
No reference (navigability) to parent

Mapping the Composition

4 Interfaces representing the two Classes

1 Datatype representing the link

Interface representing the Association, providing access to links
Composition Alternative

Technique:
- Remove the Association
- Remove the Reference
- Represent with an Attribute

Mapping the Composition Alternative

4 Interfaces representing the two Classes
Referencing with Associations

Forces:
- Non-composite Association
- Reference (navigability) only one way
- Reference owner’s end’s cardinality is optional

Mapping the Association

- MOF Interface Mapping
  - «datatype»: 1 Datatype representing the link
  - «create»: Association, providing access to links
  - 4 Interfaces representing the two Classes
Attribute & Data Type Alternative

Technique:
- Remove the Association
- Remove the Reference
- Define an Alias Data Type to represent the referenced Class
- Create an attribute of this type

Mapping the Association Alternative
Other Physical Model Optimizations

- Overlapping Associations
- Efficient AssociationClass Removal
- Unnecessary Packaging
- Constraints to Reduce Model Elements

The Effect of Thoughtful Metamodels

<table>
<thead>
<tr>
<th>Element</th>
<th>UML Metamodel</th>
<th>After Transformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Types (Classes and DataTypes)</td>
<td>128</td>
<td>102</td>
</tr>
<tr>
<td>Associations</td>
<td>105</td>
<td>65</td>
</tr>
<tr>
<td>Java source lines</td>
<td>261,935</td>
<td>137,799</td>
</tr>
</tbody>
</table>
Why Can’t a MOF Do This?

- Can these transformations be automated?
- Compares to Transformational Programming approaches?

Why Not Change the Mapping?

- Just change the mapping to reduce the result; or
- Define MOF tags that shape the mapping
  - Physical model encoded in the logical model
The Point

• If your model will be used in a MOF/XMI/JSR-40 technology, think about the implementation implications
• The MOF-RTF & major revisions can address this issue
• The UML-RTF & major revisions should address this issue