

UML – Unified Modeling Language

4/4 : concepts avancés

Yannick Prié

UFR Informatique – Université Claude Bernard Lyon 1

M1 MIAGE – SIMA / M1 Informatique MIF17

2008-2009

Objectifs de ce cours

- Introduction à des concepts et des technologies plus avancés liés à UML, notamment au traitements *automatiques* des diagrammes pour *générer* des systèmes

Plan

- **Outils UML**
- Métamodèle UML
- Object constraint language
- Model Driven Development
- Conclusions sur UML

Éléments de modélisation, de visualisation et sémantique

- Les éléments de visualisation sont utilisés dans les diagrammes visuels, les éléments de modélisation forment le modèle lui-même
- La syntaxe de UML dit comment composer les éléments. La sémantique du modèle dit ce qu'ils signifient en termes objets.
- Par exemple
 - dans un diagramme « dc1 » on a mis les éléments de visualisation
 - un conteneur composé de trois parties
 - le texte « Etudiant » dans la partie du haut,
 - le texte « suit : Cours » dans la partie médiane
 - ce qui signifie
 - qu'on a les éléments de modélisation suivant
 - classes « Etudiants », « Cours »
 - l'attribut « suit » associé à la classe « Cours »
- La sémantique du langage spécifie ce que signifie un modèle objet
 - qu'une classe peut avoir des instances
 - qu'une instance a des attributs
 - qu'un instance peut être valeur d'attribut
 - etc.

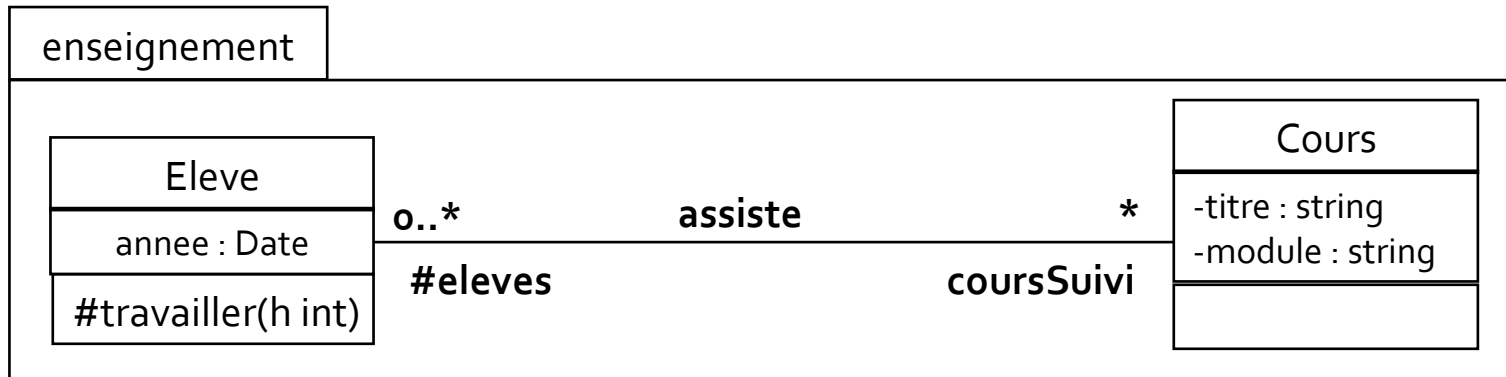
Deux types d'outils UML

- Outils de dessins améliorés
 - intègrent les diagrammes comme simples formes
 - incapable de traiter les diagrammes : pas de sémantique
- Outils UML utilisant la sémantique d'UML
 - gestion des diagrammes et du modèle
 - vérification de cohérence
 - génération de code
 - squelettes de classes / contenus des méthodes (peu)
 - rétro-ingénierie
 - diagrammes de classes
 - diagrammes de séquences (peu)
 - de plus en plus intégrés / en compléments d'autres outils
 - IDE, gestion de projet, du risque, des besoins, de la qualité, des tests, du workflow, etc.

Modèles UML et code

- Modèle avec
 - classes
 - attributs, associations
 - opérations
 - spécialisation
 - paquetages
 - interactions
- Correspond à du code dans un langage de programmation modèle dans un langage
 - Java, C++, C#, python, etc.

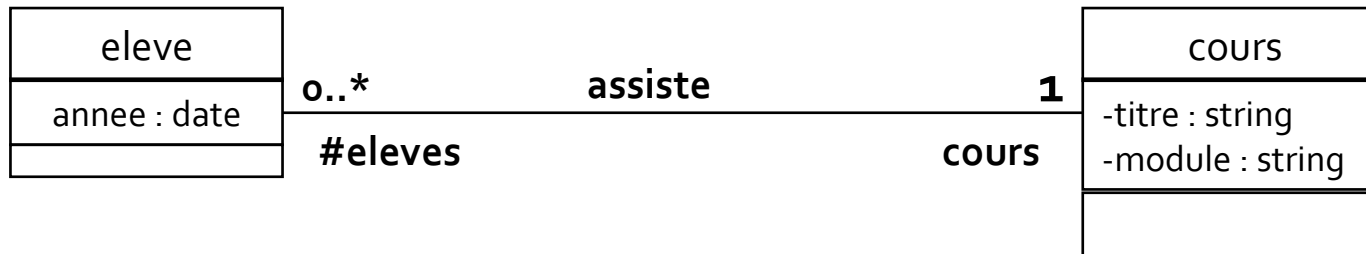
Traduction simple en objet (Java)



```
package enseignement ;
public class Eleve
{
    private Date annee ;
    private Court coursSuivi[] ;
    public Eleve() ;
    public Date getAnnee()
        { return annee ; }
    public void setAnnee(Date
uneAnnee)
        { annee = uneAnnee ; }
    ...
}
```

```
...
public associerCoursSuivi(unCours : Cours)
{ /* ecrire ici */
}
public nbCoursSuivi
{
    return coursSuivi.length ;
}
protected travailler(h : int)
{
    /* ecrire ici */
}
/* etc. */
```

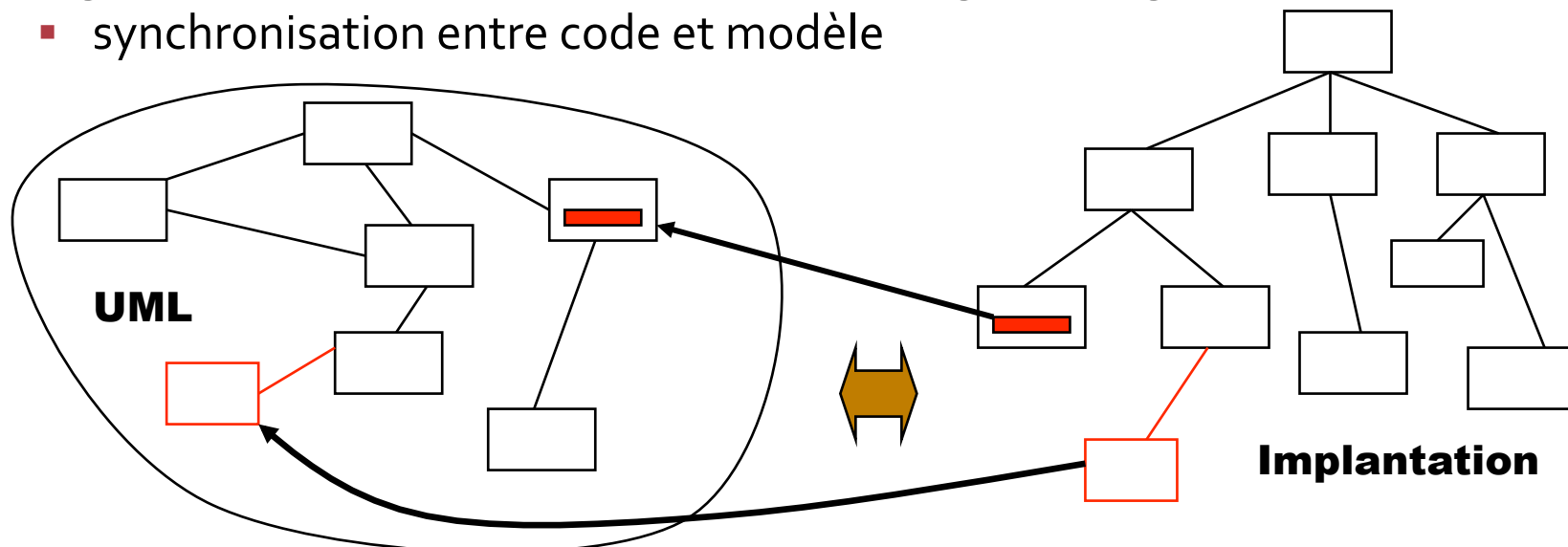
Remarque : traduction en relationnel



```
CREATE TABLE eleve (  
    eleve_id NUMBER (5),  
    annee DATE,  
    PRIMARY KEY (eleve_id)  
);  
CREATE TABLE cours (  
    eleve_id NUMBER (5) REFERENCES eleve(eleve_id),  
    cours_id NUMBER (5),  
    titre CHAR (128),  
    module CHAR(48),  
    PRIMARY KEY (cours_id)  
);
```


Ingénierie UML

- Pro-ingénierie
 - générer le code à partir du modèle
 - outils : paramétrage (héritage, associations...)
- Rétro-ingénierie
 - générer le modèle à partir de l'implantation
 - seuls les outils automatiques peuvent le faire
- Ingénierie bidirectionnelle (roundtrip engineering)
 - synchronisation entre code et modèle



Comment aller plus loin ?

- Exprimer toute la sémantique objet des diagrammes (pas seulement les classes)
 - Métamodèle UML, Meta Object Facility
- Etre capable d'exprimer des contraintes dans un langage formel
 - Object Constraint Language = OCL
- Pour atteindre l'objectif de concevoir et programmer en même temps
 - Model Driven Development

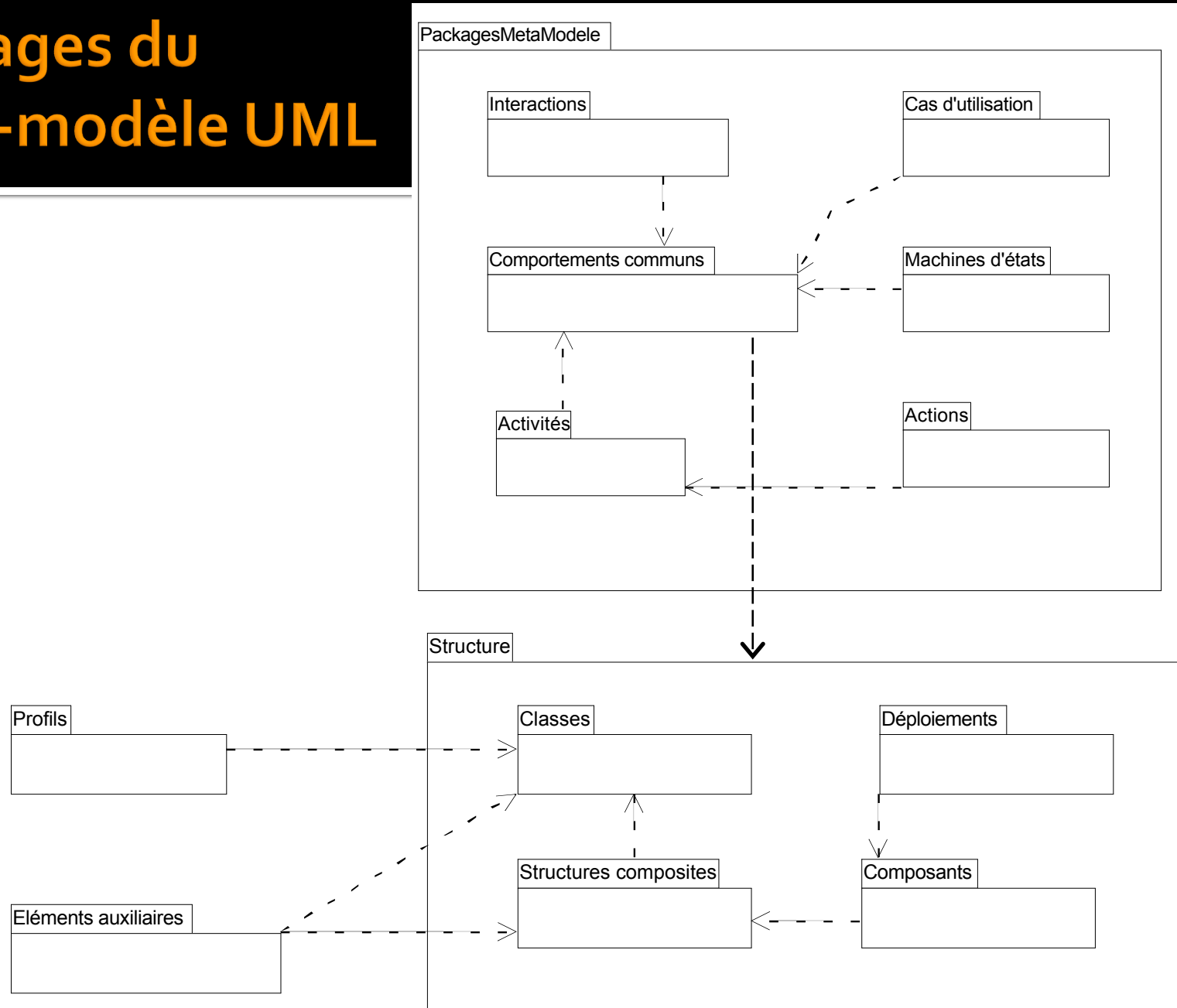
Plan

- Outils UML
- **Métamodèle UML**
- Object constraint language
- Model Driven Development
- Conclusions sur UML

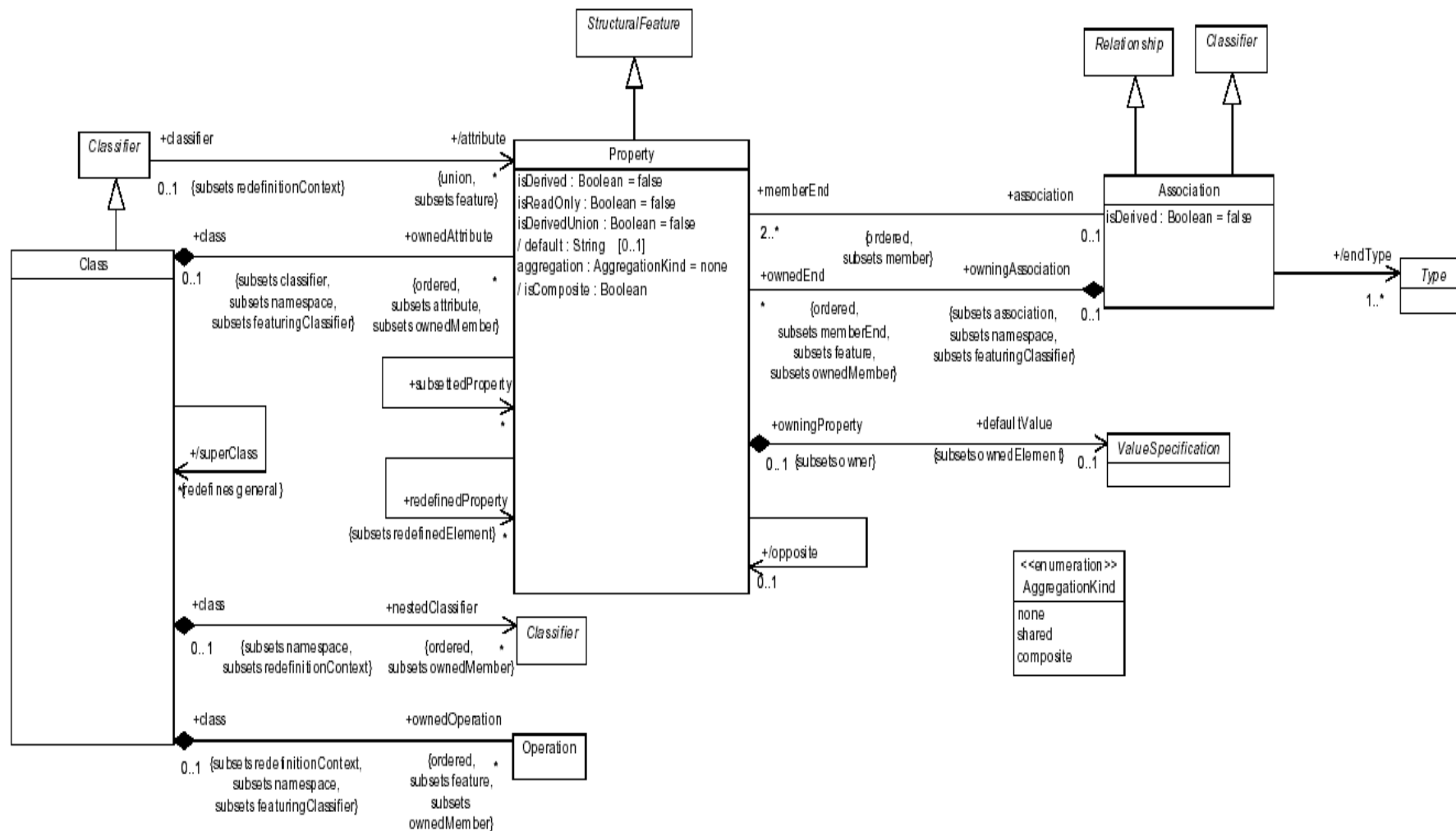
Méta-modèle

- A la base, l'ensemble de UML est décrit en UML
- Méta-modèle UML
 - description formelle de tout ce qu'il est possible de construire et de la sémantique associée
 - nécessaire pour les fabricants d'outils
« sémantiquement pertinents »
- Dans le métamodèle
 - Essentiellement des diagrammes de classes avec contraintes et la description de la signification dynamique des éléments

Packages du méta-modèle UML

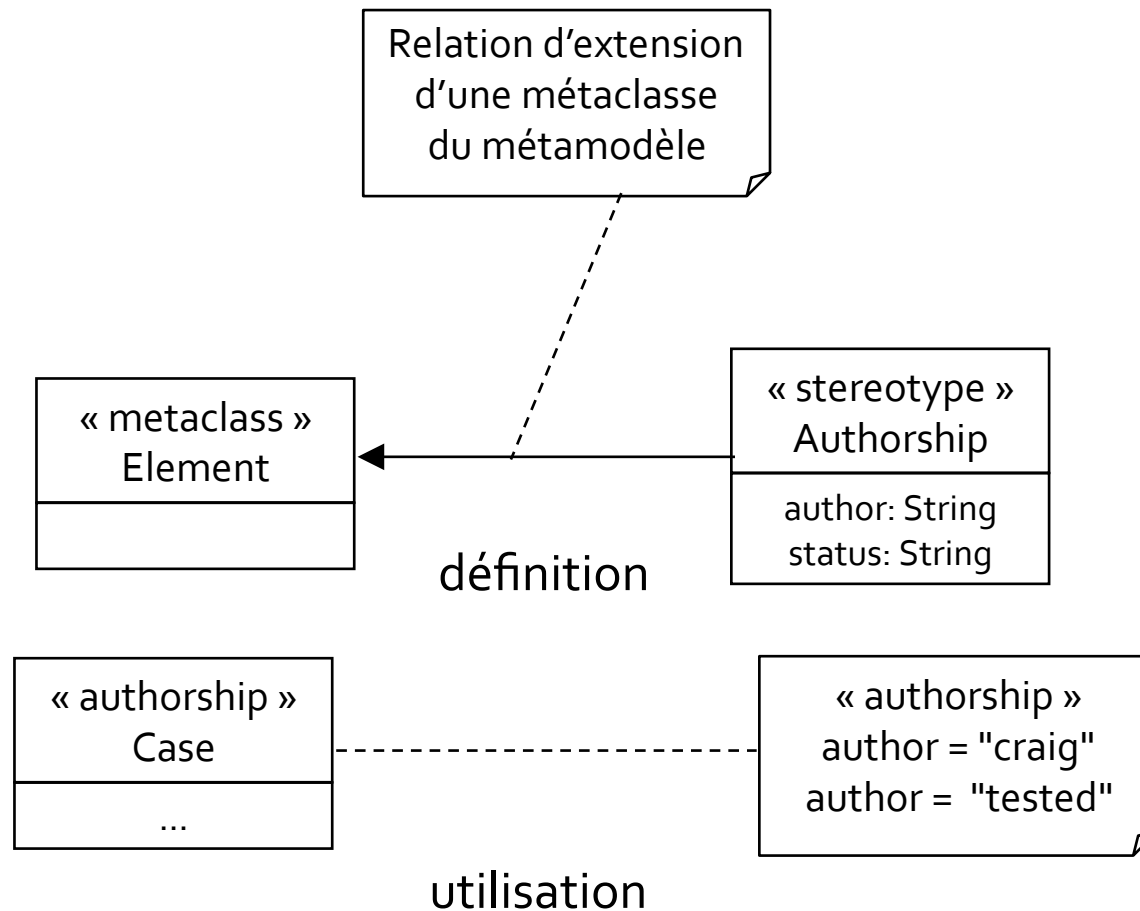


Extrait du méta-modèle



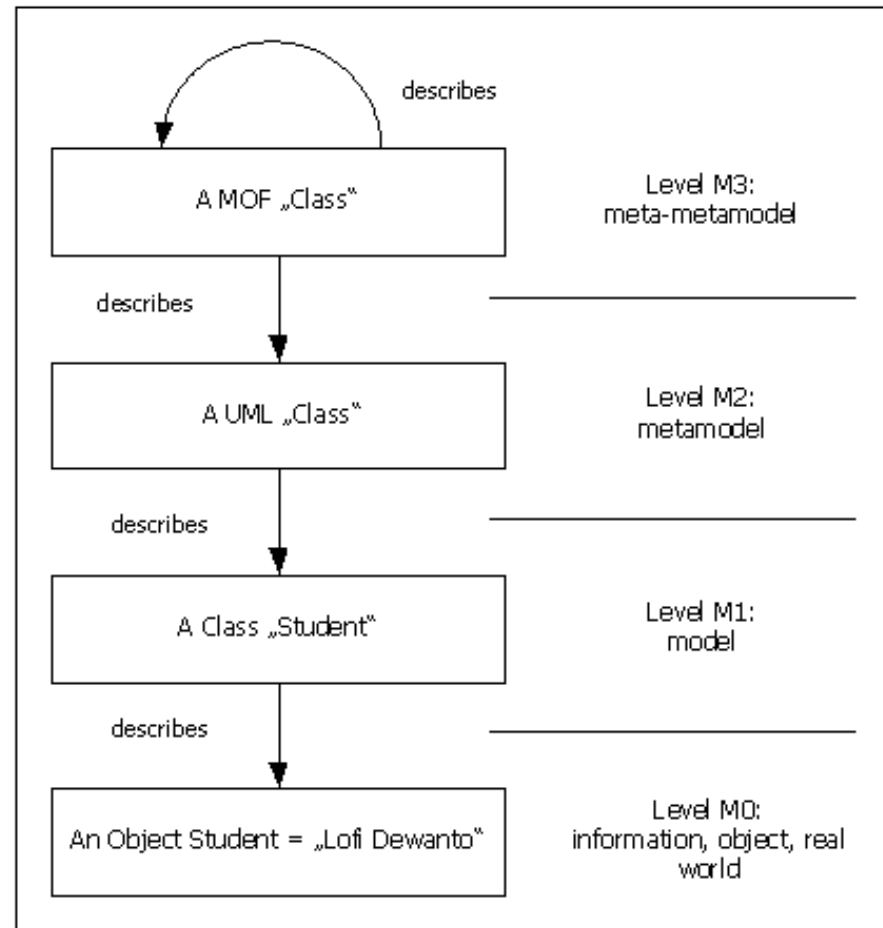
D'après OMG UML2 Superstructure, Figure 30

Extension d'UML : stéréotypes



Encore plus loin : MOF

- UML₁ → UML₂
 - très gros travail sur le métamodèle pour que rien ne reste dans l'ombre (informel)
 - toutes les conséquences de l'utilisation d'un élément de modélisation doivent être connues et maîtrisées
- Meta Object Facility
 - Métamétamodèle autodécrit permettant de décrire des métamodèles
 - UML
 - CWM
 - Common Warehouse Metamodel : structure de BD



Interopérabilité entre outils UML

- XML Metadata Interchange (OMG)
 - standard basé sur MOF pour échanger des modèles
 - syntaxe XML (eXtensible Markup Language)
- XMI permet d'échanger des modèles UML entre outils UML
- Remarques
 - génération de documentation HTML
 - transformation XSL
 - diagrammes
 - transformation en SVG (Standard vector Graphics)
 - dans les faits, implémentations pas forcément cohérentes !

Plan

- Outils UML
- Métamodèle UML
- **Object constraint language**
- Model Driven Development
- Conclusions sur UML

Contrainte

- Condition ou restriction sémantique associée à un ou plusieurs éléments de modèle exprimée
 - en langue standard (exemple : français)
 - dans un langage formel
- Assertion qui doit être vraie
 - entre des appels d'opérations (qui changent le système)
 - à des moments précis par rapport aux appels d'opérations

OCL Object Constraint Language

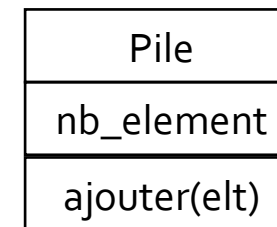
- Standardisé par l'OMG
- Permet d'exprimer des contraintes de façon formelle
- Expression
 - d'invariants au sein d'une classe ou d'un type : bon fonctionnement des instances
 - contraintes au sein d'une opération : bon fonctionnement de l'opération
 - pré- et post- conditions d'opérations : avant et après l'exécution
 - cf. programmation par contrats (Meyer)
 - gardes : sur la modification de l'état d'un objet
 - expressions de navigation : chemins
- Utilisation
 - génération de code
 - MDA

OCL : trois exemples

context nom_élément [**inv**|**pre**|**post**] : expression de la contrainte

context Pile **inv** :

self.nb_elements >= 0 -- nb_element = attribut de Pile



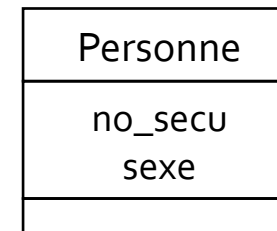
context Personne **inv** -- intégrité de l'objet personne

/ attributs no_secu et sexe

if sexe = "F" **then** no_secu.commence_par() = 2

else no_secu.commence_par() = 1

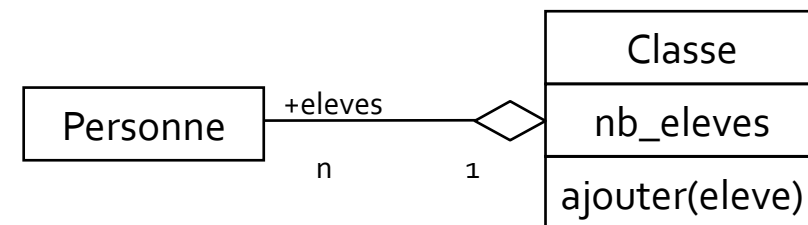
endif



context Classe::ajouter(un_eleve : eleve)

pre classe_non_surchargée : nb_eleves <= 25

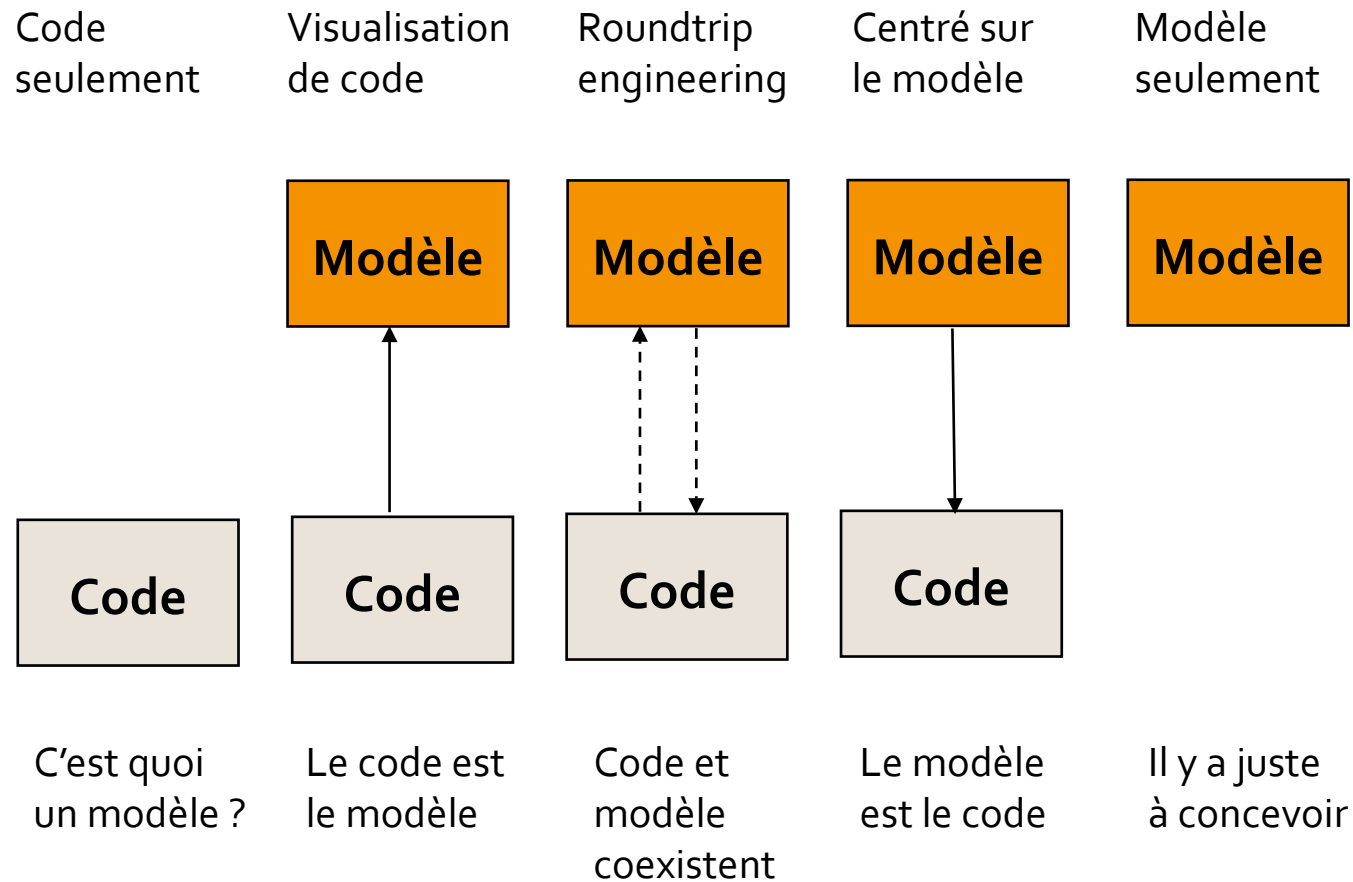
post : eleves → exists(un_eleve)



Plan

- Outils UML
- Métamodèle UML
- Object constraint language
- **Model Driven Development**
- Conclusions sur UML

Modèles et code



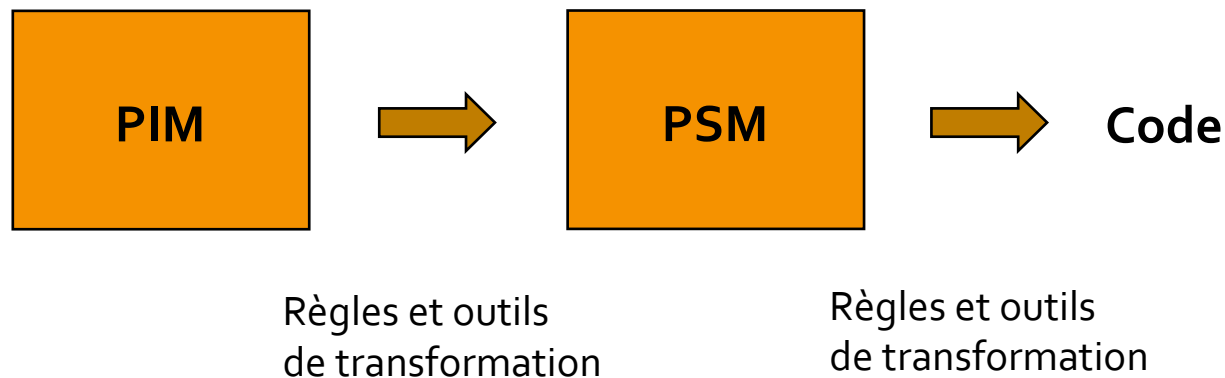
(d'après <http://www-128.ibm.com/developerworks/rational/library/3100.html>)

MDA : Model Driven Architecture

- Architecture pilotée par les modèles
 - mis en place et supporté par l'OMG
 - <http://www.omg.org/mda/>
 - UML comme langage de programmation
 - passer d'un développement centré sur le code à un développement centré sur les modèles
- Pour permettre, de la façon la plus intégrée possible
 - productivité
 - portabilité
 - interopérabilité
 - maintenance
 - documentation

MDA

- Deux types de modèles
 - PIM (Platform Independant Model)
 - en UML
 - PSM (platform specific model)
 - pas obligatoirement en UML



MDD / MDE

- Termes plus généraux, non OMG
 - MDD : Model Driven Development
 - MDE : Model Driven Engineering
- Principe de développement qui consiste à considérer
 - qu'une part importante de l'activité consiste à créer des modèles décrivant les éléments d'un système,
 - que ces tâches doivent être instrumentées pour faciliter le travail des développeurs
 - gérer les modèles, leur évolution, leur stockage, leur échange, les liens à leur utilisation, etc.
- Exemples
 - MDA – OMG
 - EMF : Eclipse Modeling Framework
 - autour de modèles exprimés en XMI

MDA : conclusion

- Pour certains
 - l'avenir de l'informatique
 - des outils existent
 - des « leaders » utilisent et promeuvent
- D'autres sont moins convaincus
 - entre
 - « ça ne marchera jamais »
 - « je demande à voir »
- Acceptation
 - possibilité de programmer mieux et plus vite (moins cher)
 - rapport apprentissage / gain
 - dans les entreprises
 - dans les écoles
- A suivre...

Plan

- Outils UML
- Métamodèle UML
- Object constraint language
- Model Driven Development
- **Conclusions sur UML**

Conclusions sur UML

- Propriétés d'UML
 - unification de concepts de modélisation
 - puissance d'expression
 - nombreux formalismes (issus de méthodes existantes)
 - compromis formalisation / niveau d'abstraction / indépendance aux langages / sémantique fixée / extensibilité
- Langage universel
 - pour de multiples domaines
 - pour diverses activités de la conception
 - dans différents modes
 - esquisse, plan, programme

Conclusions sur UML

- Standard international : adopté un peu partout
 - les diagrammes sont simples, faciles à lire et à communiquer
 - beaucoup de variantes locales
 - outils puissants
 - dessin
 - pro et rétro ingénierie
 - MDA
- UML n'est qu'un langage
 - encapsule tout ou partie de la sémantique de description
 - mais ne dit pas *comment* construire les modèles
- Il faut utiliser des méthodes
 - démarches de conception et d'utilisation des diagrammes et des modèles qui expliquent les bonnes manières de concevoir