Modèle: Définition

A **Model** represents reality for the given purpose; the model is an **abstraction** of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality

Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose

Par Jeff Rothenberg, « The nature of modeling »

- Attention au débat: abstraction = simplification?
  - La modélisation simplifie la compréhension et la communication autour du problème, elle ne simplifie pas le problème lui même!
- Attention, un modèle n'est pas forcément graphique

Modèle: Définition (pour les matheux!)

Un modèle est un graphe

- **Definition 1**. *A directed multigraph* $G = (N_G, E_G, \Gamma_G)$ consists of a set of distinct nodes N_G, a set of edges E_G and a mapping function $\Gamma_G : E_G \rightarrow N_G \times N_G$

- **Definition 2**. *A model* $M = (G, \omega, \mu)$ is a triple where:
  - ✓ $G = (N_G, E_G, \Gamma_G)$ is a directed multigraph
  - ✓ ω is itself a model, called the reference model of M, associated to a graph $G_\omega = (N_\omega, E_\omega, \Gamma_\omega)$
  - ✓ μ: $N_G \cup E_G \rightarrow N_\omega$ is a function associating elements (nodes and edges) of G to nodes of G_ω (metaElements)

Modèle: Exemple

- La pipe selon Magritte

Modèle: Exemple

représente

Système

Modèle

Méta-Modèle: Définition

- Pour être productifs, les modèles doivent être utilisés par des machines

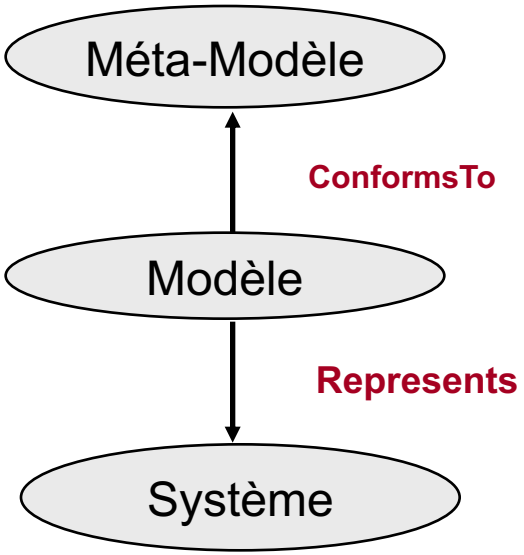
  => Besoin d'un langage précis pour les définir = **Méta-Modèle**

- A "*metamodel is a model that defines the language for expressing a model*" [OMG].

- "metamodel makes statements about what can be expressed in the valid models of a certain modeling language" [Seidewitz].

Méta-Modèle: Définition

- The relation between a model and its metamodel is a "*Conforms To*" relation and a model conforms to its metamodel if the elements and relationships between these elements are defined in the metamodel.



Méta-Modèle: Exemple

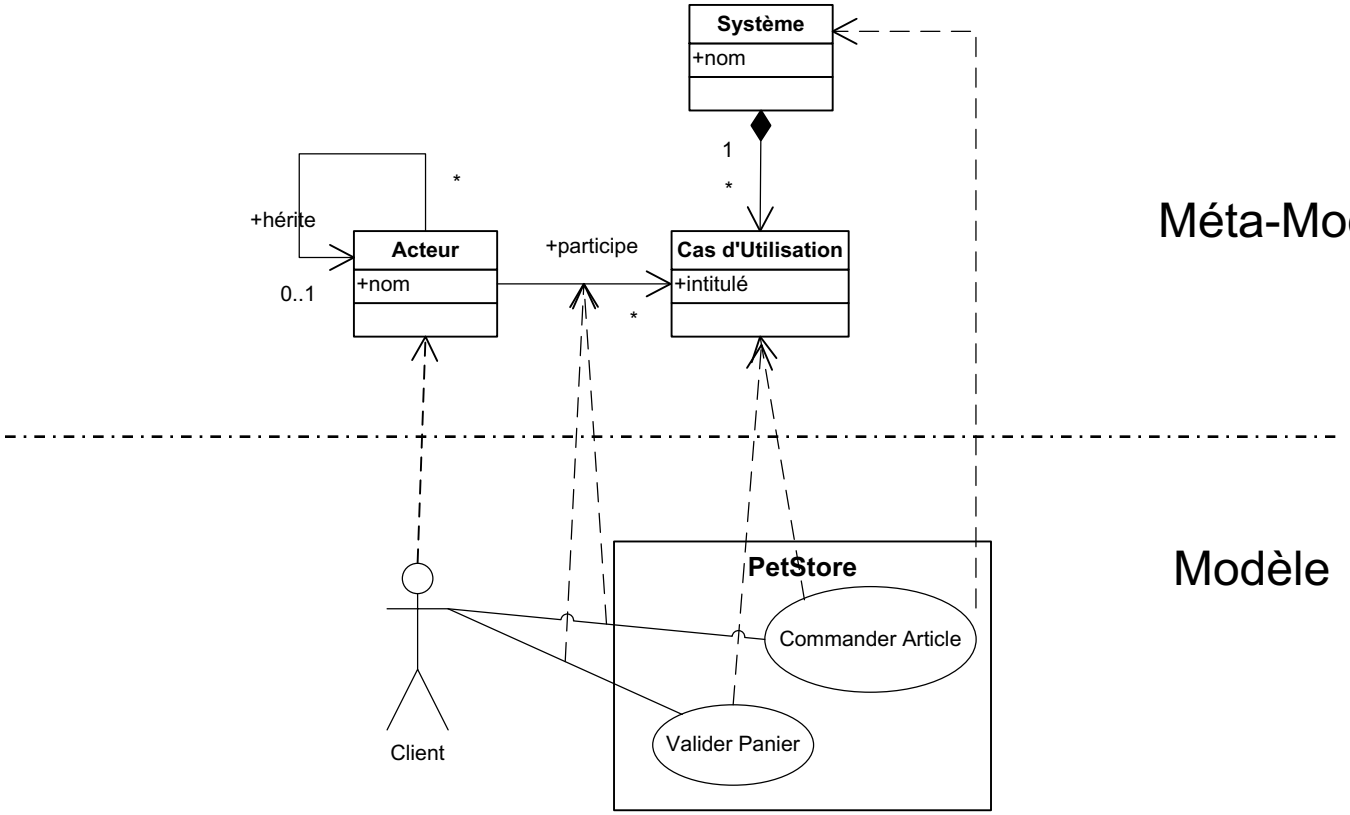


Fig. from X.Blanc

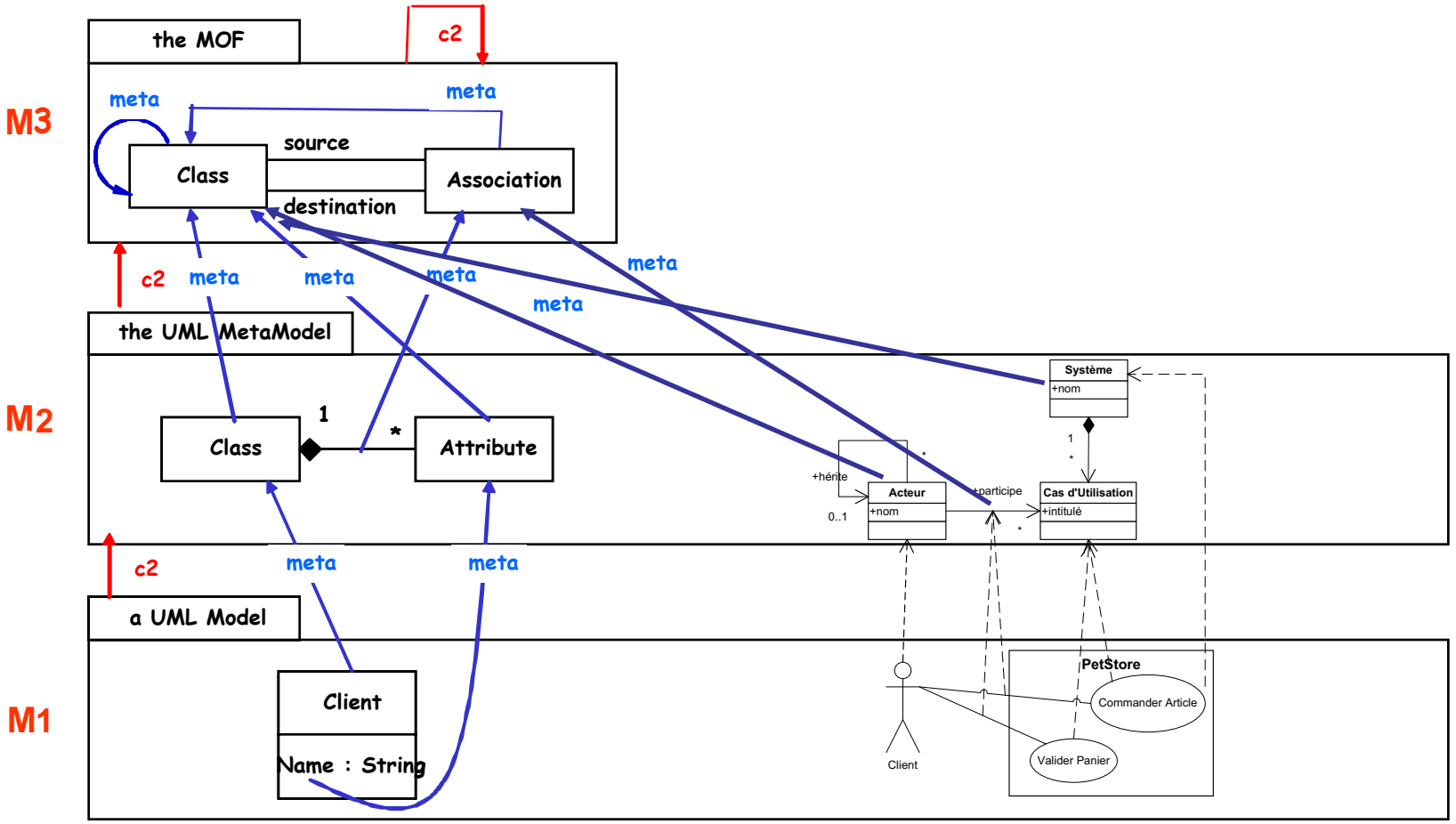
Méta Méta-Modèle: Définition

- A "*metametamodel is a model that defines the language for building metamodels*" [OMG].

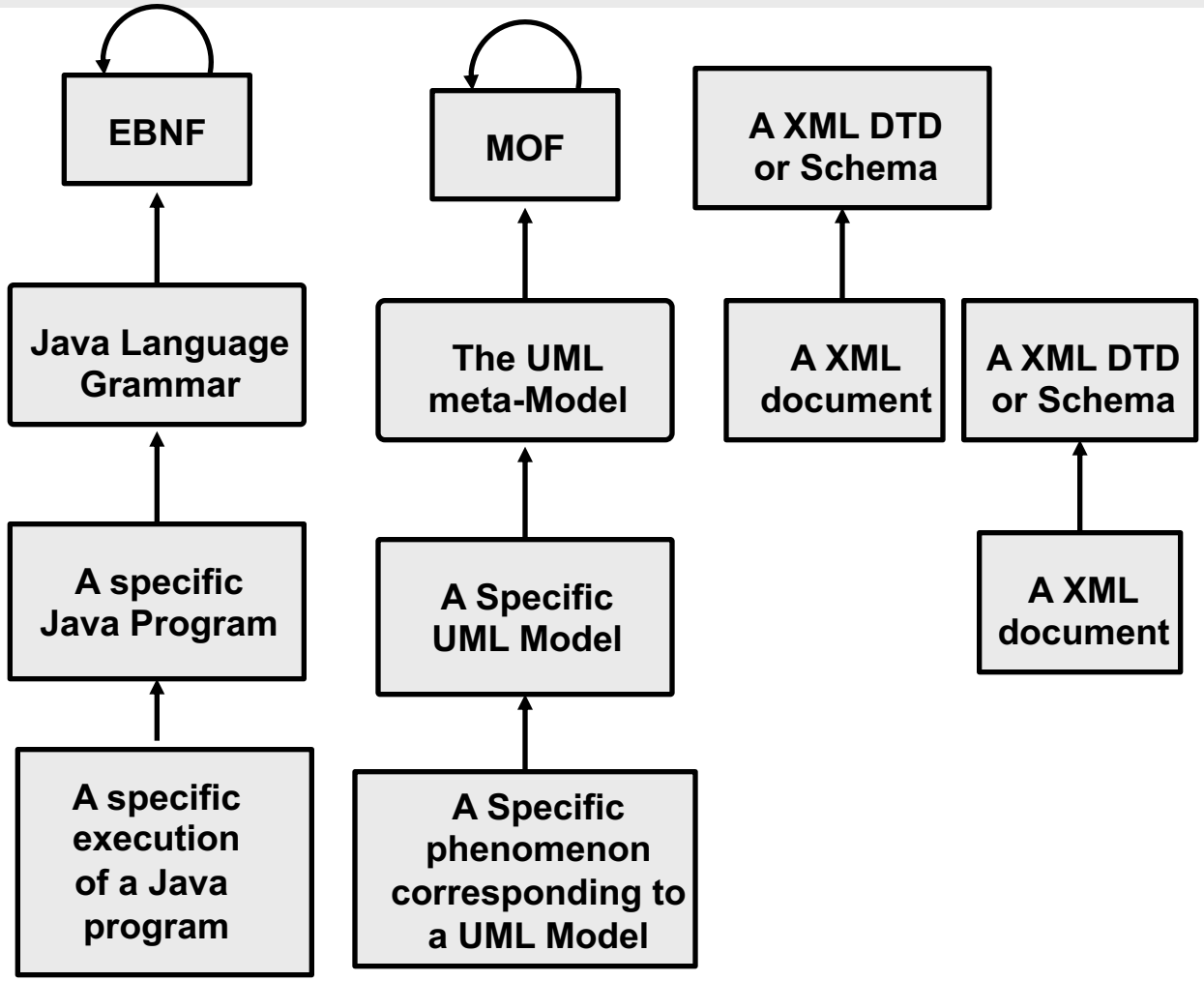
- À l'OMG, MOF représente le langage pour la spécification de nouveaux méta-modèles (langages). Exp. UML, SPEM, OCL...

- Le MOF est conforme à lui-même, auto descriptif.

Méta Méta-Modèle: Exemple

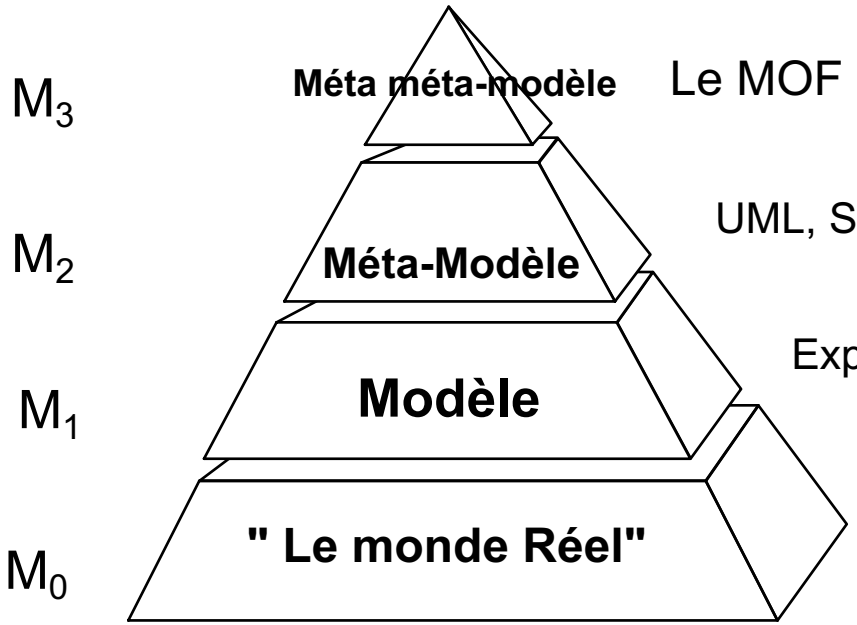


Méta Méta-Modèle: autres systèmes



Model-Driven-Engineering: Une Architecture

- Une architecture à 4 niveaux, un vocabulaire i.e. M0, M1, etc.



Syntaxe Abstraite avec le Framework EMF

- Présentation
- Le méta-modèle
- Principes et architecture
- Génération de code et manipulation de modèles

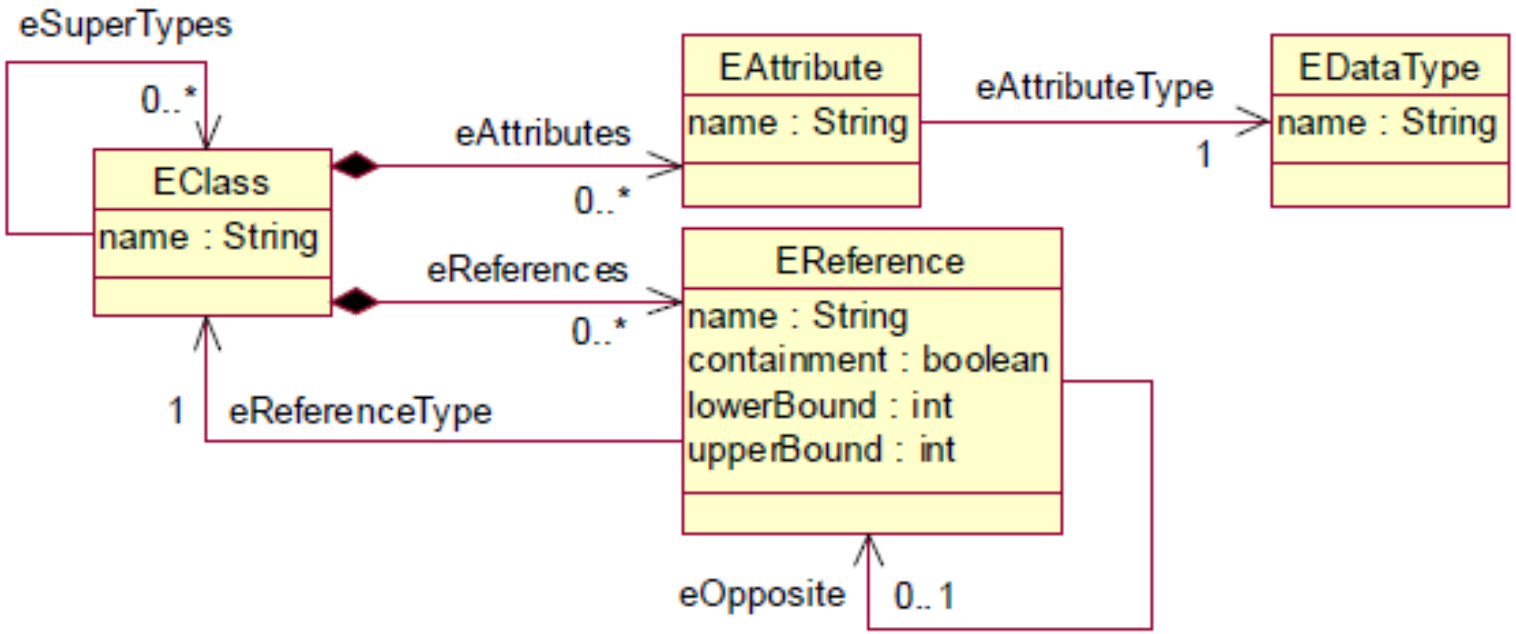
EMF: Présentation

- Eclipse Modeling Framework permet la création d'un langage et de sa suite outillée dans l'environnement Eclipse (depuis 2002)
  - Un Framework riche
  - Un éditeur pour créer des méta-modèles
  - Une génération de bibliothèque Java support à la manipulation de modèles
  - Un éditeur pour construire des modèles
  - Des facilités dédiées (transformation, éditeur graphique, vérificateur de contraintes, ...)

EMF: Le méta méta-modèle

- Utilise/implémente l'essentiel du MOF (i.e. eMOF)
- Le méta méta-modèle s'appelle **Ecore (décrit avec lui-même Ecore.ecore)**
- Notation: le diagramme de classes UML

Un bout du méta méta-modèle



EMF: Les Types

Ecore Data Type	Java Primitive Type or Class
EBoolean	boolean
EChar	char
EFloat	float
EString	java.lang.String
EByteArray	byte[ ]
EBooleanObject	java.lang.Boolean
EFloatObject	java.lang.Float
EJavaObject	java.lang.Object

EMF: Modèle? Persistance?

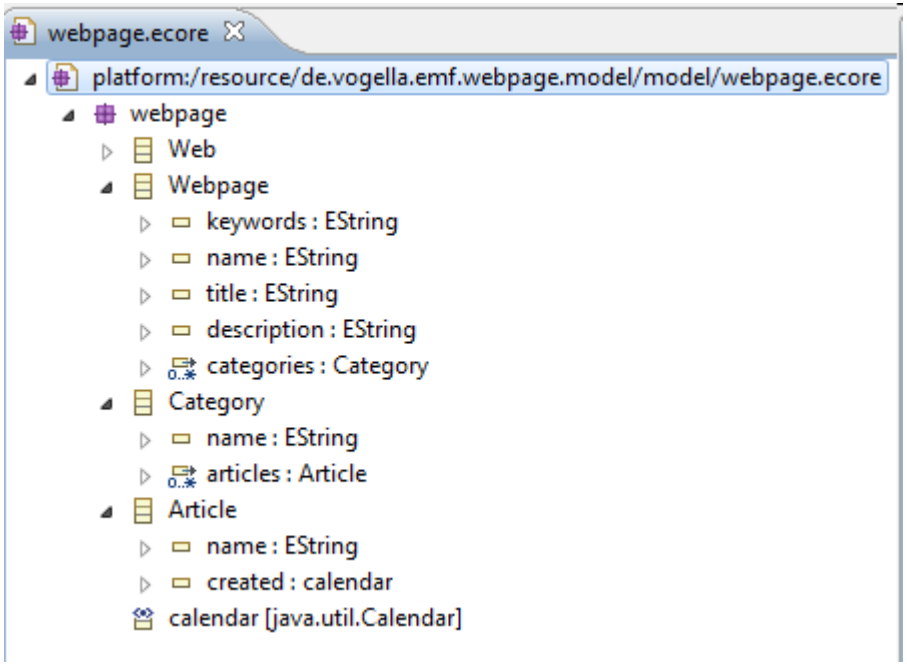
- Un modèle EMF (i.e. Ecore) représente la spécification des données d'une application
  - Attributs
  - Opérations
  - Associations
  - Contraintes, etc.

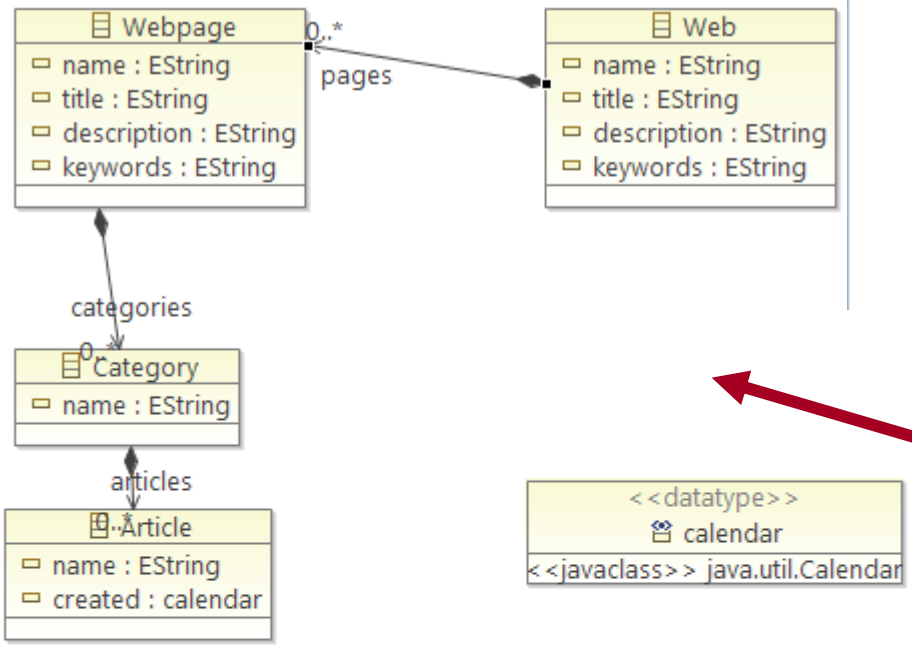
- **Dans le cas où votre application c'est la définition d'un DSML=> le modèle Ecore est le méta-modèle de votre DSML**

- **La persistance des modèles Ecore se fait en XMI**
  - Possibilité de les sérialiser en un format compatible avec eMOF
  - De créer son propre format XMI
  - D'autres mécanismes de persistance sont également possible

EMF: Comment créer des méta-modèles?

- En utilisant l'éditeur arborescent d'EMF

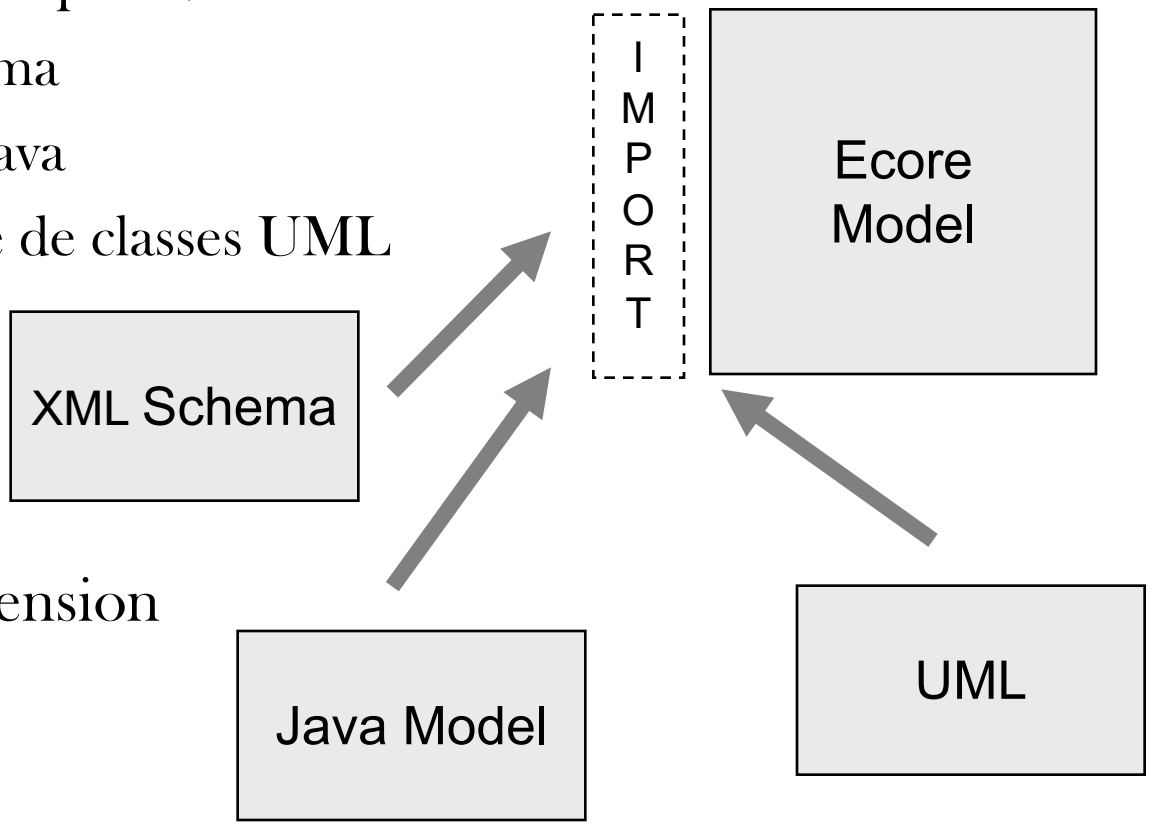




- En utilisant l'éditeur Graphique d'EMF

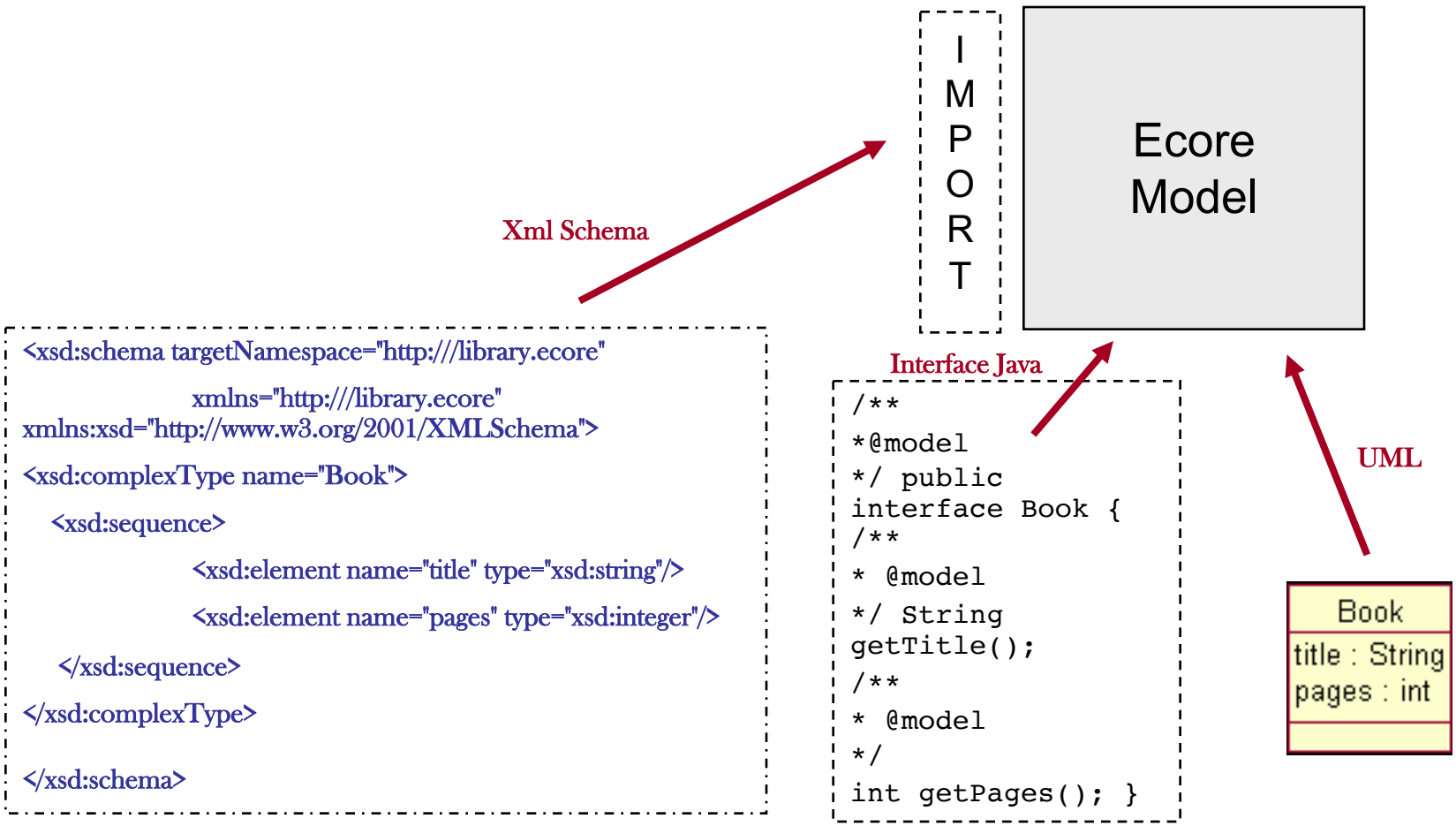
EMF: Comment créer des méta-modèles?

- Un autre moyen est de les créer en dehors d'EMF et de les importer. Trois possibilités:
  - XML Schema
  - Interfaces Java
  - Diagramme de classes UML



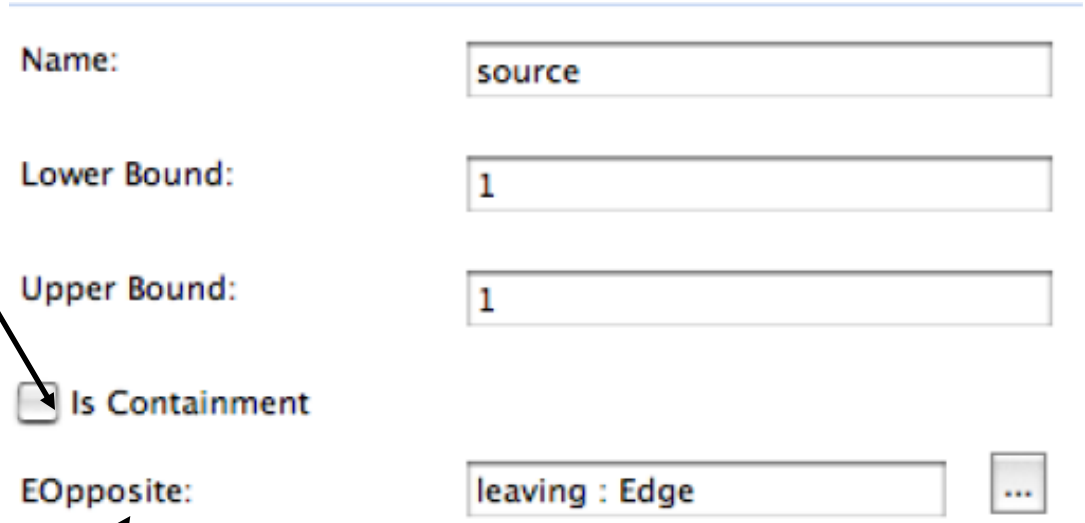
- Aussi: Par extension d'un autre Ecore

EMF: Comment créer des méta-modèles?



EMF: Comment spécifier les compositions/associations bidirectionnelles

Composition

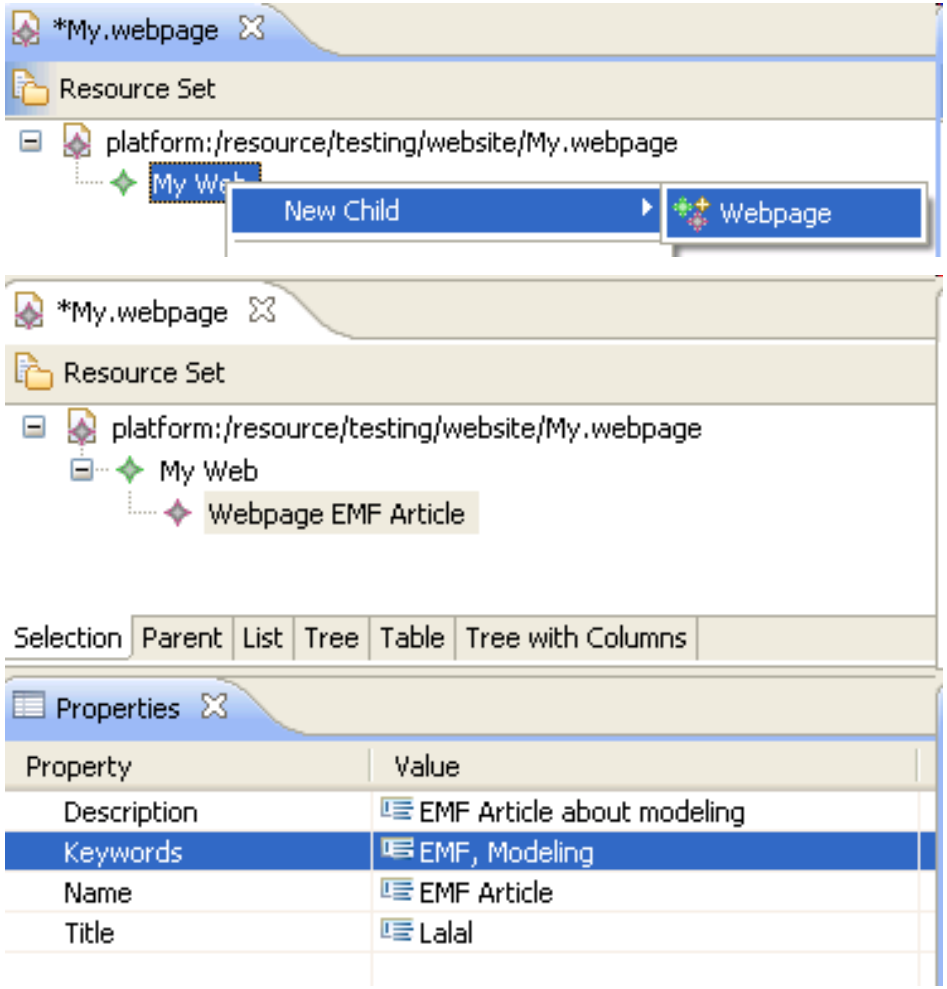


Assoc. Bidirectionnelle

Multiplicité * =-1 en EMF

EMF: Créer des modèles instances

- Le Framework offre la possibilité de créer des modèles instances de votre DSML à partir de l'Ecore
  - Éditeur arborescent
  - Notion de *dynamic instance*



EMF: Génération du code

- EMF offre la génération de code à partir de modèles
- Code: Interfaces + Classes Java
  - Pour manipuler les modèles/les créer/Les sauvegarder, etc.
    - Éléments du modèles, Factories, Adaptateurs, etc.
    - Héritent de la classe EMF EObject
  - Pour assurer la consistance par rapport au méta-modèle
  - Génération de l'éditeur de modèles + les cas de tests
- Les classes Java sont générées à partir d'un modèle *.genmodel*
  - Lui-même obtenu automatiquement à partir de l'ecore
  - Contient les informations pour la génération vers Java

EMF: Génération du code

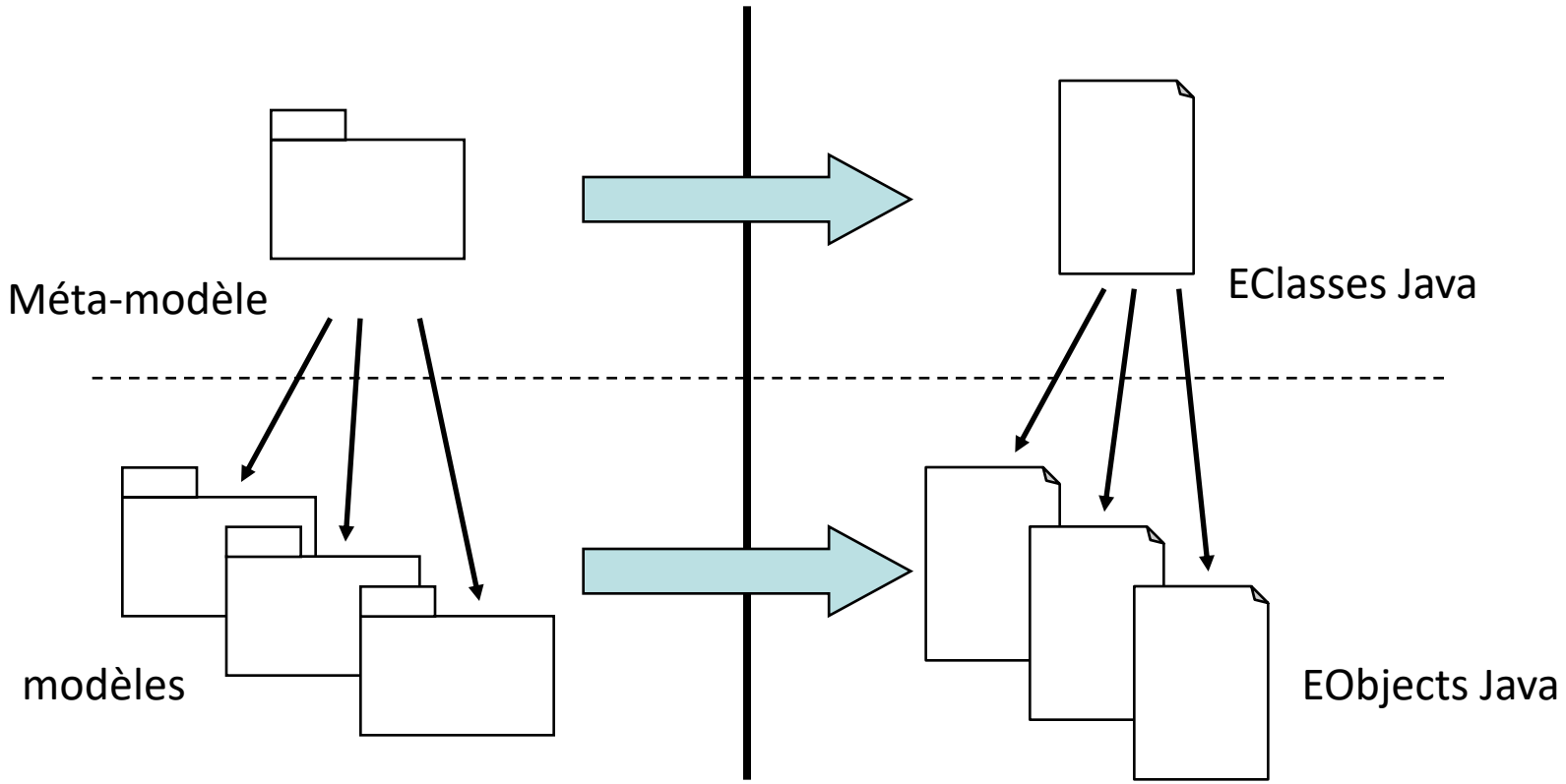


Fig. X. Blanc

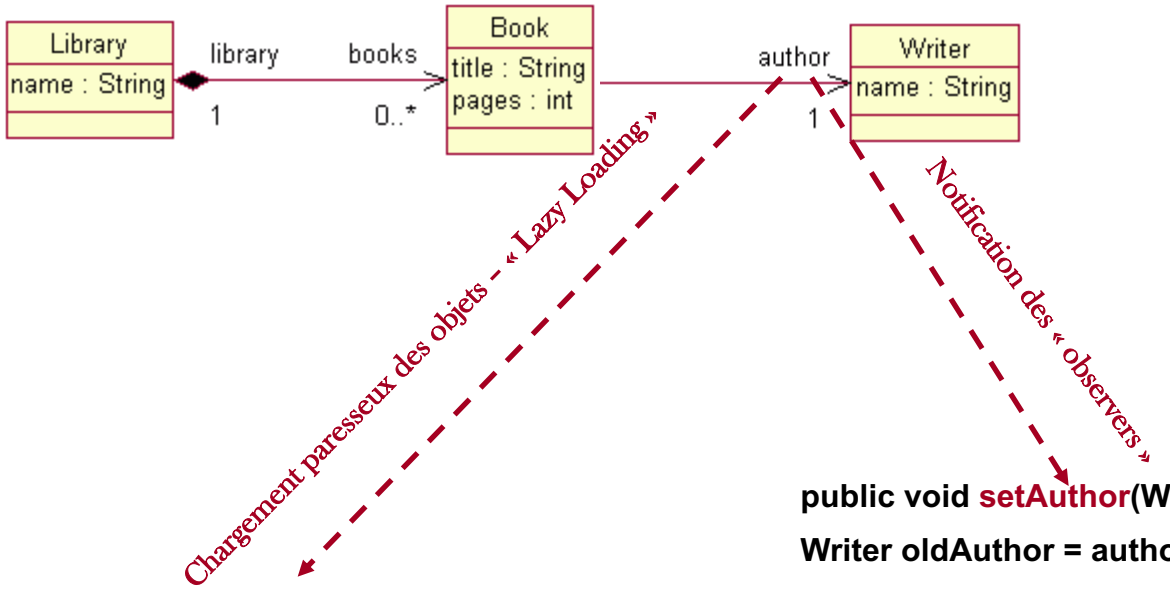
EMF: Génération du code

- En cas de re-génération de code, tous les éléments annotés par @generated seront remplacés/effacés

- Afin de préserver votre code, vos ajouts, annoter avec @generated NOT

```
/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public String getName()
{
  return name;
}
```

EMF: Génération du code




```
public void setAuthor(Writer newAuthor) {
Writer oldAuthor = author;
author = newAuthor; if(eNotificationRequired())
eNotify(new ENotificationImpl(this, ...));
}
```

```
public Writer getAuthor() {
if (author != null && author.eIsProxy()) {
Writer oldAuthor = author;
author = (Writer)eResolveProxy((InternalEObject)author);
if (author != oldAuthor) {
if (eNotificationRequired())
eNotify(new ENotificationImpl(this, Notification.RESOLVE, ...)); } }
return author; }
```

EMF: Manipulation des modèles

- Sur l'exemple de la « Library »
- Un bout de code pour créer des instances de Book et Writer

```
LibraryFactory factory = LibraryFactory.eINSTANCE;
Book book = factory.createBook();
Writer writer = factory.createWriter();
writer.setName("William Shakespeare");
book.setTitle("King Lear");

book.setAuthor(writer);
```

Si l'association était bidirectionnelle entre Book et Writer, le lien se fait automatiquement grâce aux adaptateurs générés automatiquement par EMF

EMF: Manipulation des modèles

Pour sauvegarder votre modèle dans un fichier xmi (si le fichier n'existe pas encore)

```
// Create a resource set.
        ResourceSet resourceSet = new ResourceSetImpl();

// Register the default resource factory
        resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap().put(
        Resource.Factory.Registry.DEFAULT_EXTENSION, new XMIResourceFactoryImpl());

// Get the URI of the model file.
        URI fileURI = URI.createFileURI(new File("mylibrary.xmi").getAbsolutePath());

// Create a resource for this file.
        Resource resource = resourceSet.createResource(fileURI);

// Add the book and writer objects to the contents.
        resource.getContents().add(book);
        resource.getContents().add(writer);

// Save the contents of the resource to the file system.
        try { resource.save(Collections.EMPTY_MAP); }

        catch (IOException e) {}
```

EMF: Manipulation des modèles

Pour charger votre modèle à partir d'un fichier xmi

```
// Create a resource set.
    ResourceSet resourceSet = new ResourceSetImpl();
// Register the default resource factory
    resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap().put(
    Resource.Factory.Registry.DEFAULT_EXTENSION, new XMIResourceFactoryImpl());
// Register the package
    LibraryPackage libraryPackage = LibraryPackage.eINSTANCE;
// Get the URI of the model file.
    URI fileURI = URI.createFileURI(new File("mylibrary.xmi").getAbsolutePath());
// Demand load the resource for this file.
    Resource resource = resourceSet.getResource(fileURI, true);
// Print the contents of the resource to System.out.
    try { resource.save(System.out, Collections.EMPTY_MAP); } catch (IOException e) {}
```

EMF: Conclusion

- Framework très puissant
- Commence à être mature, mais pas tous les projets!
- Une communauté qui ne cesse de croître!
- Bien documenté, pas mal de forums
- Dès qu'on sort des exemples simples, ça commence.....

Les Profils ou l'autre manière de créer vos DSML

-Présentation

-Concepts

-Exemple

Profils UML

Un profil permet d'adapter UML pour :

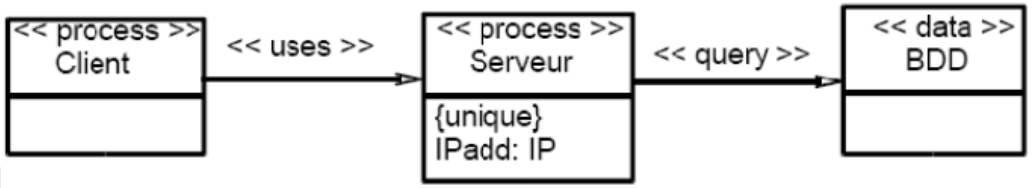
- Préciser une terminologie propre à un domaine ou à une plateforme
  - Exp. Terminologie EJB : Stateless, Statefull, Entity
- Donner une syntaxe à des concepts qui n'en ont pas
  - Les actions, des noeuds de contrôle dans le domaine du temps réel
- Donner une notation différente à des symboles existants
  - Une image d'ordinateur au lieu d'un nœud ordinaire pour représenter un ordinateur dans un réseau
- Ajouter de la sémantique
  - Gestion des priorités à la réception d'un signal, des timers

Profils UML: Concepts

- Un profil UML est composé de 3 types d'éléments
  - Des stéréotypes
  - Des tagged value
  - Des contraintes (exprimables en OCL)
    - Sur ces stéréotypes, tagged values
    - Sur des éléments du méta-modèle existant
    - Sur les relations entre les éléments
- Un profil UML est défini sous la forme d'un package stéréotypé << profile >>
- Exemple de profil : architecture logicielle
  - Des composants clients et serveur
  - Un client est associé à un serveur via une interface de service par l'intermédiaire d'un proxy

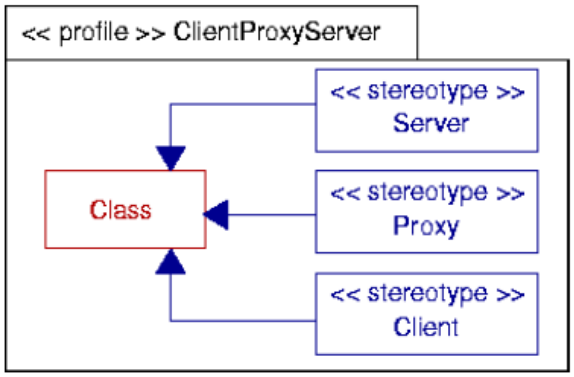
Profils UML: Concepts

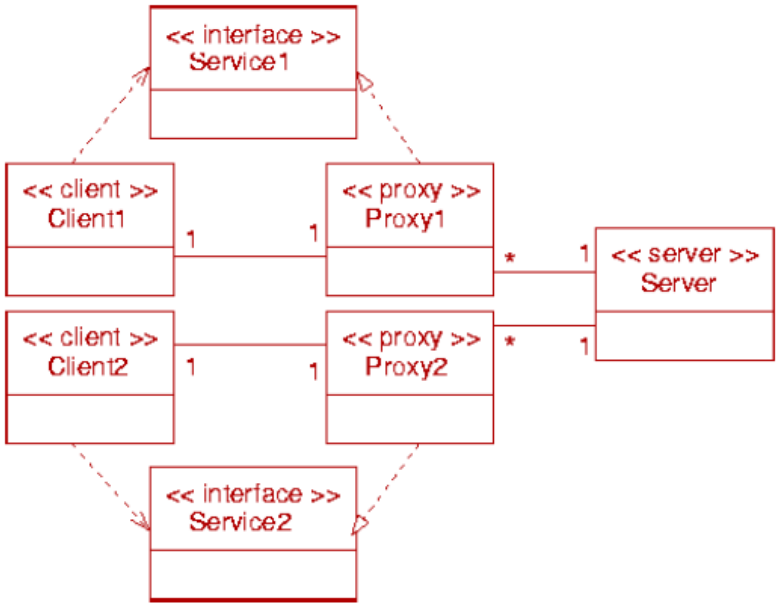
- Stéréotype
  - Extension, spécialisation d'un élément du métamodèle
  - Classe, association, attribut, opération ...
  - Le nom d'un stéréotype est indiqué entre << ... >>
  - Il existe déjà des stéréotypes définis dans UML
    - << interface >> : une interface est un classifier particulier (sans attribut)
- Tagged value (valeur marquée)
  - Pour marquer des attributs d'une classe pour préciser une contrainte ou un rôle particulier
    - Exemple : {unique} id: int



Profils UML: Exemple

- Profil défini
  - nommé ClientProxyServer
  - Définit trois stéréotypes
    - Trois classes jouant un rôle particulier : extensions de la métaclasse Class du méta-modèle UML
      - Server, Proxy, Client





Profils UML: Exemple

- Pour compléter le profil, ajout de contraintes OCL
  - Navigation sur le métamodèle UML (simplifié) en considérant que la méta-classe Class a trois spécialisations (Server, Client, Proxy)
  - Un proxy associé à un client doit implémenter une des interfaces dont dépend le client et un proxy implémentant une interface d'un client doit avoir une association avec ce client
- **context** Client **inv:**

  let proxies = self.associationEnd.association.associationEnd.class -> select (c | c.oclIsTypeOf(Proxy)) **in**

  let interfaces = self.dependsOn **in**

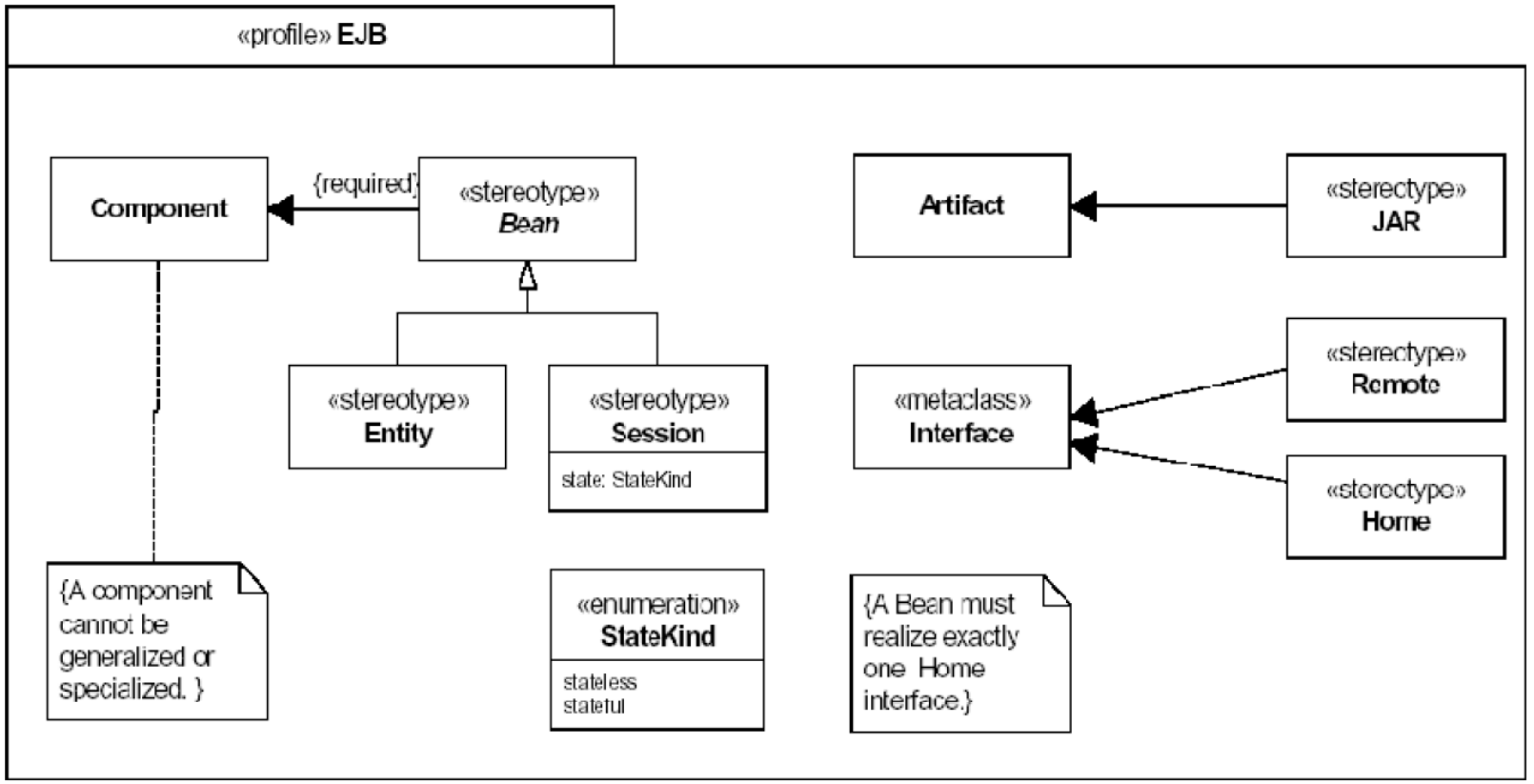
  interfaces -> forAll (i | proxies.implements -> includes (i) **and**

  proxies -> forAll (p | p.implements -> includes (i)

  implies p.hasClassRefWith(self)))
- **context** Class **def:** hasClassRefWith(cl : Class) :

  Boolean = self.associationEnd.association.associationEnd.class-> exists (c| c = cl)

Profils UML: Exemple d'un profil EJB



Profils UML: Limites

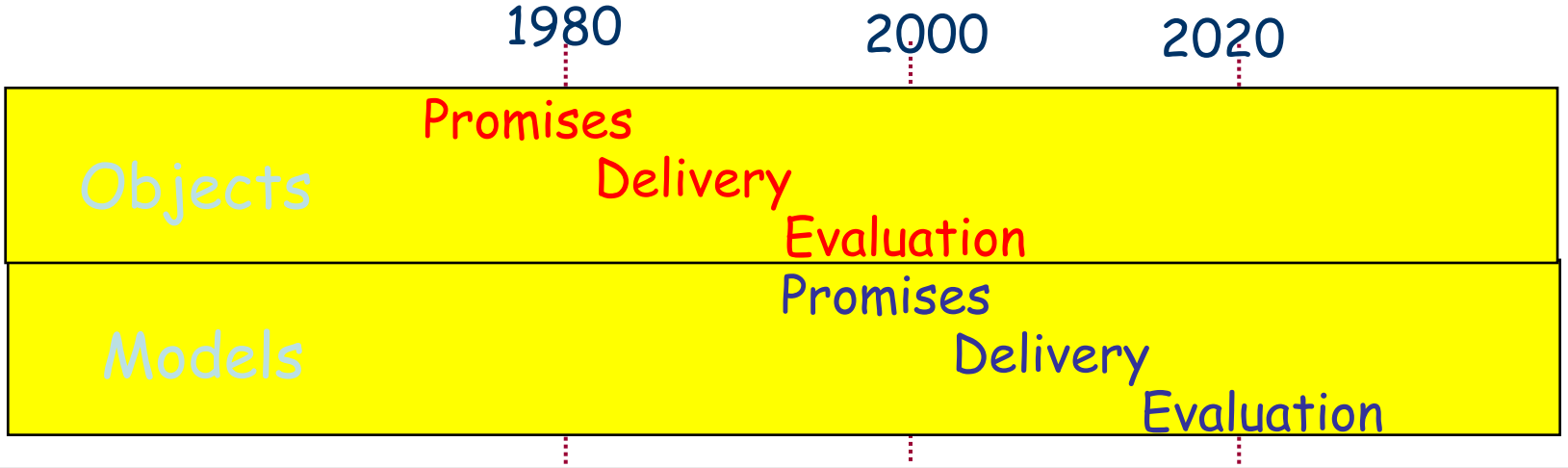
- On ne pas prendre une partie d'UML, c'est le tout ou rien ou alors écrire 1000 règles OCL
- On ne peut pas rajouter de nouveaux types
- Il faut être pro OCL
- L'outillage!

Conclusion

MDE: Recul?

- **I like to think that one day, when we grow up, perhaps we will think of the model as being the source code.** Oscar Nierstrasz *[Banquet speech given at ECOOP 2010. Maribor, June 24, 2010]*

- **Object technology realized some promises but failed to achieve others**
  - Stopping the search for generality by unification may be one of the causes for this
- **Model engineering is making many promises today**
  - Will it be able to deliver correspondingly?
  - Sticking with the principle that "everything is a model" seems a good way to make progresses



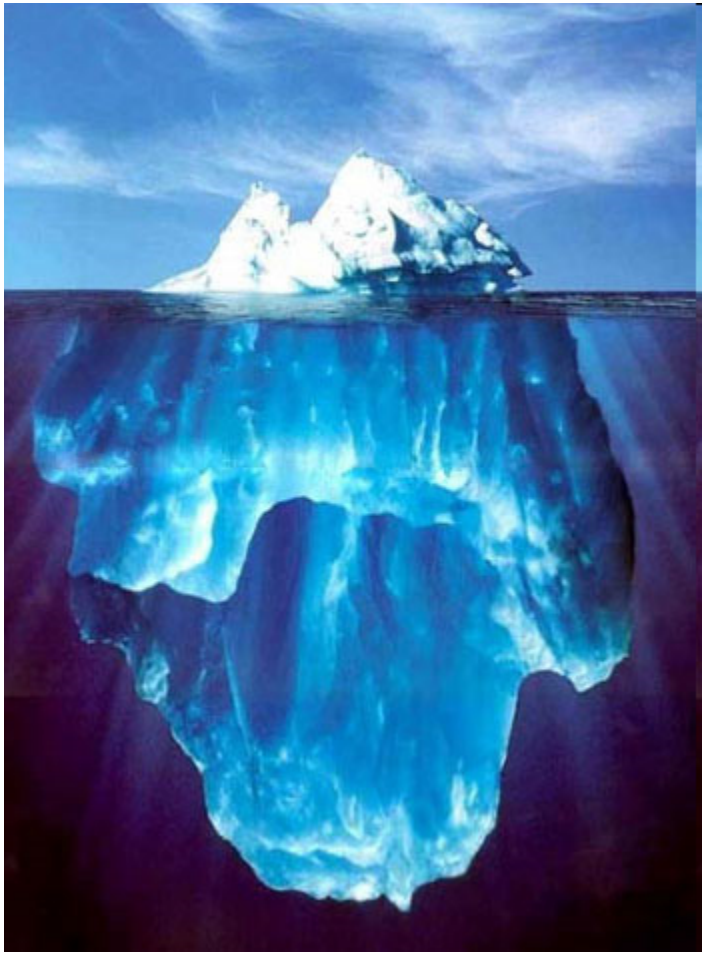
MDE: Recul?

- La métaphore de l'iceberg

- À ne pas confondre avec la brève de comptoir

« *C'est un iceberg, celui-là, sept fois plus c.. que ce qu'on voit.* »

Source: Jean-Marie Gourio - *Extrait de Brèves de comptoir* »



Lectures

- Software Engineering,
  - Ian Sommerville, Addison Wesley; 8 edition (15 Jun 2006), ISBN-10: 0321313798
- The Mythical Man-Month
  - Frederick P. Brooks JR., Addison-Wesley, 1995
- UML Distilled 3rd édition, a brief guide to the standard object modeling language
  - Martin Fowler, Addison-Wesley Object Technology Series, 2003, ISBN-10: 0321193687
- MDA en Action, de Xavier Blanc, 2005, chez Eyrolles,
- Domain-Specific Modeling: Enabling full code generation, par S. Kelly et J-P, Tolvanen, Wiley Interscience 2008
- Cours de Software Engineering du Prof. Bertrand Meyer à cette @:
  - http://se.ethz.ch/teaching/ss2007/252-0204-00/lecture.html
- Cours d'Antoine Beugnard à cette @:
  - http://public.enst-bretagne.fr/~beugnard/
- Cours très intéressants du Prof. Jean-Marc Jézéquel:
  - http://www.irisa.fr/prive/jezequel/
- Cours de Jean Bézivin, Benoit Combemale, Sébastien Mosser, Mireille –blay fornarino, Anne Etien (Google is your friend: nom + mde ou page perso)
- Cours Xavier Blanc pour l'école d'été MDE 2009 (supports non disponibles en ligne)
- La page de l'OMG dédiée à UML: http://www.omg.org/mda/ + Le guide MDA de Richard Soley sur omg.org
- Design patterns. Catalogue des modèles de conception réutilisables
  - Richard Helm (Auteur), Ralph Johnson (Auteur), John Vlissides (Auteur), Eric Gamma (Auteur), Vuibert informatique (5 juillet 1999), ISBN-10: 2711786447