

Applications of Metamodeling

Feature Modeling

Software Engineering Model Driven Architecture Applications of Metamodeling

Prof. Dr. Peter Thiemann

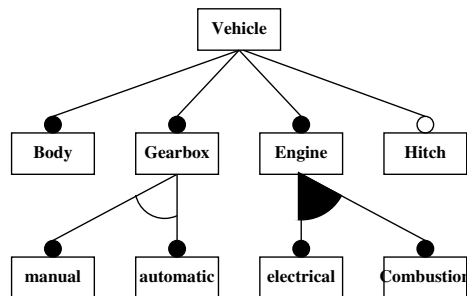
Universität Freiburg

21.07.2011

- ▶ 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
- ▶ Conceived for software domain analysis: Kang, Cohen, Hess, Novak, Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical report CMU/SEI-90-TR-21. 1990.
- ▶ Popularized for Generative Programming by Czarnecki and Eisenäcker
- ▶ Also for analyzing other domains

Feature Modeling

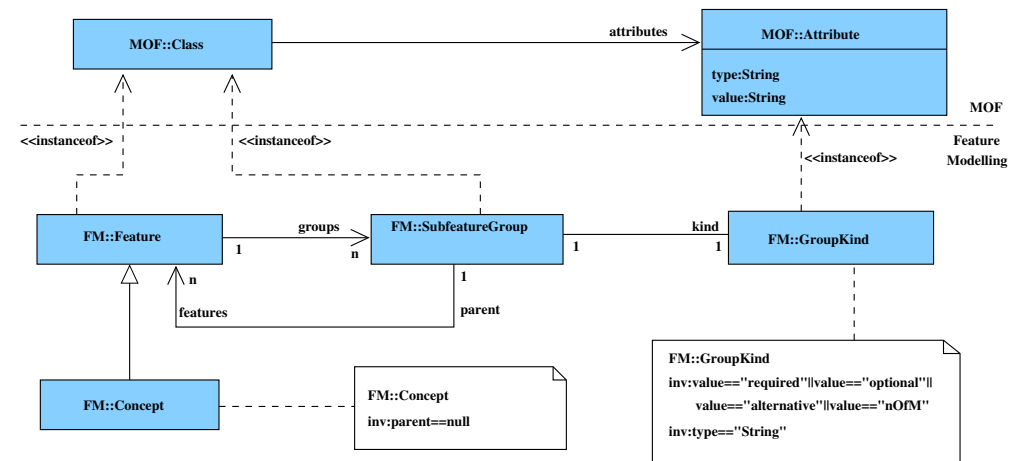
Example



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

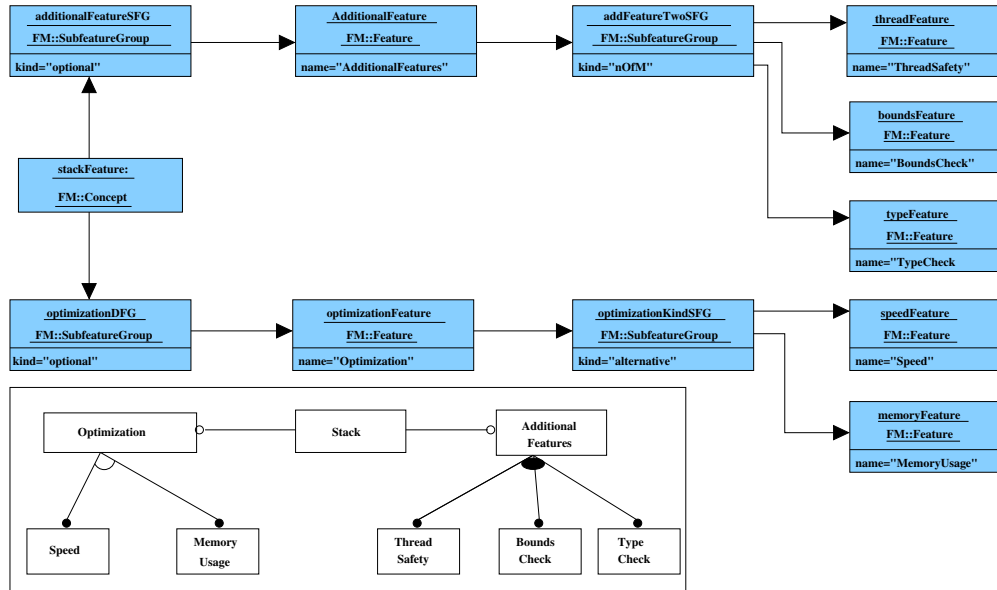
Feature Modeling

MOF-based Metamodel



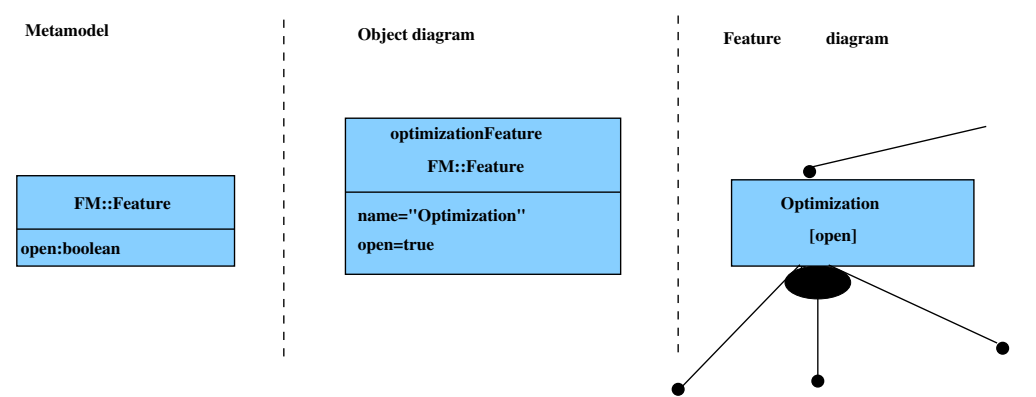
Feature Modeling

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

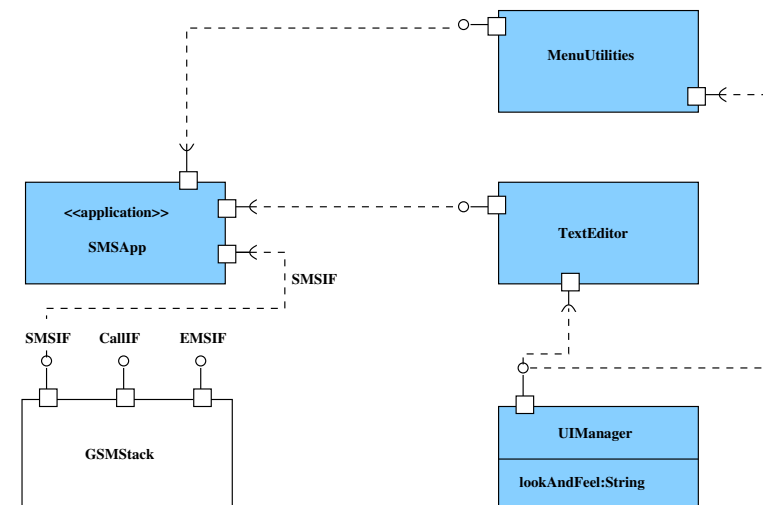
Applications of Metamodeling

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)

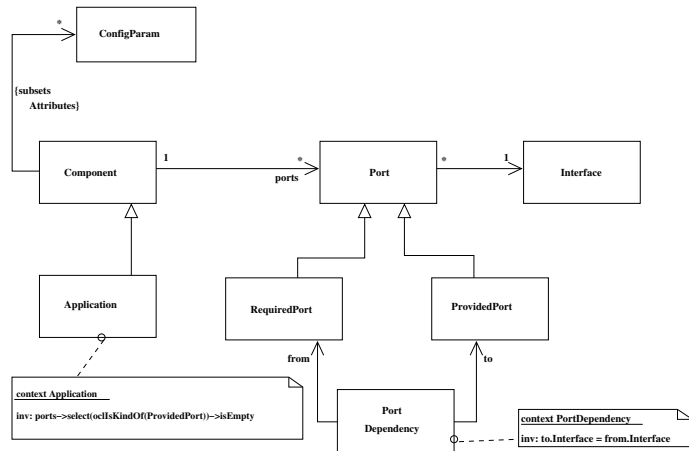
Component Modeling

Example



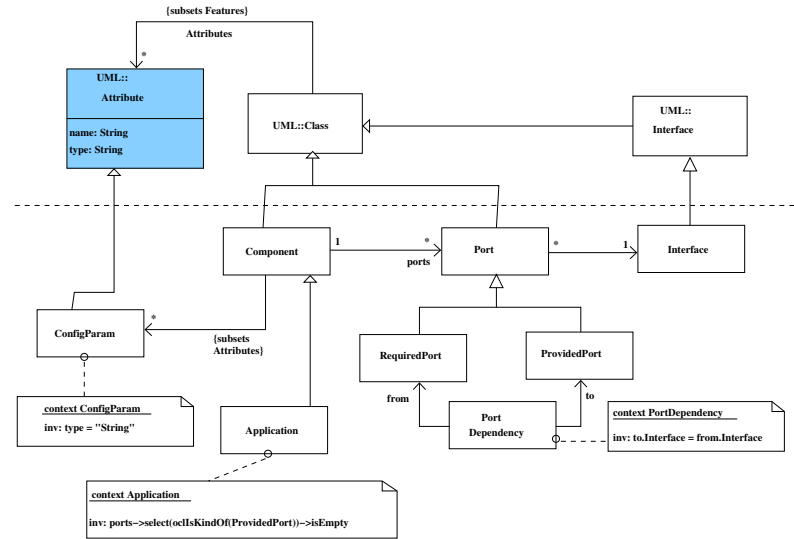
Component Modeling

Simple Component Metamodel



Component Modeling

MOF-based Simple Component Metamodel



Pitfalls in Metamodeling

How to avoid

- ▶ confusion with UML notation
- ▶ mixing metalevels

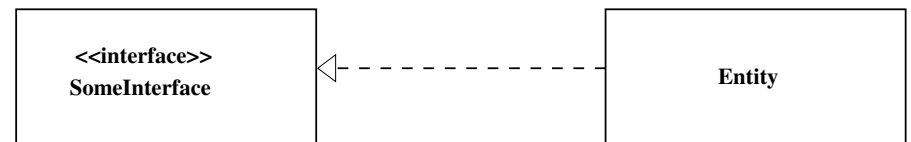
Central question

- ▶ what is the mapping to a programming language?

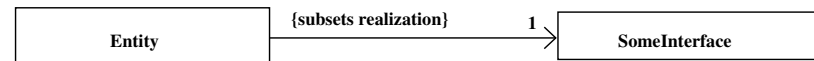
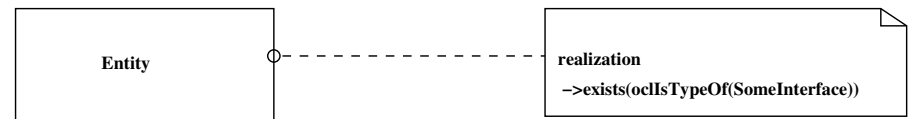
Interfaces

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

- ▶ **wrong approach**



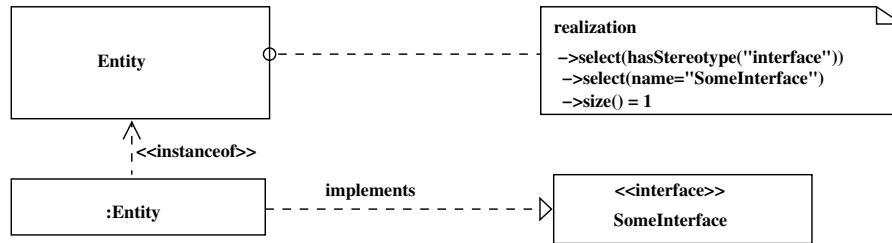
- ▶ **book solution** use OCL or subsetting of metaassociation



Interfaces/2

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

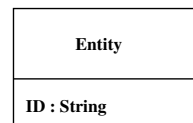
- ▶ correct solution use OCL



Identifying Attribute

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

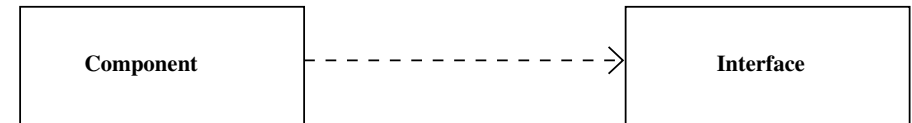
- ▶ wrong approach



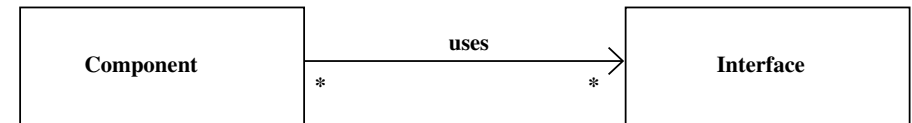
defines a tagged value ID for all **Entity** instances in the model

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**”

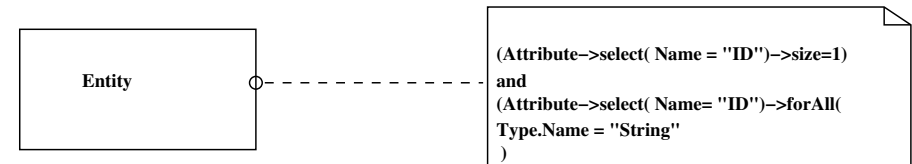


- ▶ correct solution a metaassociation “uses”



Identifying Attribute

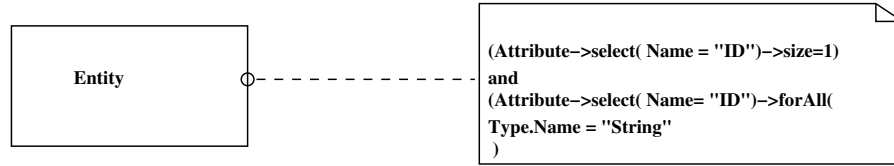
- ▶ correct solution



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

Identifying Attribute

▶ correct solution



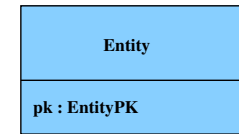
- ▶ there must be exactly one attribute with name ID
- ▶ all attributes named ID must have type String

▶ incorrect attempt

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

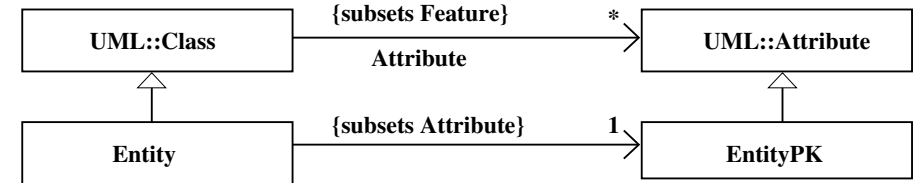
Primary Key Attribute

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

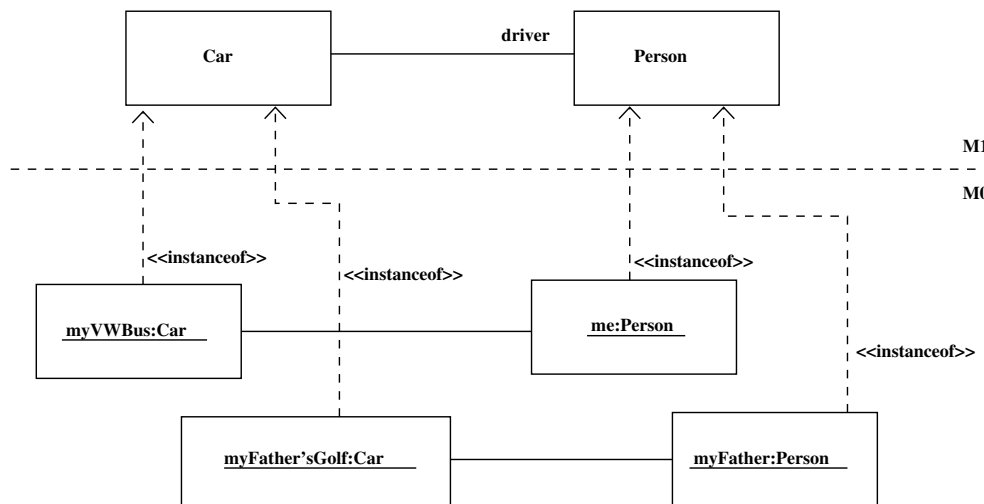


▶ wrong approach

▶ correct solution



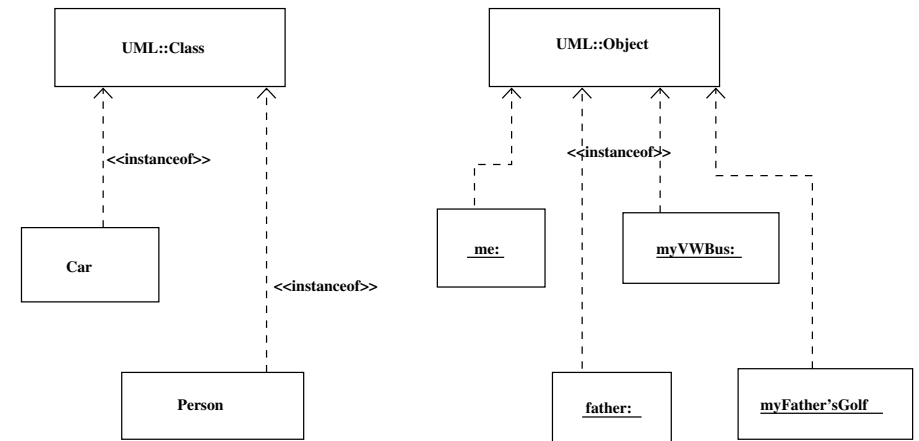
Metalevels and Instanceof



- ▶ Objects are instances of classes
- ▶ Links are instances of associations

Metalevels and Instanceof

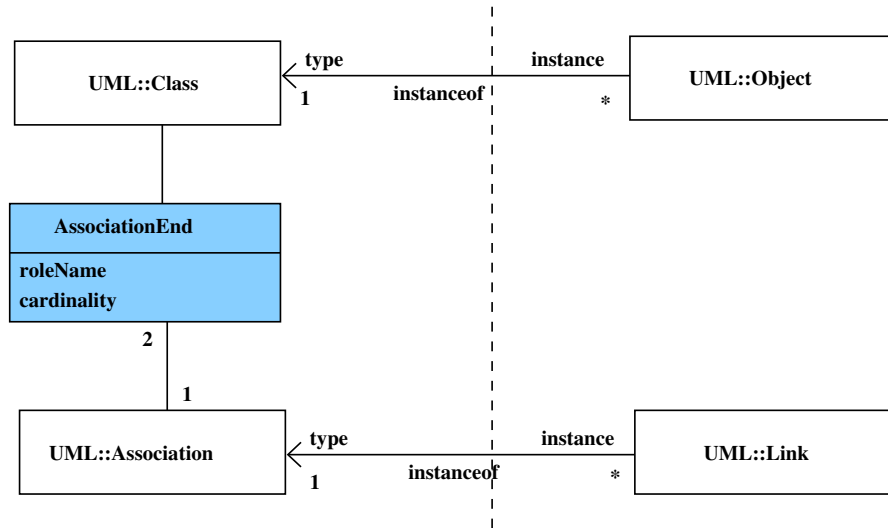
Model Elements as Instances of Metamodel Elements



- ▶ The **Auto** and **Person** classes are instances of the MOF metaclass **UML::Class**
- ▶ The objects **me:** and **myFather:** are instances of the MOF metaclass **UML::Object**

Metalevels and Instanceof

A Look at the Metamodel



► ⇒ two different instanceof relations

Summary

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