Software Architecture: Meta-modeling in UML2

Uwe Zdun
Software Architecture Research Group
Faculty of Computer Science
University of Vienna
http://cs.univie.ac.at/swa

UML2: METAMODEL

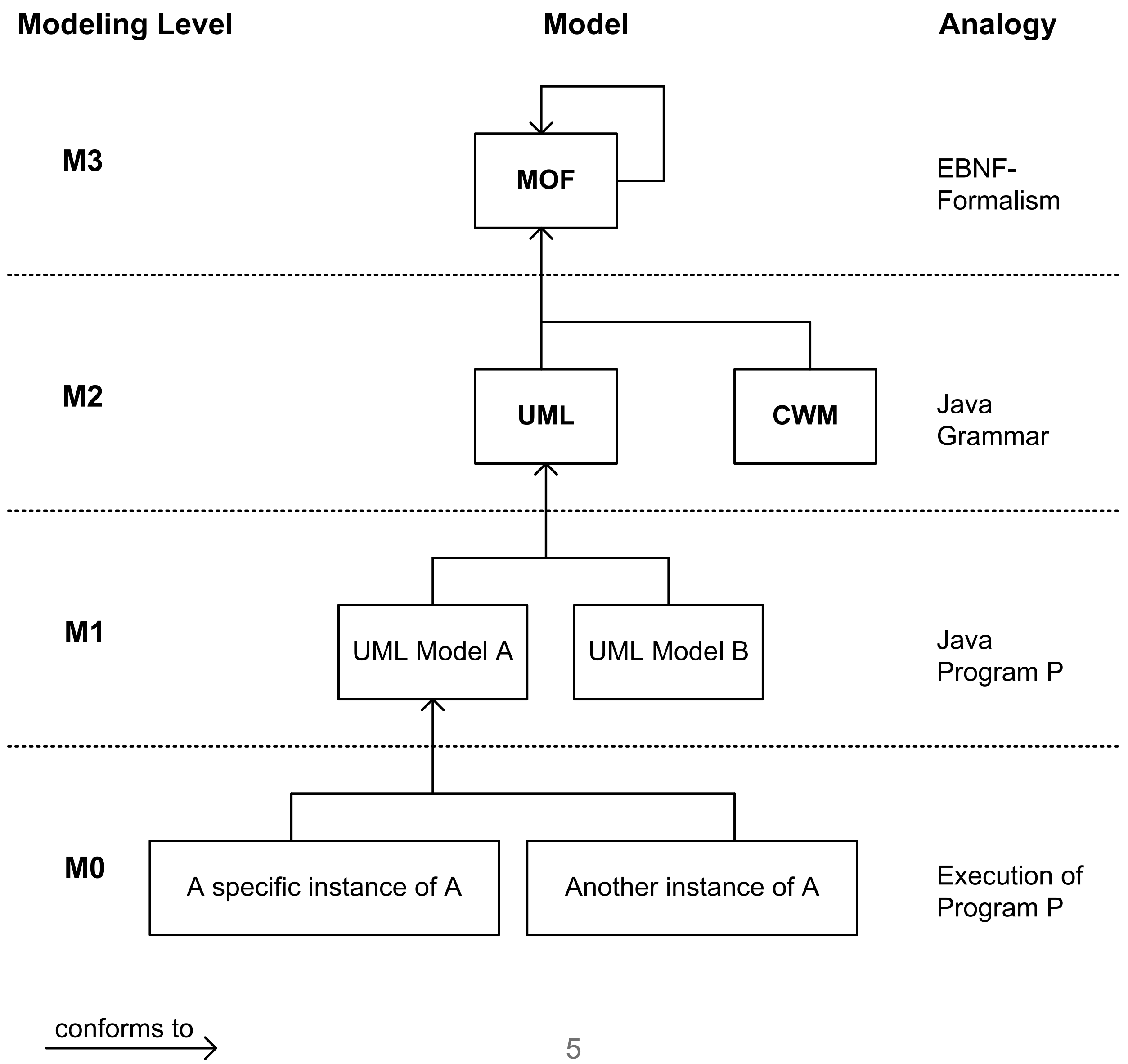
Meta-modeling

- To be able to **automatically process model data**, it must be **described precisely** in a formal language
- To achieve this, models are typically described through model, which are then called **meta-models**
- A model is hence the **instance** of its meta-model
- The cascade of abstraction by creating a meta-model for models can be continued arbitrarily, leading to a number of **modeling-levels**

Unified Modeling Language (UML): MOF

- The Meta Object Facility (MOF) provides basic meta-meta model elements to build meta-models
- The OMG's meta-modeling architecture defines 4 modeling levels
- It is itself defined using the elements of the UML infrastructure

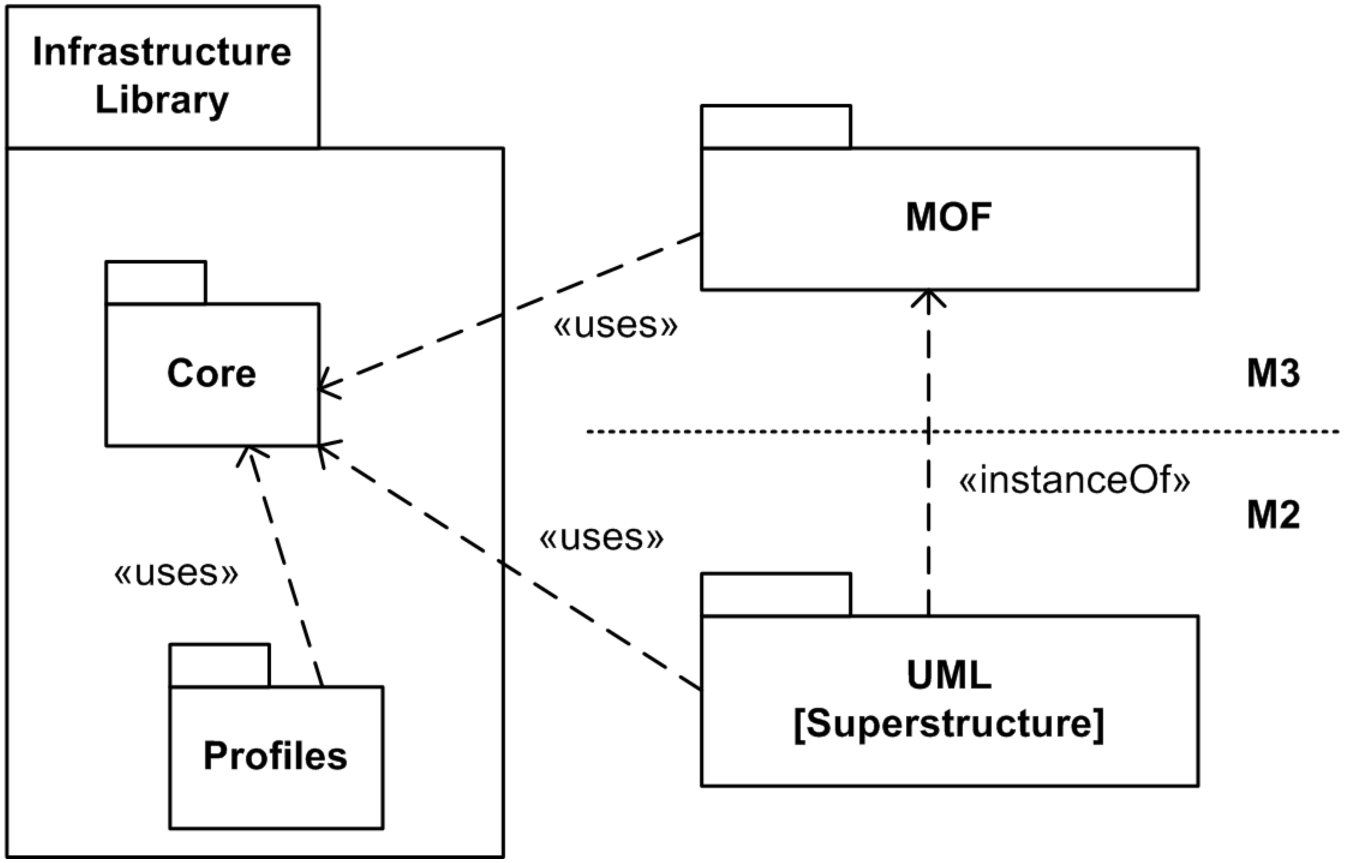
Four modeling levels of the OMG



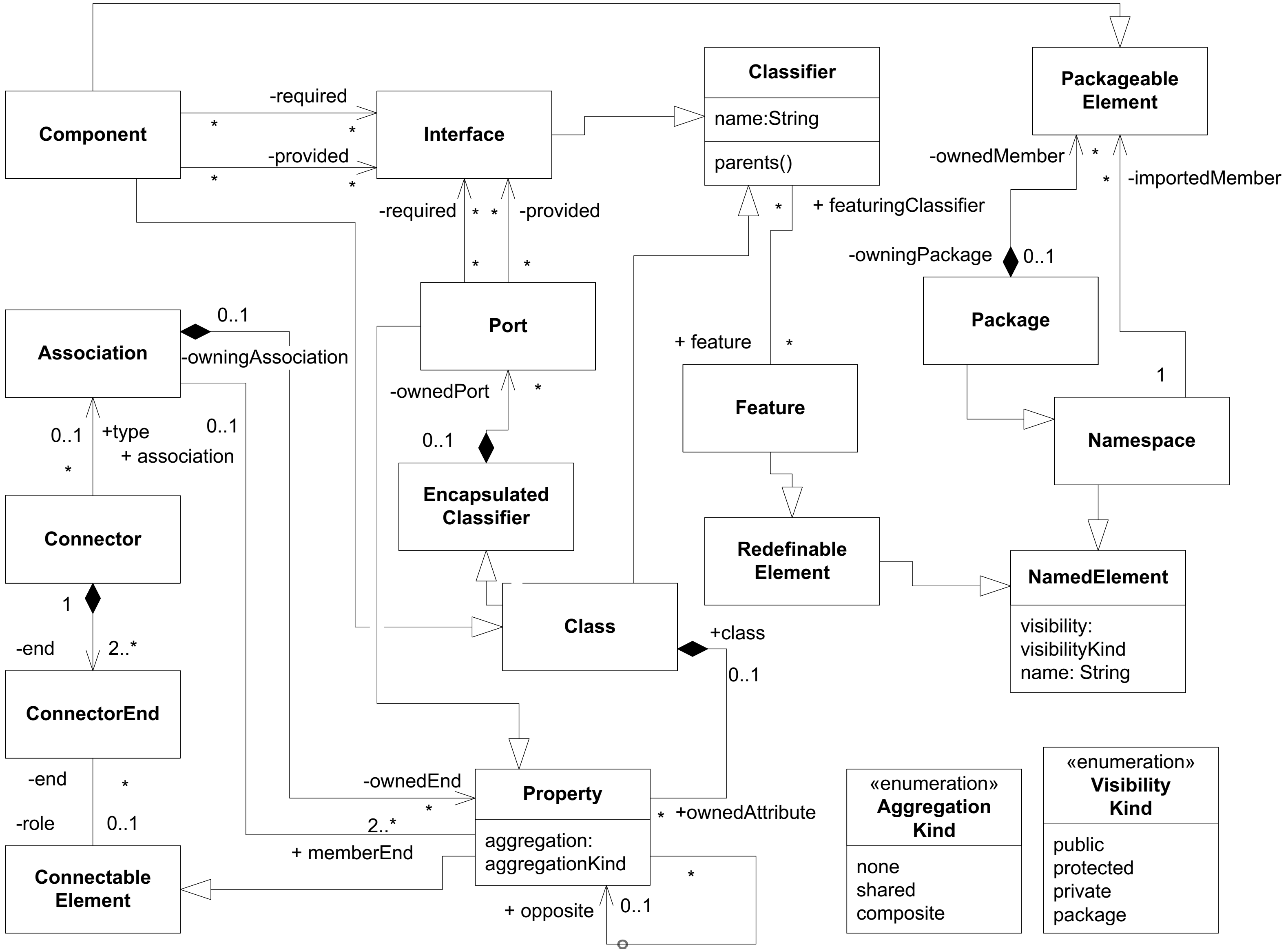
Unified Modeling Language (UML): MOF

- The UML specification itself is split into the **UML Infrastructure** and the **UML Superstructure** specifications
- The **UML infrastructure** defines elements used in both the meta-meta-model of UML (MOF) and the superstructure
- The **UML meta-model** (i.e. the language definition) is defined in the **UML superstructure**
  - The infrastructure is **merged** into the superstructure

Application of the UML Infrastructure in MOF and UML



Example: UML2 Meta-Model Excerpt for Defining Component Architectures



EXTENDING THE UML

Extending the UML Meta-Model

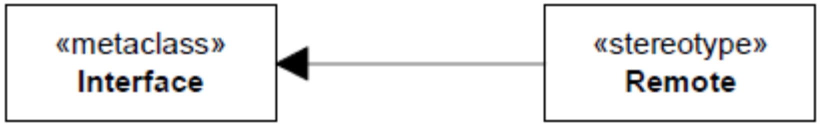
- According to the UML standard there are two ways to extend the language:
  - the hard extension produces an **extension of the language meta-model**, i.e., a new member of the UML family of languages is specified
  - the soft extension results in a **profile**, which is a set of **stereotypes**, **tag definitions**, and **constraints** that are based on existing UML elements with some extra semantics according to a specific domain

Defining UML Profiles

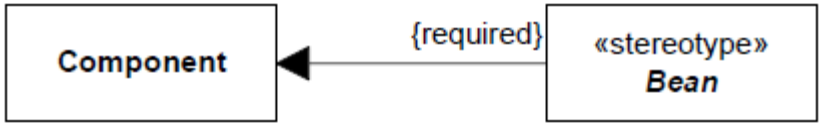
- A **stereotype** can extend any element (**meta-class**) of the meta-model
  - new types of classes, components, actors, …
  - new types of relationships
  - new features like attributes
- (OCL) **Constraints** can be used to formally define the semantics of the stereotyped meta-classes
- Stereotypes can have a **custom image** for the concrete syntax

Extension Relationship: Defining Stereotypes

- UML meta-classes are extended using the **extension** relationship



- Extensions can be **required**
  - Models, for which the profile is applied, are not well-formed unless the stereotype is applied
  - Used to express extensions that should always be present for all instances of the base metaclass

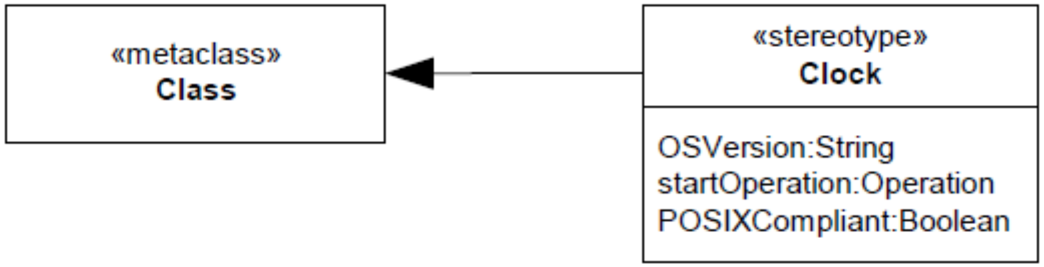


Stereotype Icons

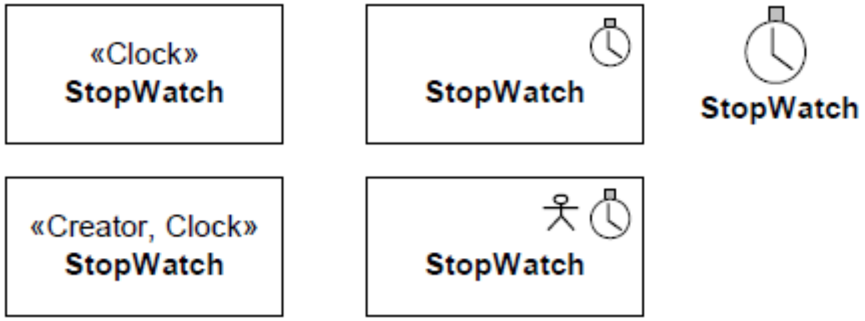
- When a stereotype includes the definition of an **icon**, this icon can be **graphically attached to the model elements extended by the stereotype**
- Every model element that has a graphical presentation can have an attached icon
  - **Boxes**:
    - The box can be replaced by the icon, and the name of the model element appears below the icon
    - The icon can be presented in a reduced shape, inside and on top of the box representing the model element
  - **Links**: the icon can be placed close to the link
  - **Textual notation**: the icon can be presented to the left of the textual notation

Example

- Defining a stereotype:



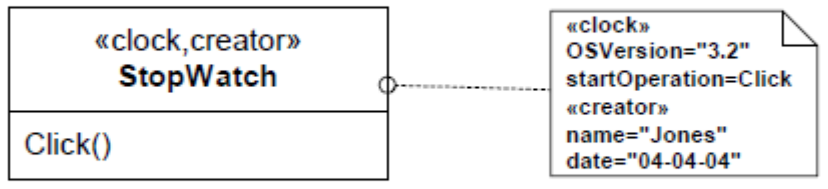
- Presentation options for the extended class:

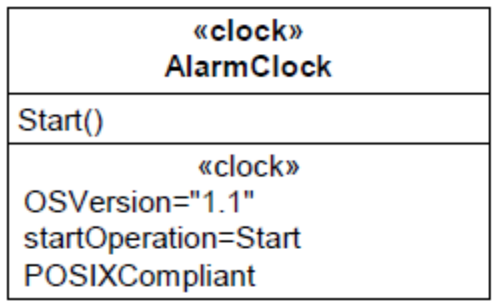


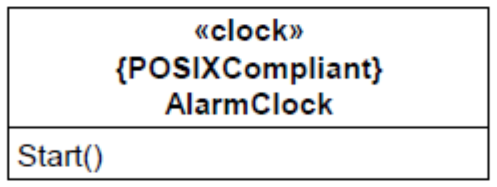
Tagged Values

- Just like a class, a stereotype may have **properties**, which may be referred to as **tag definitions**
- When a stereotype is applied to a model element, the values of the properties may be referred to as **tagged values**
  - In UML 2.0, a tagged value can only be represented as an attribute defined on a stereotype
  - Therefore, a model element must be extended by a stereotype in order to be extended by tagged values

Notation Options: Showing Tagged Values on the Extended Class





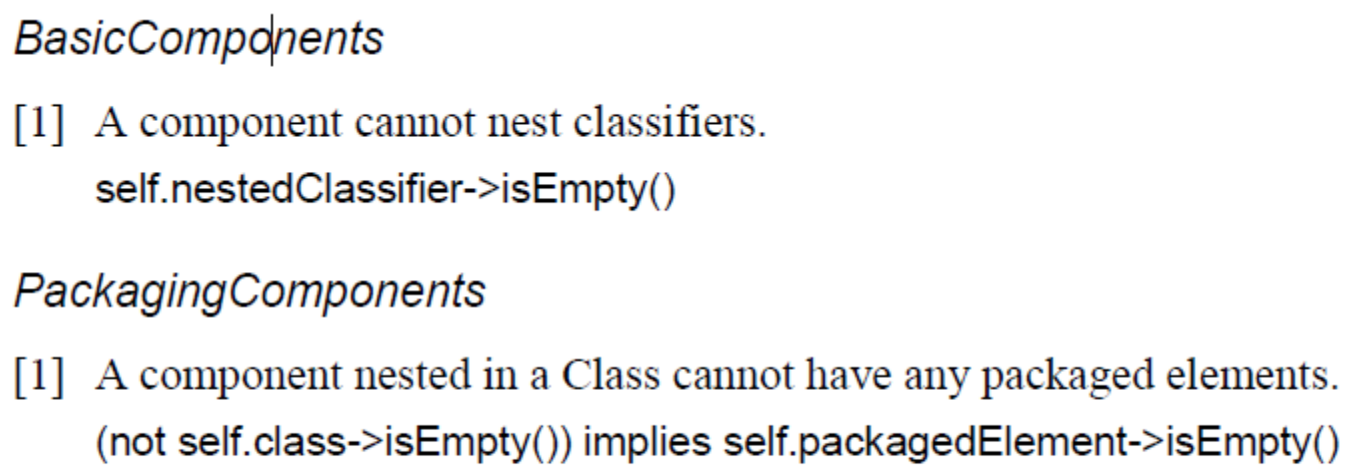


Constraints

- **Every Element** in UML2 can have **constraints**
  - Constraints are not restricted to profiles or meta-model extensions, but often used for defining them
- The UML standard uses informal, **textual constraints**, as well as formal **OCL constraints**
  - More on OCL: OCL Specification, www.omg.org/cgi-bin/doc?ptc/2003-10-14

Constraints: Example

From the definition of the Component meta-class in the UML standard:



BasicComponents

[1] A component cannot nest classifiers.

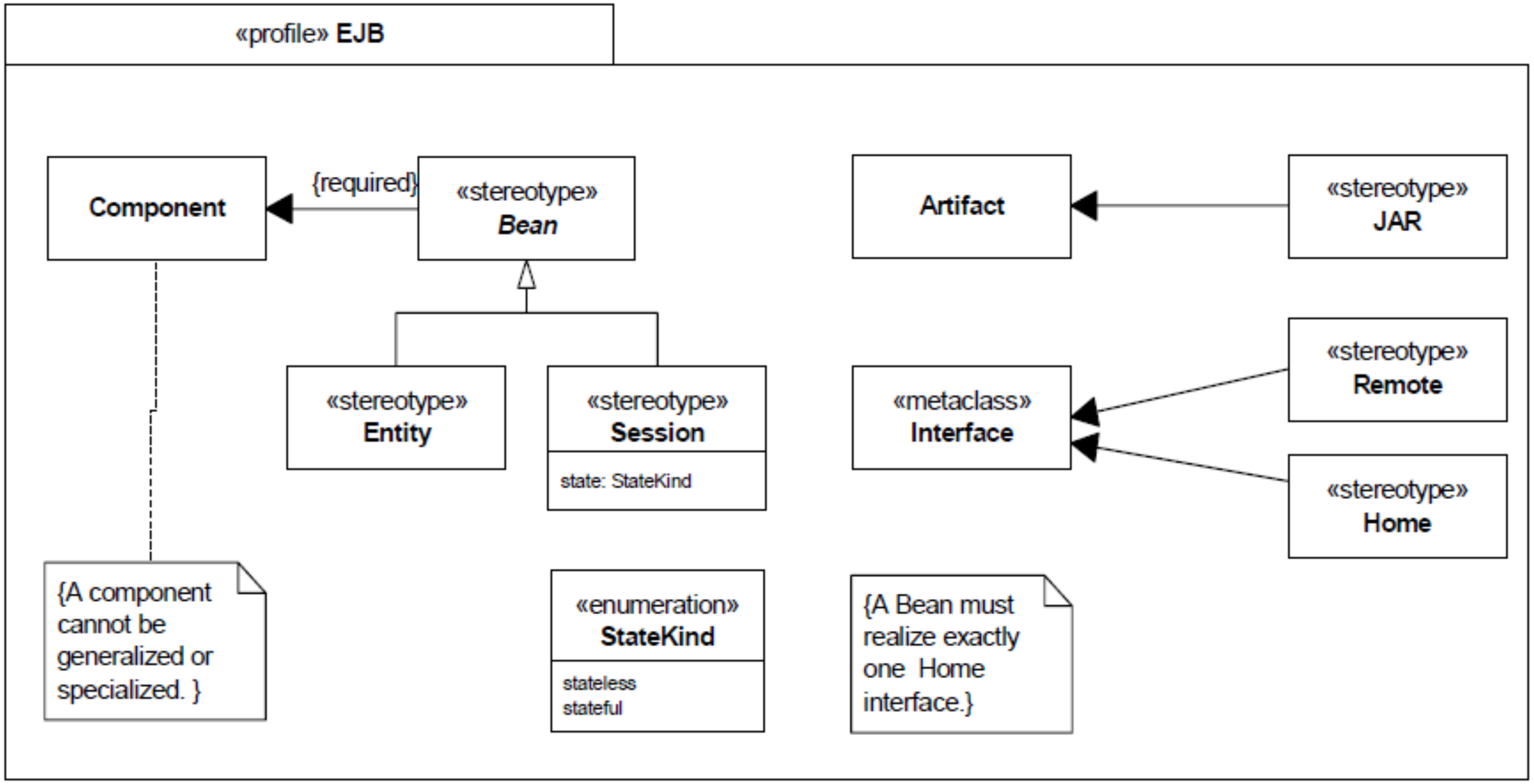
```
self.nestedClassifier->isEmpty()
```

PackagingComponents

[1] A component nested in a Class cannot have any packaged elements.

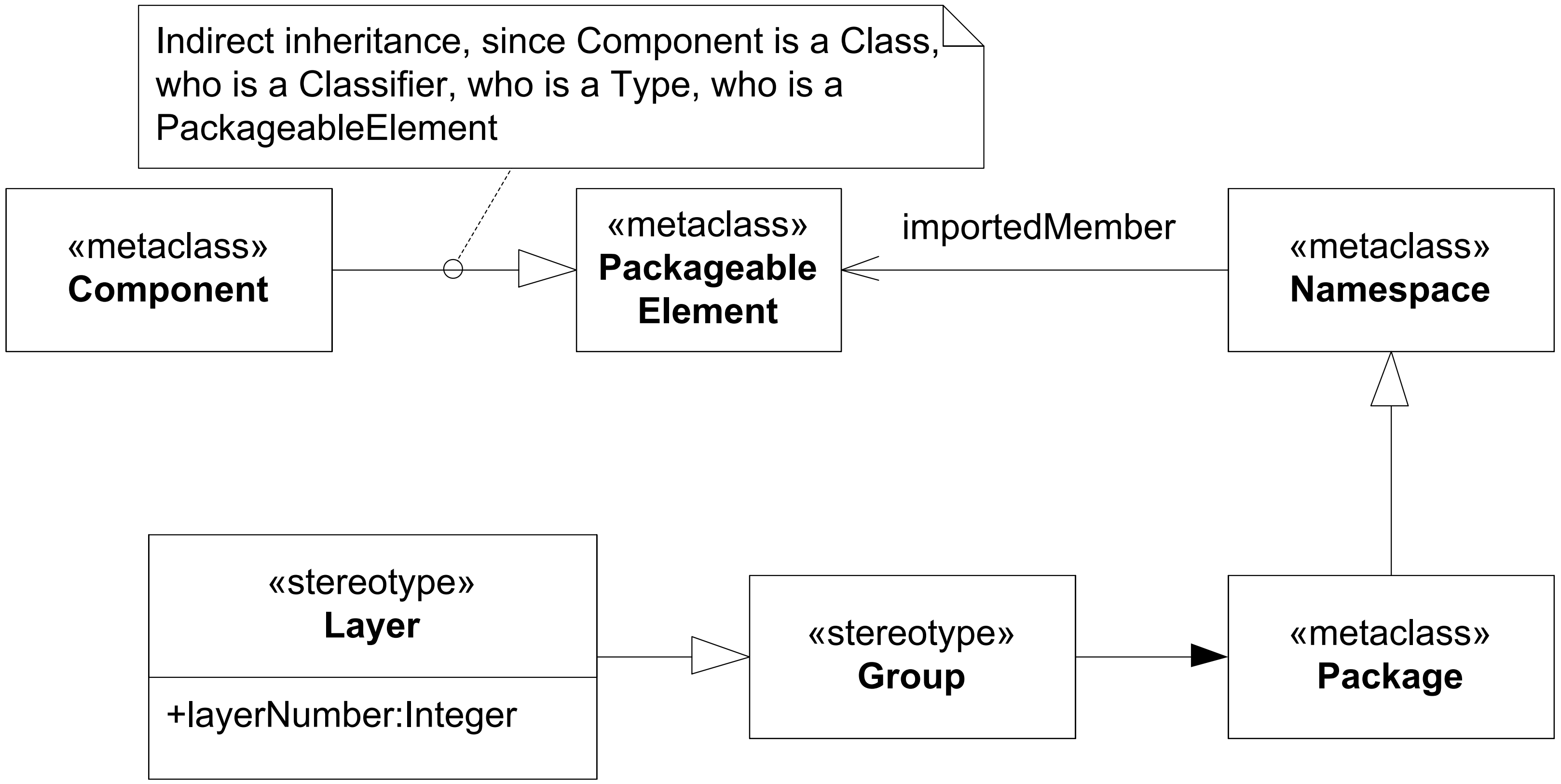
```
(not self.class->isEmpty()) implies self.packagedElement->isEmpty()
```

Simple example of an EJB profile

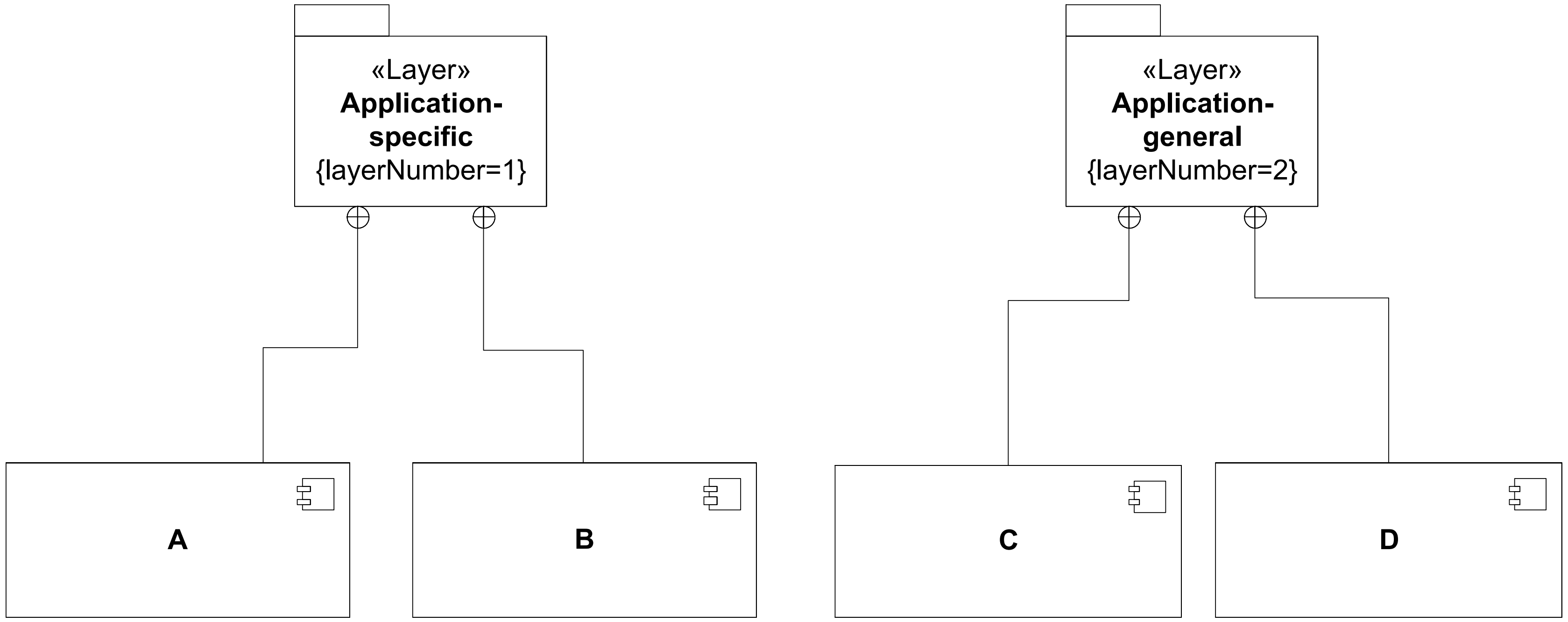


EXTENDING THE UML COMPONENT METAMODEL: MODELING LAYERS

Meta-model Extension


Indirect inheritance, since Component is a Class, who is a Classifier, who is a Type, who is a PackageableElement
«metaclass» Component
«metaclass» Packageable Element
importedMember
«metaclass» Namespace
«stereotype» Layer
+layerNumber:Integer
«stereotype» Group
«metaclass» Package

Grouping Components in Layers Packages



EXTENDING THE UML COMPONENT METAMODEL: CALLBACKS

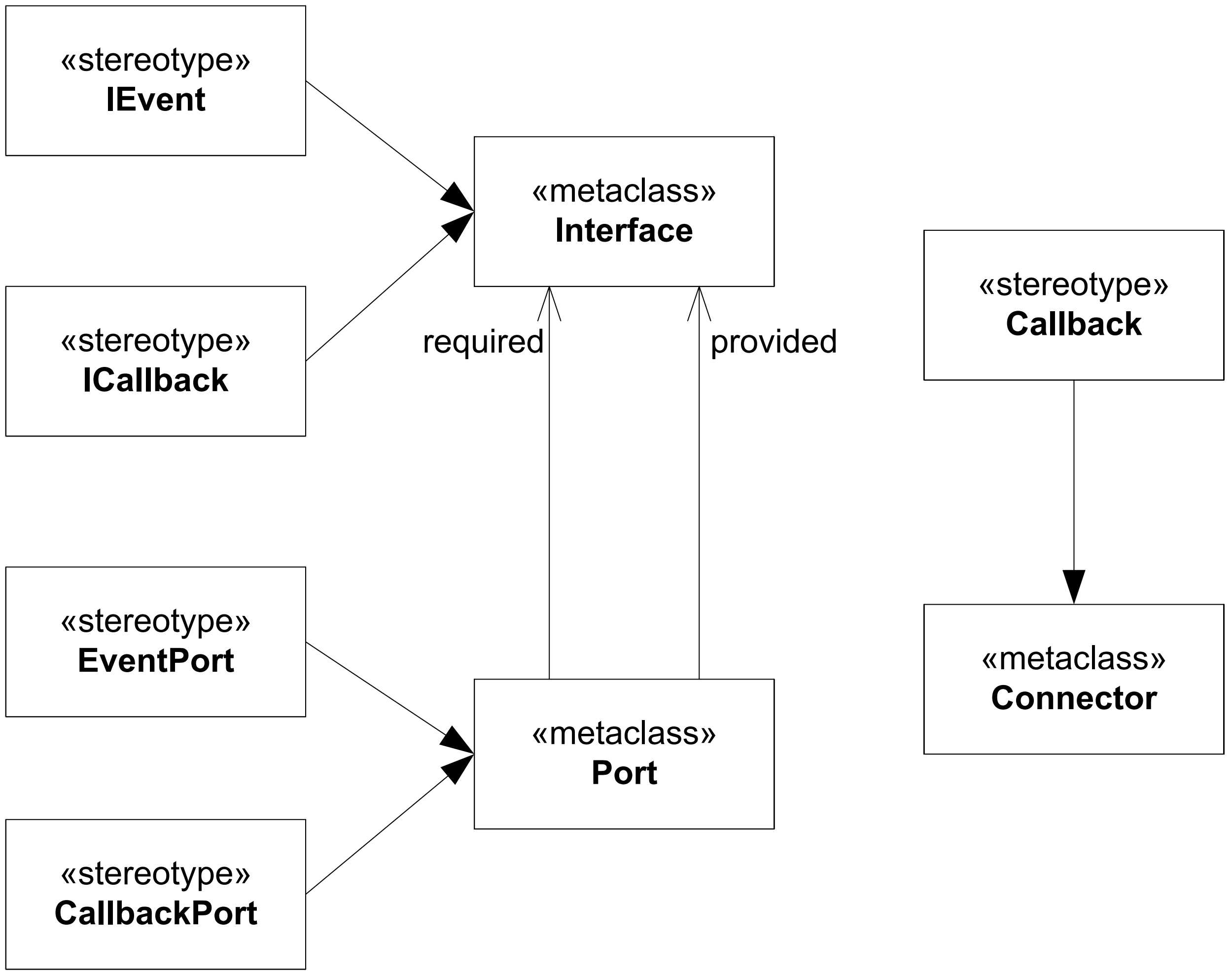
Example: Extending the UML Meta-Model with Support for Callbacks

A callback denotes an invocation to a component B that is stored as an invocation reference in a component A.

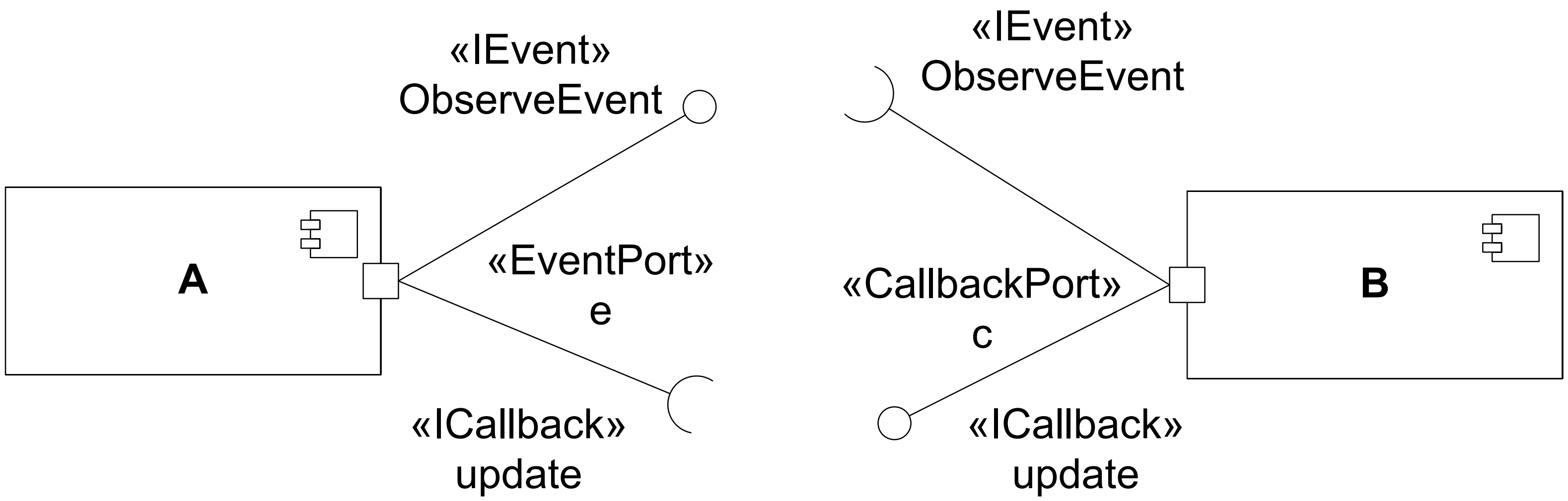
The callback invocation is executed later, upon a specified set of runtime events, usually implemented as methods.

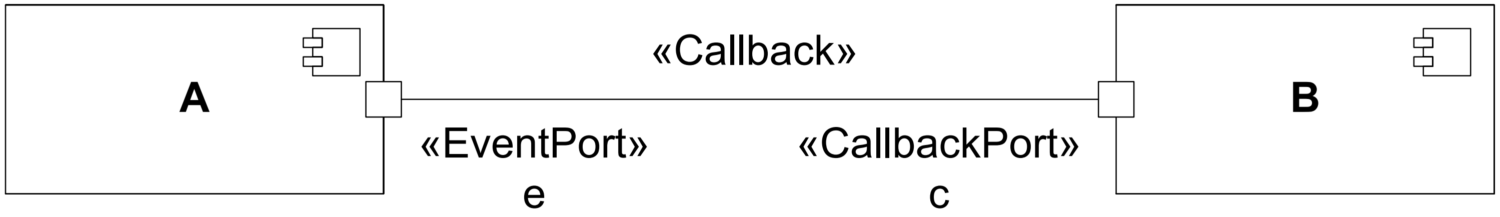
Between two components A and B, a set of callbacks can be defined, also usually implemented

Example: Stereotypes for modeling Callback



Example: Stereotypes for Modeling Callback





Relevant Literature and Sources

- UML 2.3 specifications (especially superstructure) http://www.omg.org/spec/UML/2.3/
- U. Zdun and P. Avgeriou. A Catalog of Architectural Primitives for Modeling Architectural Patterns. *Information and Software Technology*, vol. 50, no. 9-10, pages 1003-1034, Elsevier, 2008.
- Examples from: http://www.uml-diagrams.org/profile-diagrams.html

Many thanks for your attention!

Uwe Zdun

Software Architecture Research Group
Faculty of Computer Science
University of Vienna
http://cs.univie.ac.at/swa