# The UML Profile technology

SOFTEAM

144 Ave des Champs Elysées
75008 Paris, France
www@softeam.com

# Context of this work

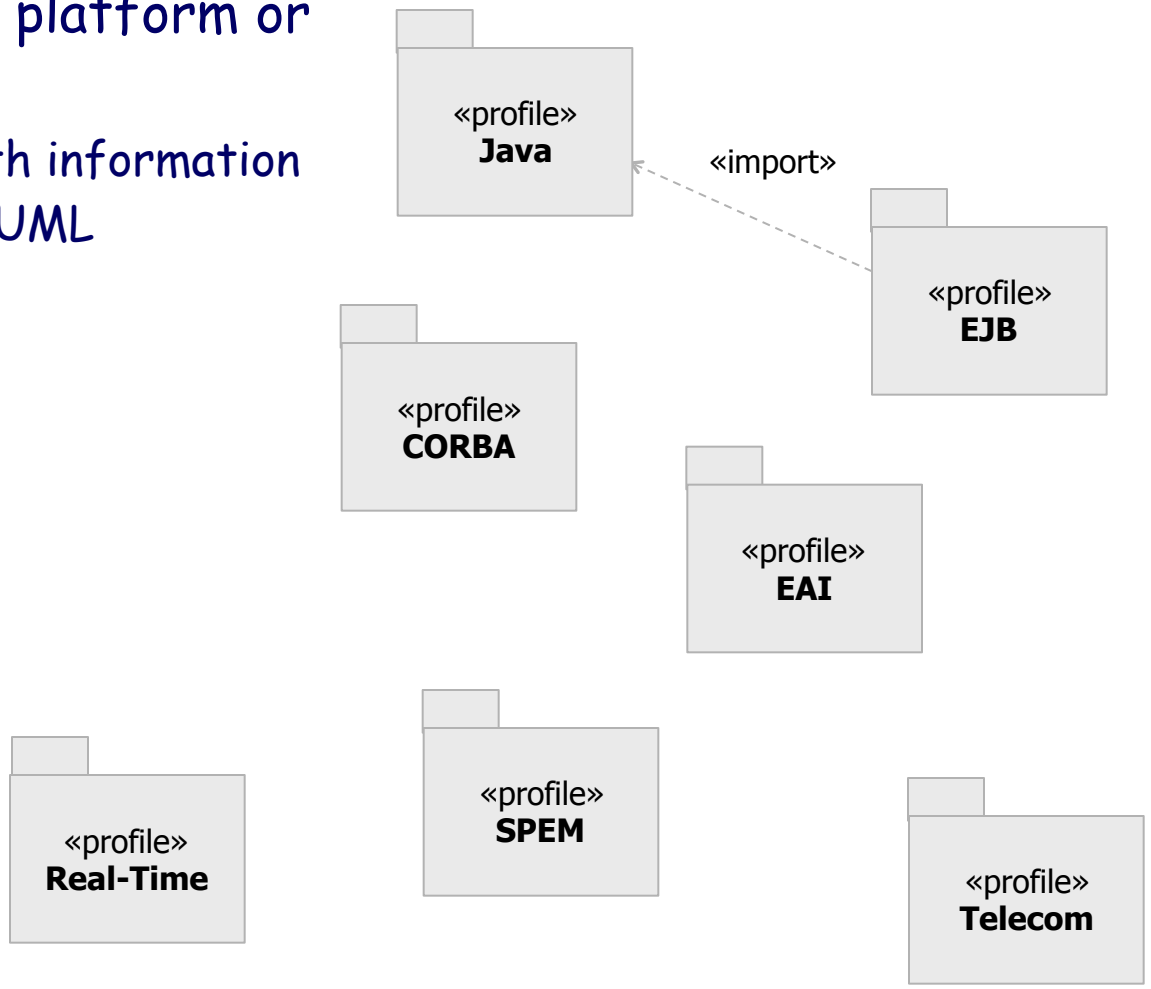

- The present courseware has been elaborated in the context of the MODELWARE European IST FP6 project (http://www.modelware-ist.org/).
- Co-funded by the European Commission, the MODELWARE project involves 19 partners from 8 European countries. MODELWARE aims to improve software productivity by capitalizing on techniques known as Model-Driven Development (MDD).
- To achieve the goal of large-scale adoption of these MDD techniques, MODELWARE promotes the idea of a collaborative development of courseware dedicated to this domain.
- The MDD courseware provided here with the status of open source software is produced under the EPL 1.0 license.

# UML Profiles : Table of content

- Definition
- Benefits & positioning VS MOF
- History
- Notation
- Semantic & Mechanisms

# Profiles (Definition)

- A profile is used to tailor a metamodel (like UML) to a specific platform or domain
  - used to tag a design with information that is not captured in UML



# Profiles (Definition)

- A profile defines a coherent extension to UML in order to adapt it to a specific target or domain.
- Stereotypes allow to define extensions : Kind of metaclasses that extend UML metaclasses.
  - Extension is expressed by a new specific notation
- Tagged-values are defined as specific attributes of stereotypes.

# Profiles extension feature (benefits)

UML Profile is a restricted extension mechanism which :

1. Defines extensions that can be dynamically added to or retracted from an existing model (extension flexibility)

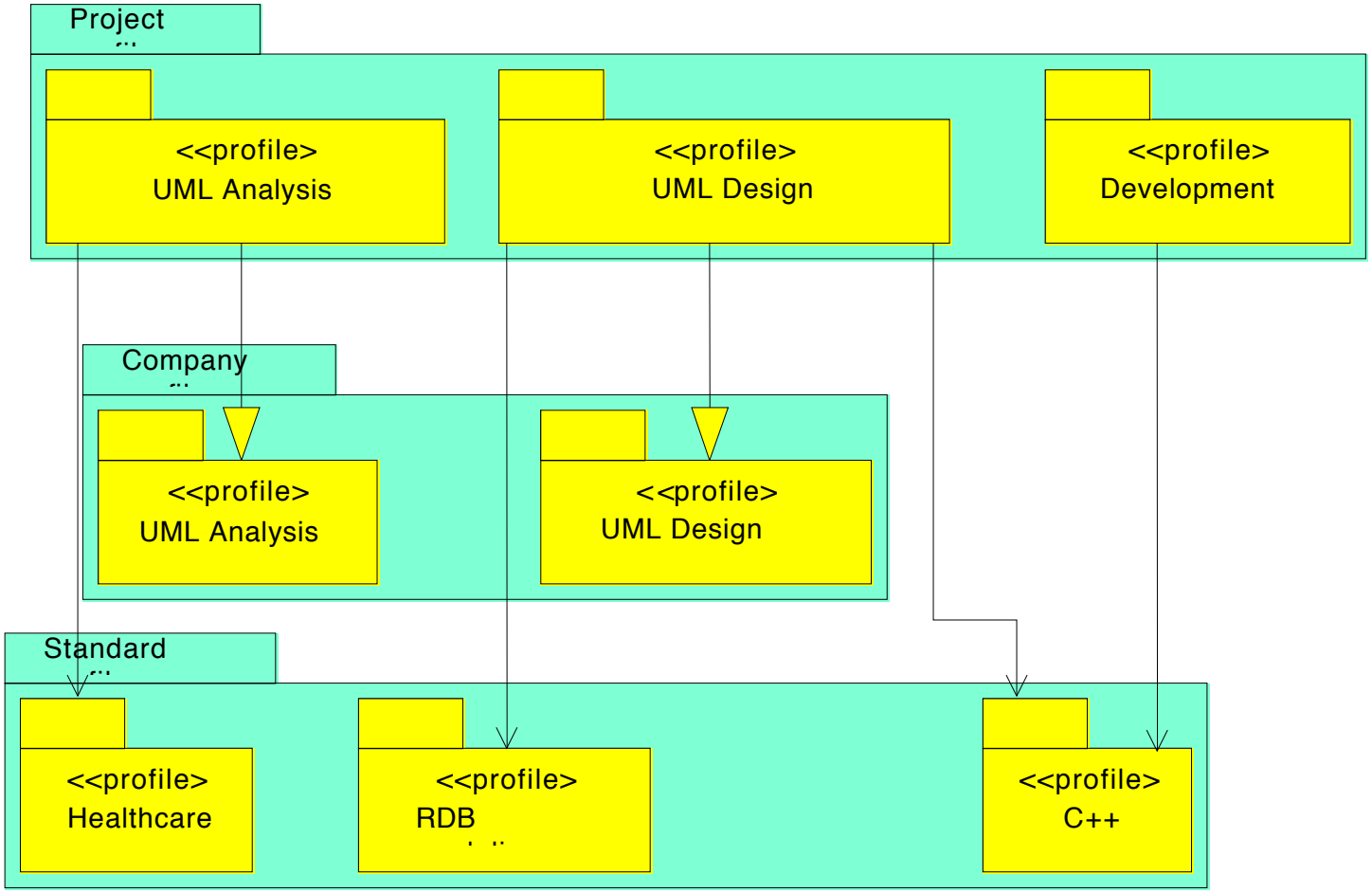
2. Guarantees that the model will remain consistent with the UML standard, even when extended by one or several profiles. (Standard UML conformance)
   - Preserves UML semantics for the extended models
   - Extended models can be exchanged with or without their extensions

# Extending UML : Standard conformance guarantee (benefits)

- MDA requires a high degree of formalization of the different models, architectures, and methodologies
- In addition, mappings have to be specified in detail
- This involves highly skilled people, and implies a great risk to development.
- It necessitates investments in time and money before starting development, and getting ROI

- Using UML profiles guarantees that the adapted models will still be legal UML models
- Many Standard Profile solutions exist for some PSM's or PIMS (examples : CORBA, EJB, Test, SysML, Qos, CCM)
- Some very important targets have not yet standard profiles, but on the shelf solutions (RDB, C++, etc.)
- Build on a common basis: low learning curve, existing tools, interoperability, etc.

# Example of a profile based (benefits) MDA configuration



# UML for a specific purpose (benefits) = a specific profile

- Profiles exist for :
  - technical targets (C++, RDB, Java, CORBA, XML, Real Time, etc.)
  - specific disciplines (Analysis, Design, etc.)
  - requirement analysis
  - tests (test for Java, test for EJB, test for C++, etc.)
  - metrics, quality checking, and profiles for managing configuration management rules.
- From early requirement down to final tests, UML profiles can drive each specific discipline involved.

# MOF/Profile alignment (benefits)

- Core/MOF Package → Profile
- Core/MOF Class → Stereotype (restricted usage)
- Core/MOF Property → Property (was tagged definition)
- Core/MOF Operation → Operation
- Core/MOF Constraint → Constraint
- Core/MOF association → Association (restricted usage)
- Core/MOF association → Extension (specific "pattern")

# Rationale for Using Profiles Vs MOF (benefits)

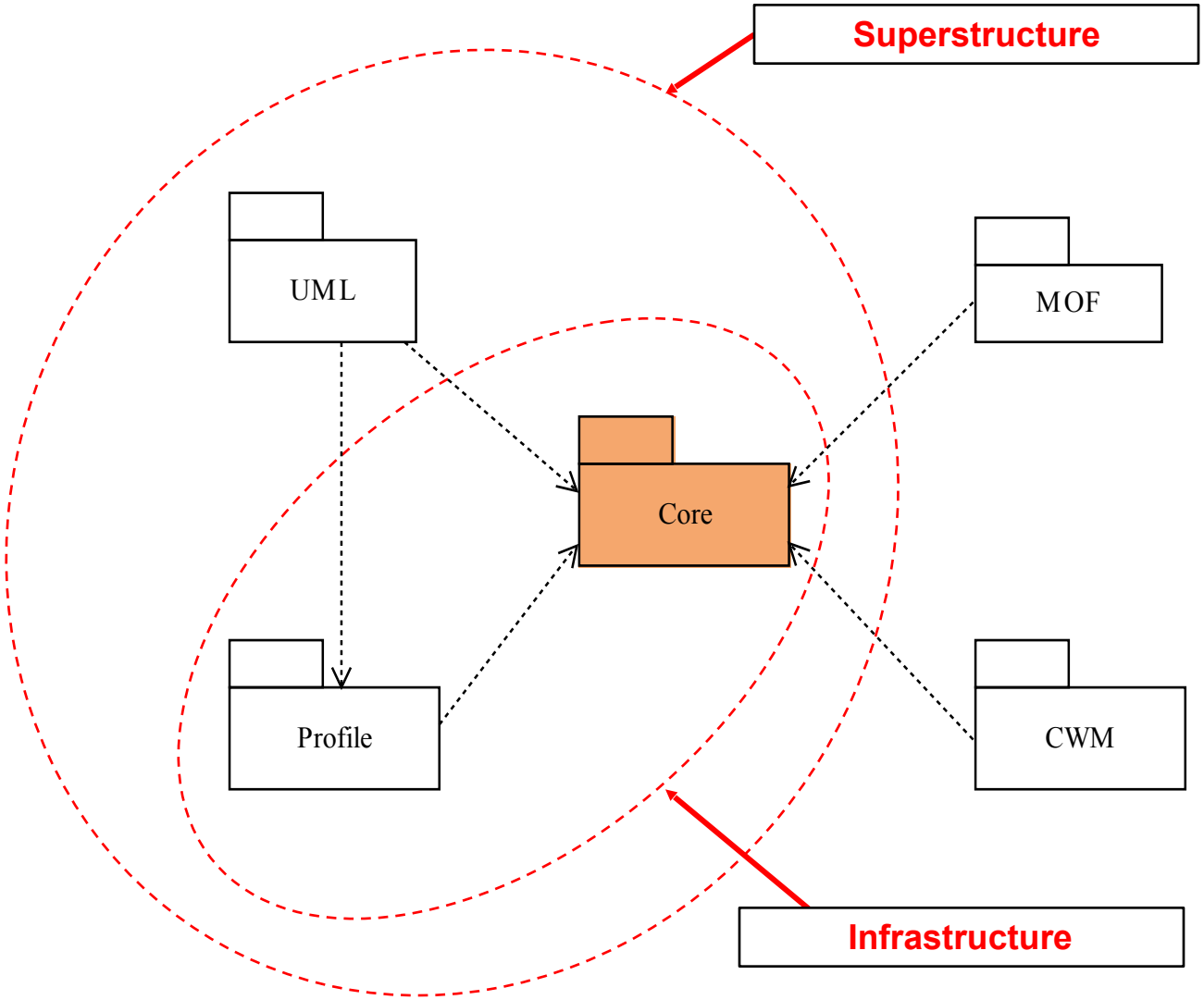
- Profiles
  - Are used for extending the UML language (the « reference metamodel »)
  - Are supported by UML Case tools
  - Guarantee the UML conformance of the extensions
  - Provide a dynamic extension capacity (i.e. extending an existing model)
  - Typical example: UML for a certain purpose

- MOF extensions
  - Are used to create new metamodels
  - Apply to any metamodel
  - New models are created from MOF extensions (no existing model updates)
  - Are supported by MetaCase tools or infrastructure.
  - Typical example: New metamodel (e.g. DSL, workflow, UML, etc.)

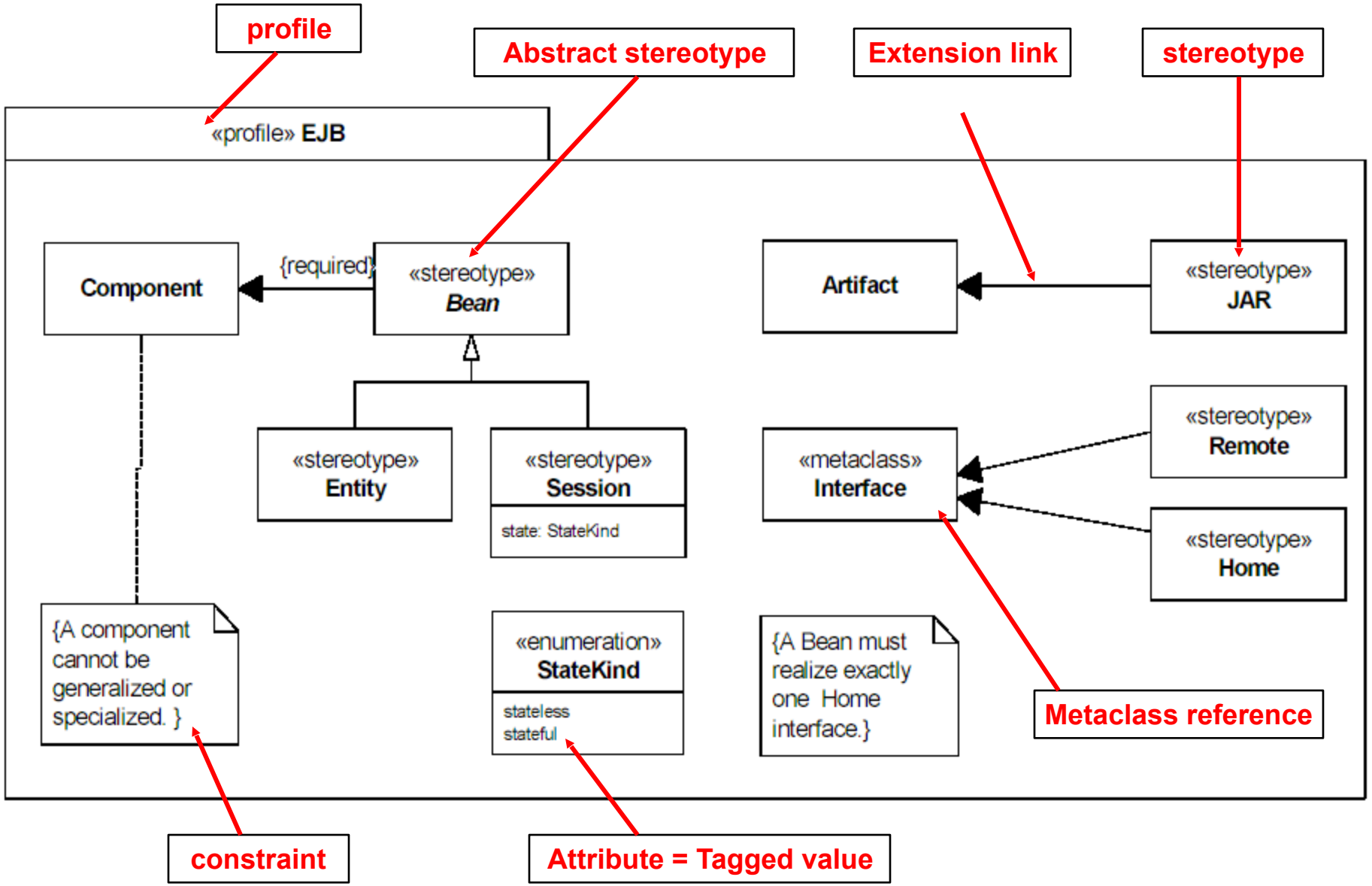
# Profile: History

- UML 1.0: (1999) Stereotypes and tagged values were defined as specific strings attached to ModelElements. Stereotypes were defined at the level of the model itself.
- UML1.4: (2002) The notion of UML profile was more formally defined. Profiles organize Stereotypes, which may contain tagged values. Tagged values can be directly extend model elements, without stereotypes
- UML 2.0: (2005) Stereotypes are now defined at M2 (metaclass level). The Profile notion is completely formalized, in correspondence and conjunction with the metamodel (MOF). A notation is defined, and the "extension" capacity is formalized.

# Profile's position within the UML2/MOF2 architecture

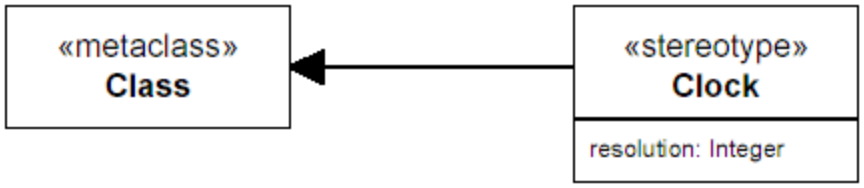


# Profile : Notations 1

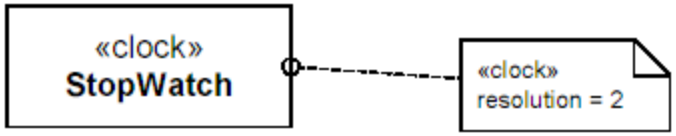


# Profile : Notations 2

- Defining the « Clock » stereotype

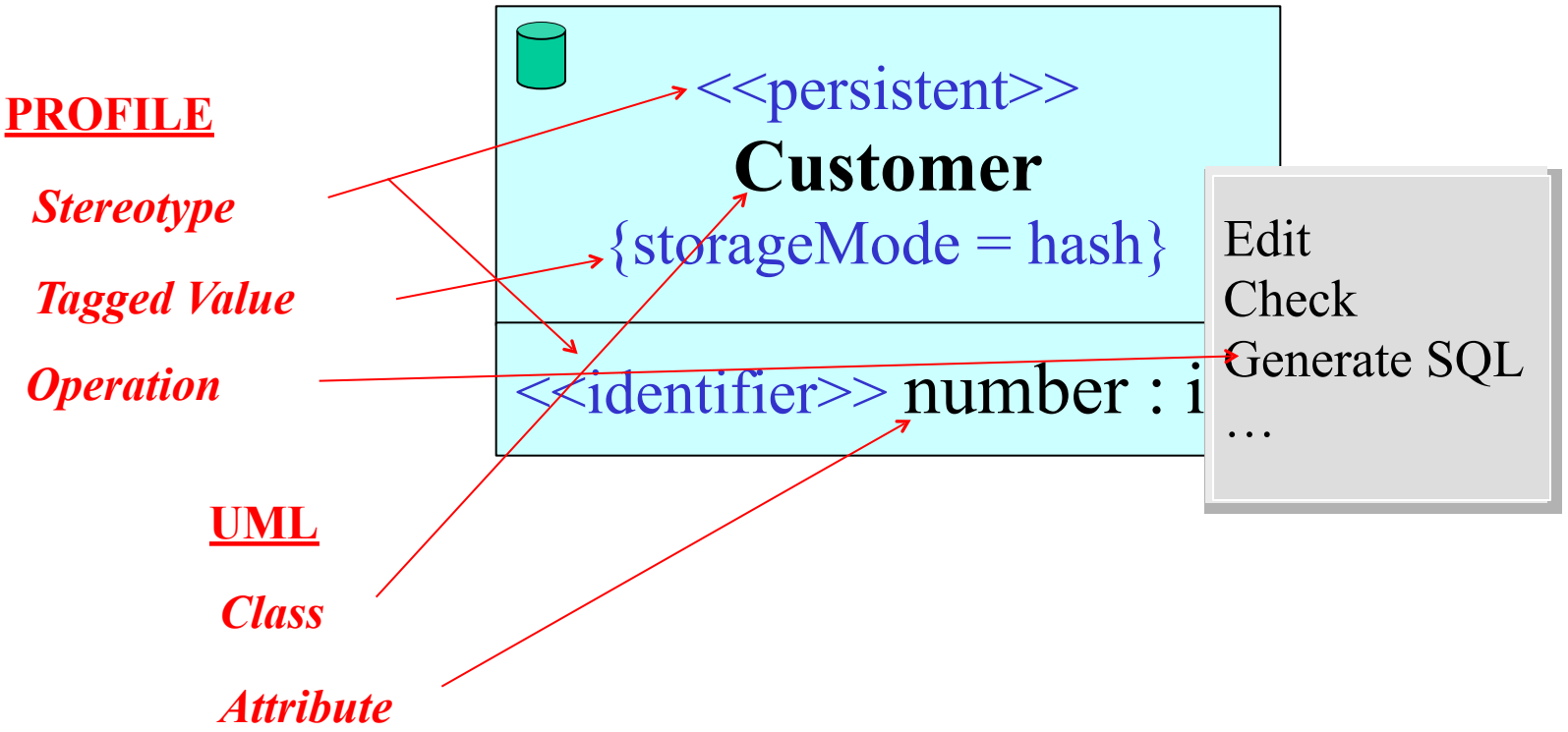



- Applying the Clock stereotype on the StopWatch class

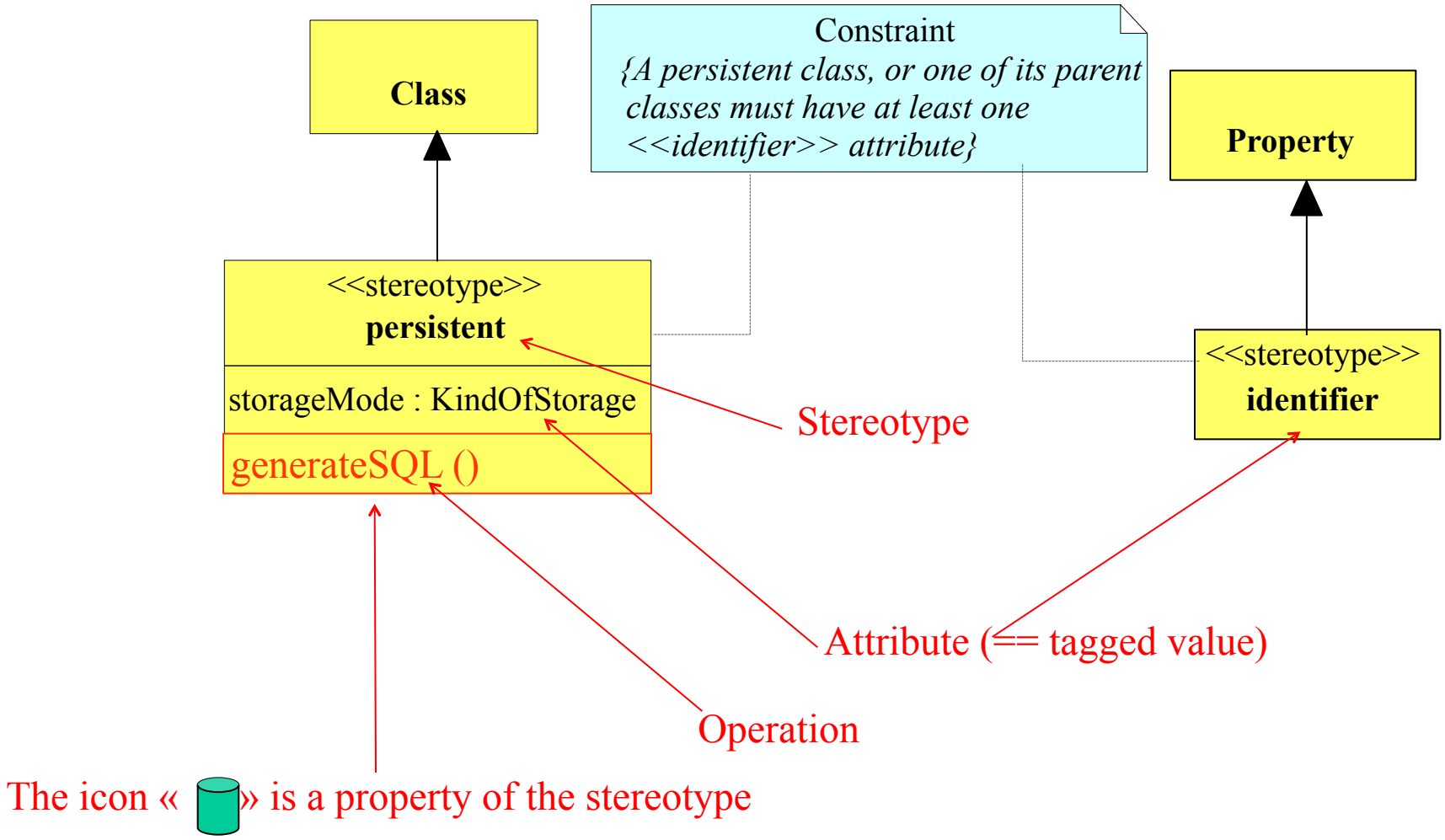


# Profile : Example (Usage of UML extensions)

Below is a user model example, with extensions. That example supports extensions.
We will see how they are defined at the profile definition level.

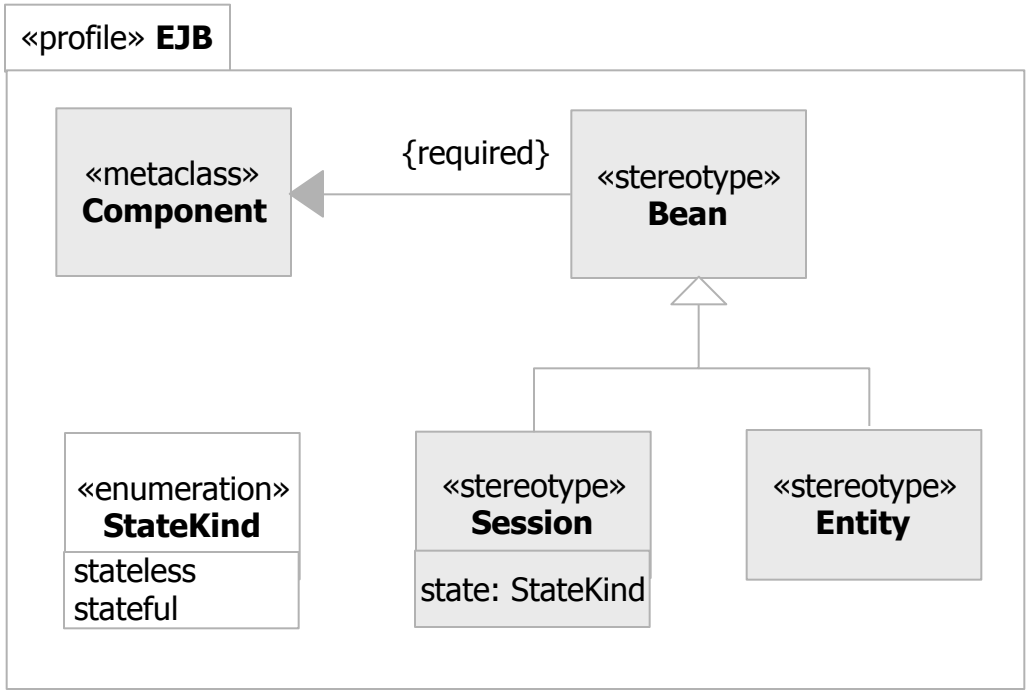


# Profile : Example (Definition of the UML extension)

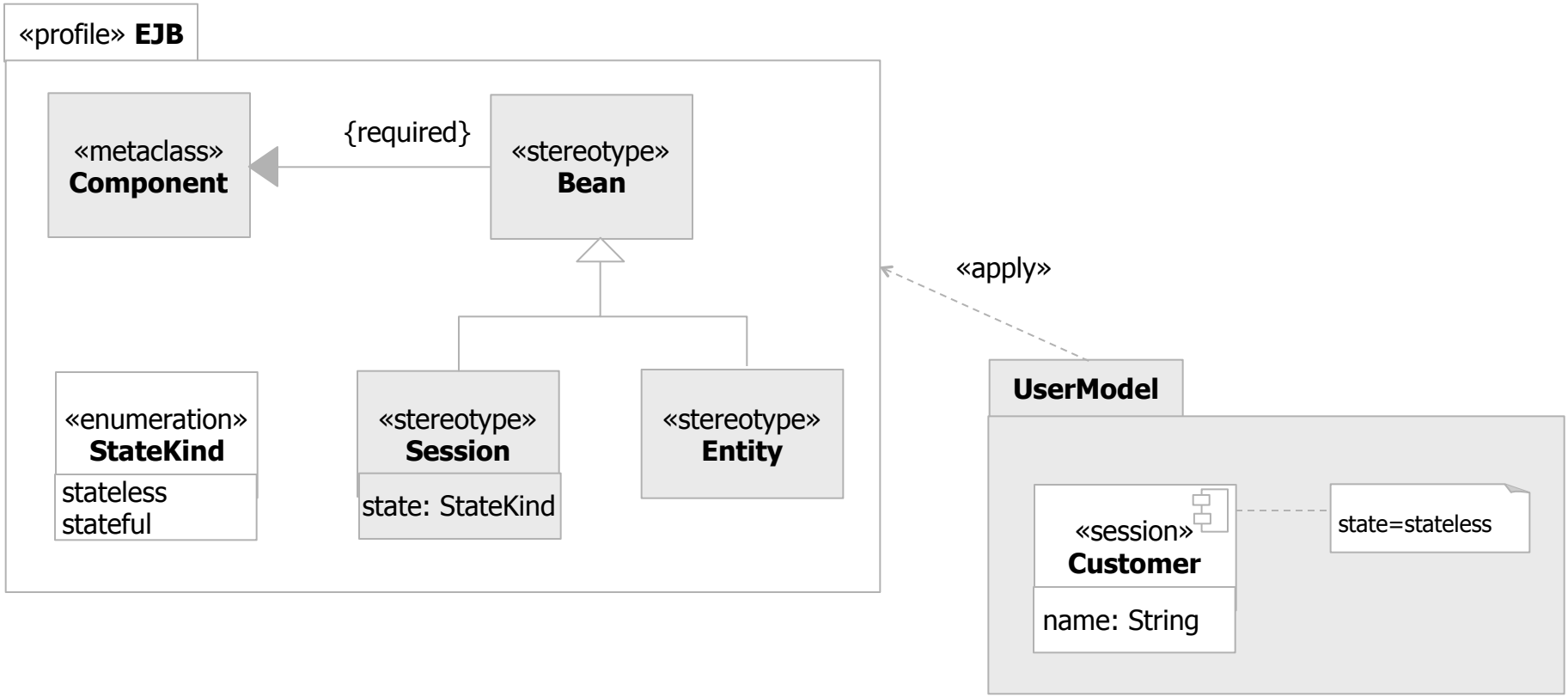


# Extending Metaclasses

- A profile consists primarily of stereotypes
  - restricted metaclasses that can only be used to extend existing metaclasses

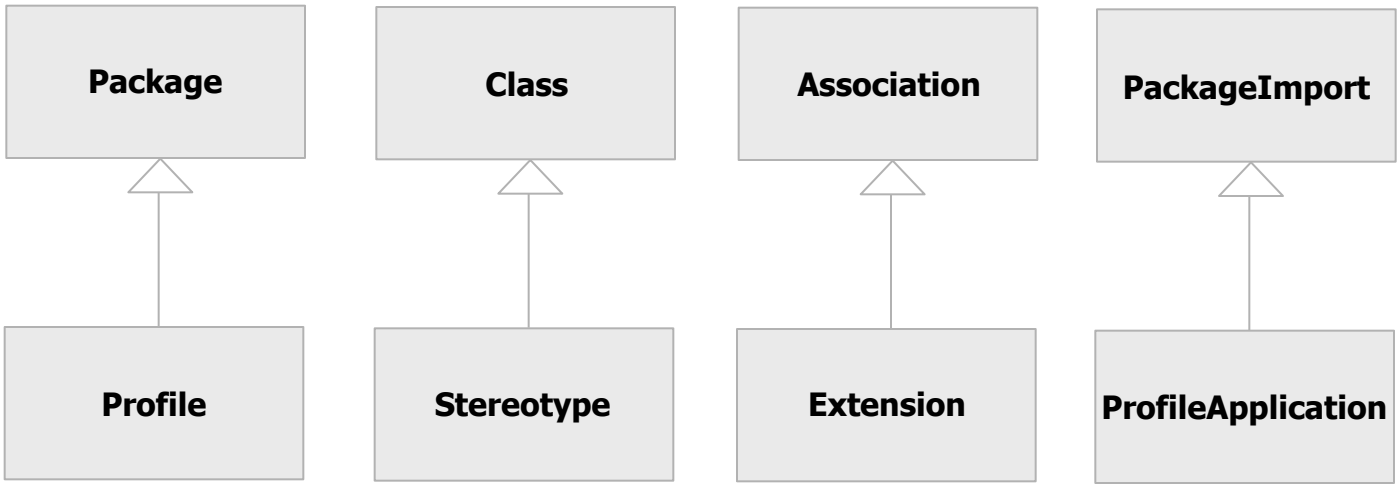


# Applying a Profile




Applying a profile makes the profile extension available to the model packages to which it is applied.

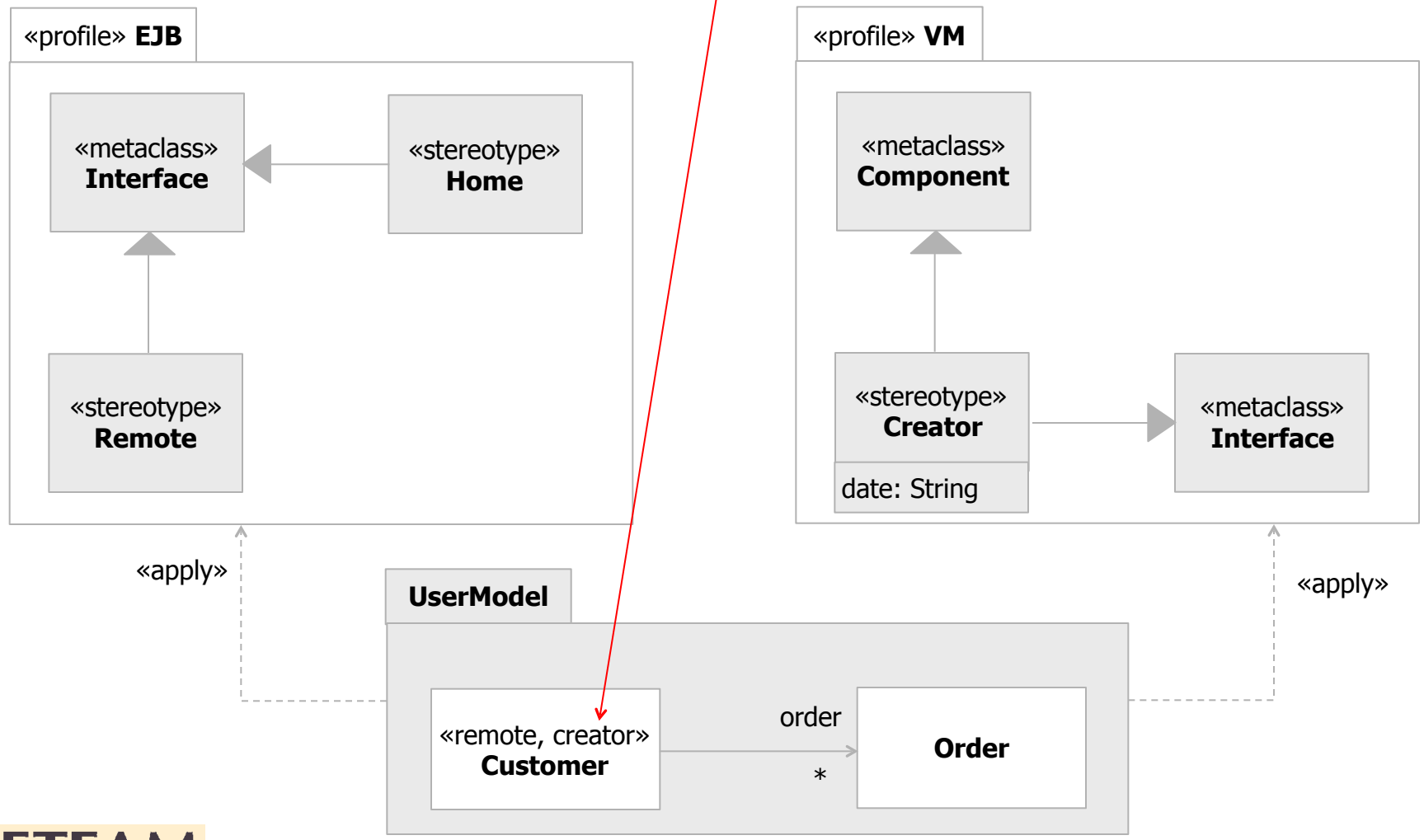
# Metamodel Excerpt



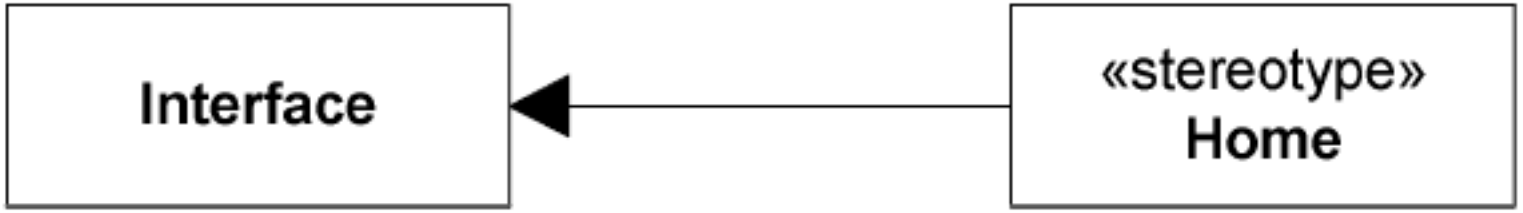
# Multiple Stereotypes

Several stereotypes can be applied to a model element.

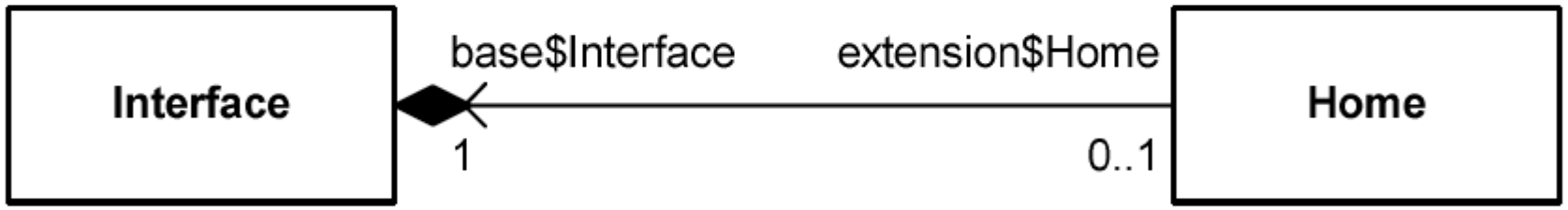
A specific notation supports this feature



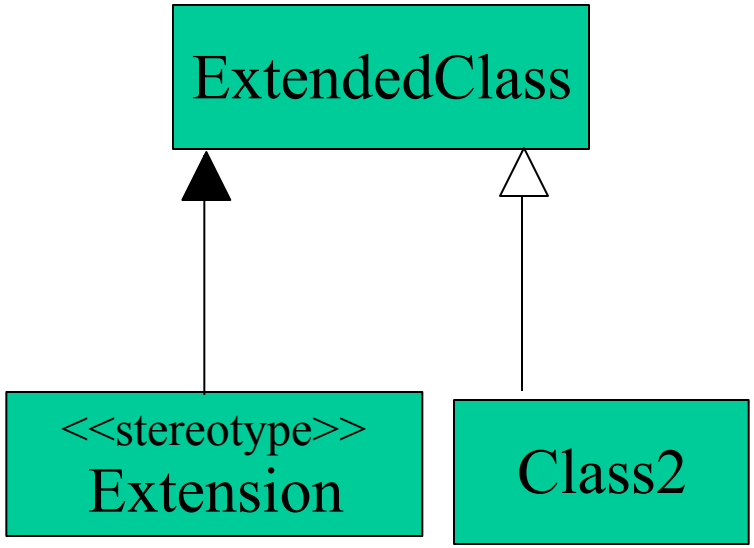
# Stereotype extension semantics

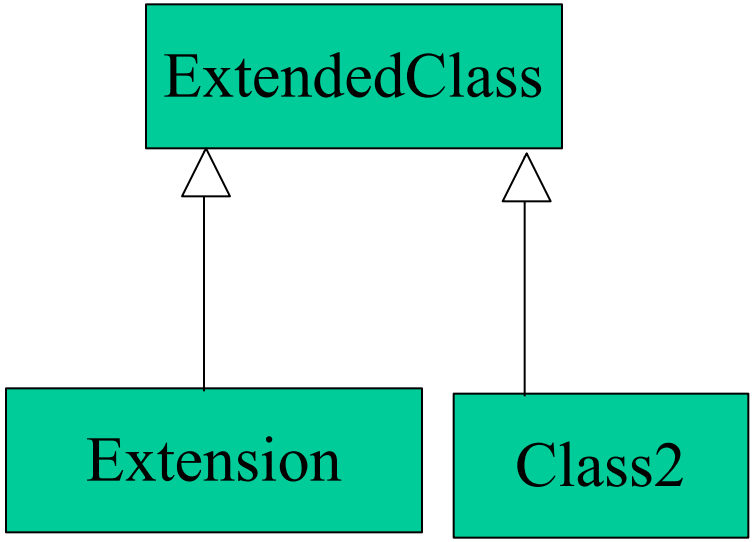



Extension is not to be confused with Generalization. It is indeed a delegation pattern. Below, a MOF equivalent construct is shown. The dynamic extensibility features of profiles are due to the capacity to add links to stereotype instances on existing model elements



# Extension semantics (VS Generalization)



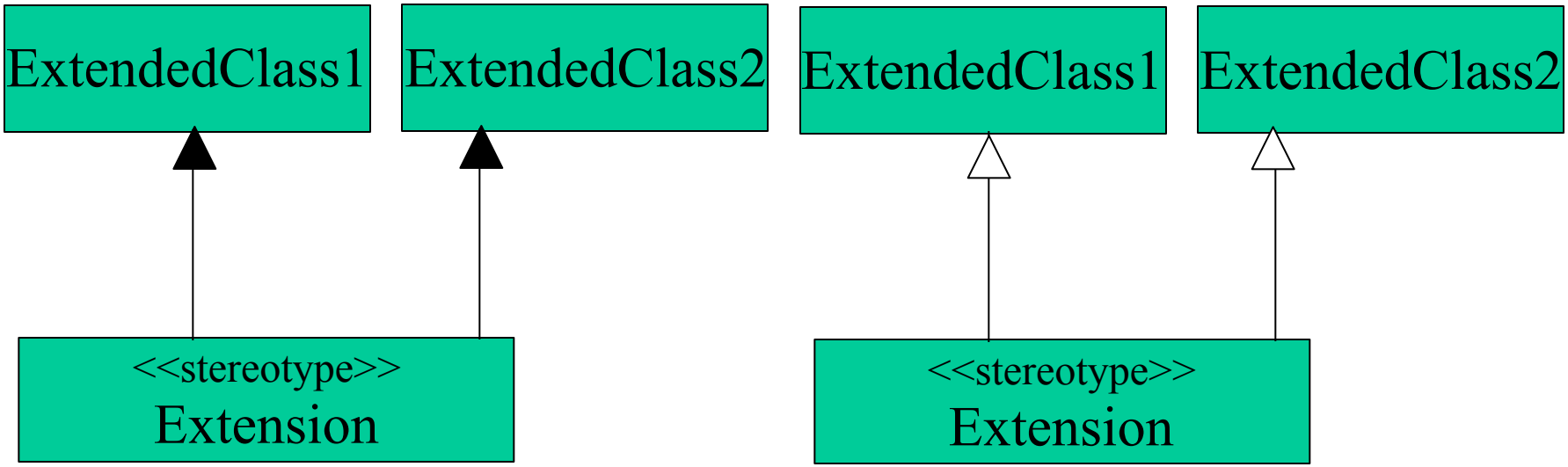

Semantics :

- Extension can extend existing « ExtendedClass » or subclass instances.
- Class2 instances can for example also be extended by « Extension »

« Usual » Semantics :

- There will exist once for all separated « Extension » instances, « Class2 » instances and also « ExtendedClass only» instances

# Extension semantics (1)



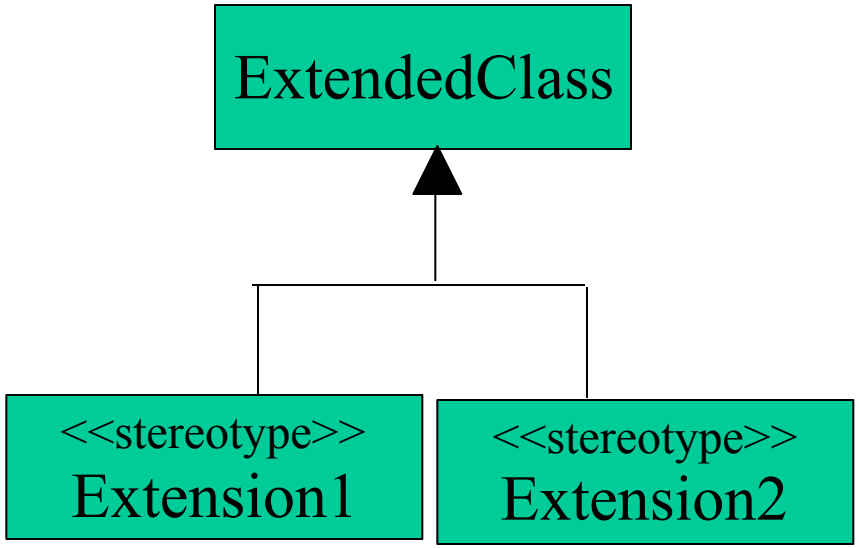

Semantics :

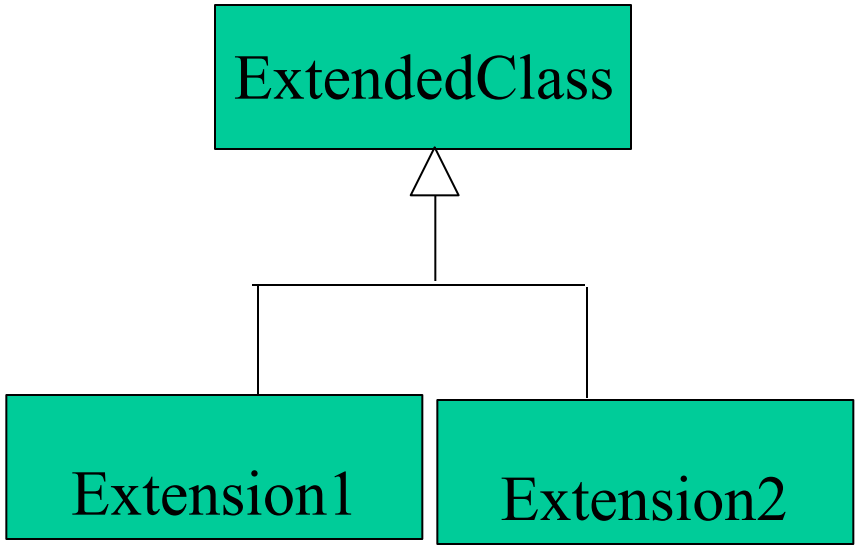
- Extension can extend either (exclusively) ExtendedClass1 or ExtendedClass2. Extension does not define a « merge instance » of ExtendedClass1 and ExtendedClass2

« Usual » Semantics :

- « Extension » instance will always be also instances of ExtendedClass1 and ExtendedClass2

# Extension semantics (2)




Semantics :

- An instance of « ExtendedClass » will be able to be simultaneously extended by « Extension1 » and « Extension2 »

« Usual » Semantics :

- There may exist separate instance of « Extension1 » extending « ExtendedClass » and « Extension2 » extending « ExtendedClass »

# Miscellaneous

- An extension can be "required". In this case, when a profile is applied to a model, each element which metaclass is extended by a required stereotype will be extended by this stereotype.
- A stereotype cannot inherit a metaclass, nor the opposite
- The extended metamodel is called the reference metamodel. It is a "read only" metamodel.
- Stereotype properties can have complex types (such as classes/metaclasses)
- Associations can exist, navigable from a stereotype to a metaclass or stereotype.