Lecture 20: Applications of Metamodelling

Generic Lecture

Peter Thiemann

University of Freiburg, Germany

26.07.2012
Feature models are a tool for domain analysis
▶ Provide a hierarchical view of features and their dependencies
▶ Establish an ontology for categorization

Visualized by feature diagrams


Popularized for Generative Programming by Czarnecki and Eisenäcker

Also for analyzing other domains
Feature Modeling

Example

- Hierarchical, but **not** is-a relation (as in a class diagram)
- Features may be qualified as required, optional, alternative, or $n$-of-$m$ (selection)
Feature Modeling

MOF-based Metamodel

MOF::Class

FM::Feature

FM::SubfeatureGroup

FM::Concept

MOF::Attribute

type:String
value:String

FM::GroupKind

inv:value=="required"||value=="optional"||
value=="alternative"||value=="nOfM"
inv:parent==null

FM::Concept

inv:parent==null

inv:type=="String"
Feature Modeling

Feature Model in Abstract Syntax

Feature Modeling
Extended Metamodel and Concrete Syntax

New feature ⇒
- new attribute in metamodel
- new slot in model
- extension of concrete syntax
Applications of Metamodelling
Component Modeling

- Domain specific modeling language for small and embedded systems
- Main abstraction: component
- A component may
  - provide services via interfaces
  - require services via interfaces
  - have configuration parameters
  - be an application (does not provide services)
Component Modeling

Example
Component Modeling

Simple Component Metamodel
Component Modeling

MOF-based Simple Component Metamodel
Pitfalls in Metamodelling

How to avoid

- confusion with UML notation
- mixing metalevels

Central question

- what is the mapping to a programming language?
Interfaces

Constraint: Every instance of **Entity** should implement **SomeInterface**

▶ wrong approach

▶ book solution use OCL or subsetting of metaassociation
Pitfalls in Metamodeling

Interfaces/2

Constraint: Every instance of **Entity** should implement **SomeInterface**

▶ **correct solution** use OCL

```
Entity

:Entity

realization

-->select(hasStereotype("interface"))

-->select(name="SomeInterface")

-->size() = 1

<<interface>>

SomeInterface
```
Dependency

Problem: A Component may depend from multiple Interfaces because the Component may invoke operations of the Interfaces.

Wrong approach “metaclass Component depends on metaclass Interface”

Correct solution a metaassociation “uses”
Identifying Attribute

An **Entity** must have an identifying attribute with name **ID** and type **String**. **Entity** is a subclass of **UML::Class**.

▶ **wrong approach**

<table>
<thead>
<tr>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID : String</td>
</tr>
</tbody>
</table>

defines a tagged value **ID** for all **Entity** instances in the model
Identifying Attribute

- **correct solution**

```plaintext
Entity

(Attribute->select( Name = "ID")->size=1)
and
(Attribute->select( Name = "ID")->forAll( Type.Name = "String")
)
```

- there must be exactly one attribute with name ID
- all attributes named ID must have type String
Identifying Attribute

- **correct solution**

```
Entity
```

```
(Attribute->select(Name= "ID")->size=1)
and
(Attribute->select(Name= "ID")->forall(
Type.Name = "String"
))
```

- there must be exactly one attribute with name ID
- all attributes named ID must have type String

- **incorrect attempt**

```
context Entity inv:
Attribute
  ->select (Name="ID" and Type.Name="String")
  ->size() = 1
```
Primary Key Attribute

*Each instance of **Entity** must have exactly one attribute of type **EntityPK**, where **EntityPK** is a subclass of **Attribute**.*

**Wrong approach**

- \( pk : \text{EntityPK} \)

**Correct solution**

- \( \{\text{subsets Feature}\} \rightarrow \ast \)
- \( 1 \rightarrow \{\text{subsets Attribute}\} \)

\( \text{Entity} \rightarrow \text{UML::Class} \)

\( \text{Attribute} \rightarrow \text{UML::Attribute} \)

\( \text{EntityPK} \rightarrow \text{Entity} \)
Metalevels and Instanceof

- Objects are instances of classes
- Links are instances of associations
Metalevels and Instanceof
Model Elements as Instances of Metamodel Elements

- **Car** and **Person** classes are instances of the metaclass \texttt{UML::Class}
- **me:** and **myFather:** are instances of the metaclass \texttt{UML::Object}
- How can that be?
Metalevels and Instanceof

A Look at the Metamodel

- two different instanceof relations
Summary

- Metamodelling required for customizing UML
- OMG relies on MOF to define profiles
- OCL defines static semantics of models
- Metalevels should not be confused