Ingénierie Dirigée par les Modèles

Olivier Caron

Polytech Lille
Avenue Paul Langevin Cité Scientifique
Université de Lille
59655 Villeneuve d'Ascq cedex

http://ocaron.polytech-lille.net
Olivier.Caron@polytech-lille.fr

Plan

- 3 séances de cours (6h) :

Plan

- 3 séances de cours (6h) :
  - L'approche IDM ou MDE (Model Driven Engineering) :

Plan

- 3 séances de cours (6h) :
  - L'approche IDM ou MDE (Model Driven Engineering) :
    - Vers des artefacts manipulables (vérifiables, composables, transformables, . . .)

Plan

- 3 séances de cours (6h) :
  - L'approche IDM ou MDE (Model Driven Engineering) :
    - Vers des artefacts manipulables (vérifiables, composables, transformables, . . .)
    - La vision descendante de l'IDM : MDA (Model Driven Architecture)

Plan

- 3 séances de cours (6h) :
  - L'approche IDM ou MDE (Model Driven Engineering) :
    - Vers des artefacts manipulables (vérifiables, composables, transformables, . . .)
    - La vision descendante de l'IDM : MDA (Model Driven Architecture)
  - l'IDM et patterns au cœur de la société Axellience (GenMyModel).

Plan

- 3 séances de cours (6h) :
  - L'approche IDM ou MDE (Model Driven Engineering) :
    - Vers des artefacts manipulables (vérifiables, composables, transformables, . . .)
    - La vision descendante de l'IDM : MDA (Model Driven Architecture)
  - l'IDM et patterns au cœur de la société Axellience (GenMyModel).
- 3 séances de TP (10h) : tutoriels (Modeling Tools Eclipse) et projet IDM (méta-modélisation, édition, programmation, vérification, transformation)

Plan

- 3 séances de cours (6h) :
  - L'approche IDM ou MDE (Model Driven Engineering) :
    - Vers des artefacts manipulables (vérifiables, composables, transformables, . . .)
    - La vision descendante de l'IDM : MDA (Model Driven Architecture)
  - l'IDM et patterns au cœur de la société Axellience (GenMyModel).
- 3 séances de TP (10h) : tutoriels (Modeling Tools Eclipse) et projet IDM (méta-modélisation, édition, programmation, vérification, transformation)

Merci à

Anne Etien pour avoir initié ce cours IDM,
Bernard Carré pour la relecture et les améliorations apportées.

Constat sur le développement logiciel

- Un univers en constante évolution :

Constat sur le développement logiciel

- Un univers en constante évolution :
  - Multiplication des plates-formes technologiques (JEE, .Net, Android, Web 2.0,. . .)

Constat sur le développement logiciel

- Un univers en constante évolution :
  - Multiplication des plates-formes technologiques (JEE, .Net, Android, Web 2.0,. . .)
  - Obsolescence rapide des logiciels (60% de l'activité informatique consiste en la maintenance et l'évolution)

Constat sur le développement logiciel

- Un univers en constante évolution :
  - Multiplication des plates-formes technologiques (JEE, .Net, Android, Web 2.0,. . .)
  - Obsolescence rapide des logiciels (60% de l'activité informatique consiste en la maintenance et l'évolution)
  - Des systèmes de plus en plus complexes (réseau, sécurité, . . .)

Constat sur le développement logiciel

- Un univers en constante évolution :
  - Multiplication des plates-formes technologiques (JEE, .Net, Android, Web 2.0,. . .)
  - Obsolescence rapide des logiciels (60% de l'activité informatique consiste en la maintenance et l'évolution)
  - Des systèmes de plus en plus complexes (réseau, sécurité, . . .)
- Des exigences supérieures :

Constat sur le développement logiciel

- Un univers en constante évolution :
  - Multiplication des plates-formes technologiques (JEE, .Net, Android, Web 2.0,. . .)
  - Obsolescence rapide des logiciels (60% de l'activité informatique consiste en la maintenance et l'évolution)
  - Des systèmes de plus en plus complexes (réseau, sécurité, . . .)
- Des exigences supérieures :
  - Coût et temps de production

Constat sur le développement logiciel

- Un univers en constante évolution :
  - Multiplication des plates-formes technologiques (JEE, .Net, Android, Web 2.0,. . .)
  - Obsolescence rapide des logiciels (60% de l'activité informatique consiste en la maintenance et l'évolution)
  - Des systèmes de plus en plus complexes (réseau, sécurité, . . .)
- Des exigences supérieures :
  - Coût et temps de production
  - Qualité de production

L'approche MDA (1/2)

- L'architecture dirigée par les modèles (MDA) est une démarche de réalisation de logiciels, proposée par l'OMG.

L'approche MDA (1/2)

- L'architecture dirigée par les modèles (MDA) est une démarche de réalisation de logiciels, proposée par l'OMG.
- Consortium **O**bject **M**anagement **G**roup (`http://www.omg.org`)

L'approche MDA (1/2)

- L'architecture dirigée par les modèles (MDA) est une démarche de réalisation de logiciels, proposée par l'OMG.
- Consortium **O**bject **M**anagement **G**roup (`http://www.omg.org`)

Principe de base

Élaboration de différents modèles partant d'un modèle métier indépendant d'une plate-forme cible puis la transformation de ce dernier en modèle spécifique à la plate-forme cible.

L'approche MDA (1/2)

- L'architecture dirigée par les modèles (MDA) est une démarche de réalisation de logiciels, proposée par l'OMG.
- Consortium **O**bject **M**anagement **G**roup (`http://www.omg.org`)

Principe de base

Élaboration de différents modèles partant d'un modèle métier indépendant d'une plate-forme cible puis la transformation de ce dernier en modèle spécifique à la plate-forme cible.

- MDA est à l'origine de l'IDM

L'approche MDA (1/2)

- L'architecture dirigée par les modèles (MDA) est une démarche de réalisation de logiciels, proposée par l'OMG.
- Consortium **O**bject **M**anagement **G**roup (`http://www.omg.org`)

Principe de base

Élaboration de différents modèles partant d'un modèle métier indépendant d'une plate-forme cible puis la transformation de ce dernier en modèle spécifique à la plate-forme cible.

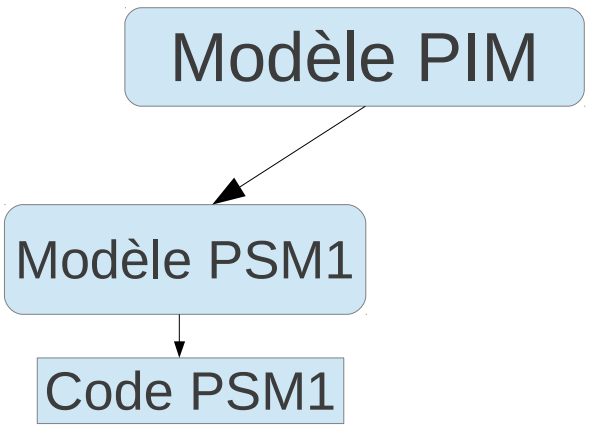
- MDA est à l'origine de l'IDM
- Vision générative (ou descendante) de l'IDM

l'approche MDA (2/2)

- **P**latform **I**ndependent **M**odel

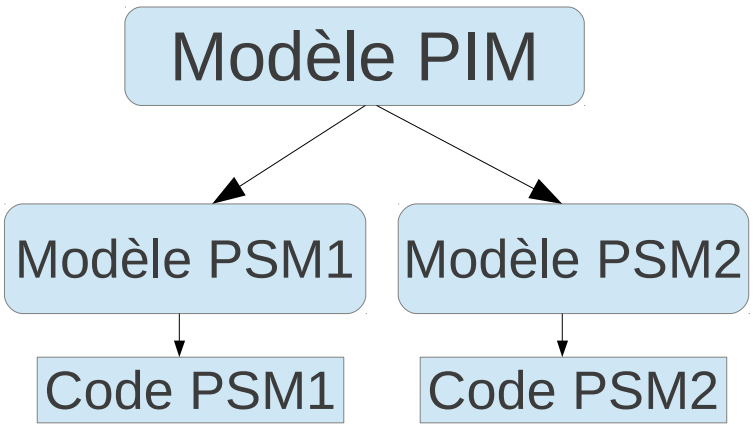
Modèle PIM

l'approche MDA (2/2)



- **P**latform **I**ndependent **M**odel
- **P**latform **S**pecific **M**odel
  (ex : Java, C#, Web,. . .)

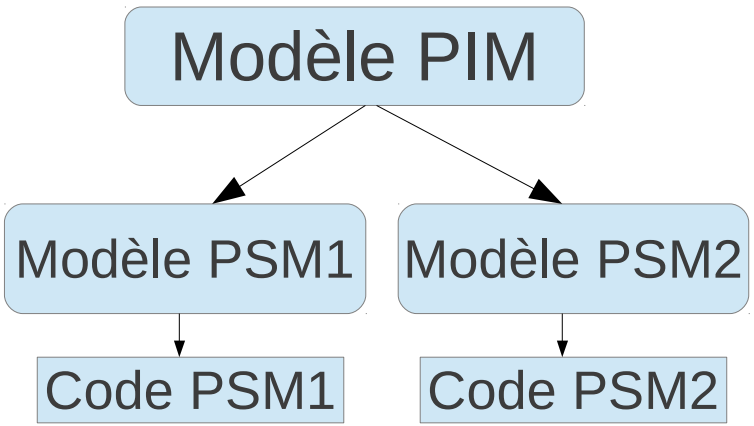
l'approche MDA (2/2)



- **P**latform **I**ndependent **M**odel
- **P**latform **S**pecific **M**odel
  (ex : Java, C#, Web,. . .)
- Génération de code
  (ex : UML to SQL, UML to Java, . . .))

"Model Once, Generate Anywhere"

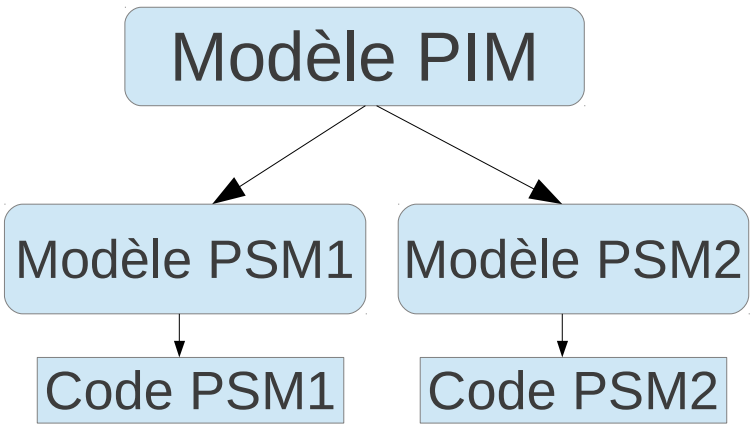
l'approche MDA (2/2)



"Model Once, Generate Anywhere"

- **P**latform **I**ndependent **M**odel
- **P**latform **S**pecific **M**odel
  (ex : Java, C#, Web,. . .)
- Génération de code
  (ex : UML to SQL, UML to Java, . . .))
- Avantages MDA :
  capitalisation du "code" métier,
  indépendance des plates-formes
  (migration, . . .),
  génération partielle de code.

l'approche MDA (2/2)



"Model Once, Generate Anywhere"

- **P**latform **I**ndependent **M**odel
- **P**latform **S**pecific **M**odel
  (ex : Java, C#, Web,. . .)
- Génération de code
  (ex : UML to SQL, UML to Java, . . .))
- Avantages MDA :
  capitalisation du "code" métier,
  indépendance des plates-formes
  (migration, . . .),
  génération partielle de code.

Les techniques de base associées à l'IDM dans le cadre MDA

- Des techniques de **méta-**modélisation :

Les techniques de base associées à l'IDM dans le cadre MDA

- Des techniques de **méta-**modélisation :
  - Soit en spécifiant un nouveau type de modèle (DSM, MOF)

Les techniques de base associées à l'IDM dans le cadre MDA

- Des techniques de **méta-**modélisation :
  - Soit en spécifiant un nouveau type de modèle (DSM, MOF)
  - Soit en utilisant UML ou une extension d'UML via la notion de **profil**

Les techniques de base associées à l'IDM dans le cadre MDA

- Des techniques de **méta-**modélisation :
  - Soit en spécifiant un nouveau type de modèle (DSM, MOF)
  - Soit en utilisant UML ou une extension d'UML via la notion de **profil**
  - Sérialisation des modèles (XMI)

Les techniques de base associées à l'IDM dans le cadre MDA

- Des techniques de **méta-**modélisation :
  - Soit en spécifiant un nouveau type de modèle (DSM, MOF)
  - Soit en utilisant UML ou une extension d'UML via la notion de **profil**
  - Sérialisation des modèles (XMI)
- Des techniques de vérification de modèles (OCL)

Les techniques de base associées à l'IDM dans le cadre MDA

- Des techniques de **méta-**modélisation :
  - Soit en spécifiant un nouveau type de modèle (DSM, MOF)
  - Soit en utilisant UML ou une extension d'UML via la notion de **profil**
  - Sérialisation des modèles (XMI)
- Des techniques de vérification de modèles (OCL)
- Des techniques de transformation (QVT) :

Les techniques de base associées à l'IDM dans le cadre MDA

- Des techniques de **méta-**modélisation :
  - Soit en spécifiant un nouveau type de modèle (DSM, MOF)
  - Soit en utilisant UML ou une extension d'UML via la notion de **profil**
  - Sérialisation des modèles (XMI)
- Des techniques de vérification de modèles (OCL)
- Des techniques de transformation (QVT) :
  - De modèle à modèle (PIM vers PSM)

Les techniques de base associées à l'IDM dans le cadre MDA

- Des techniques de **méta-**modélisation :
  - Soit en spécifiant un nouveau type de modèle (DSM, MOF)
  - Soit en utilisant UML ou une extension d'UML via la notion de **profil**
  - Sérialisation des modèles (XMI)
- Des techniques de vérification de modèles (OCL)
- Des techniques de transformation (QVT) :
  - De modèle à modèle (PIM vers PSM)
  - De modèle vers texte (ex : UML vers SQL)

Construire un nouveau type de modèle

- Pourquoi ?

Construire un nouveau type de modèle

- Pourquoi ?
  - Adapté à un domaine particulier, facilité d'utilisation, efficacité.

Construire un nouveau type de modèle

- Pourquoi ?
  - Adapté à un domaine particulier, facilité d'utilisation, efficacité.
  - Des exemples : diagramme de classes, diagramme de cas d'utilisation, diagramme relationnel, diagramme de graphes orientés, diagramme de composants. . .

Construire un nouveau type de modèle

- Pourquoi ?
  - Adapté à un domaine particulier, facilité d'utilisation, efficacité.
  - Des exemples : diagramme de classes, diagramme de cas d'utilisation, diagramme relationnel, diagramme de graphes orientés, diagramme de composants. . .
- Comment ?

Construire un nouveau type de modèle

- Pourquoi ?
  - Adapté à un domaine particulier, facilité d'utilisation, efficacité.
  - Des exemples : diagramme de classes, diagramme de cas d'utilisation, diagramme relationnel, diagramme de graphes orientés, diagramme de composants. . .
- Comment ?
  - Par la définition d'un **métamodèle**.

Construire un nouveau type de modèle

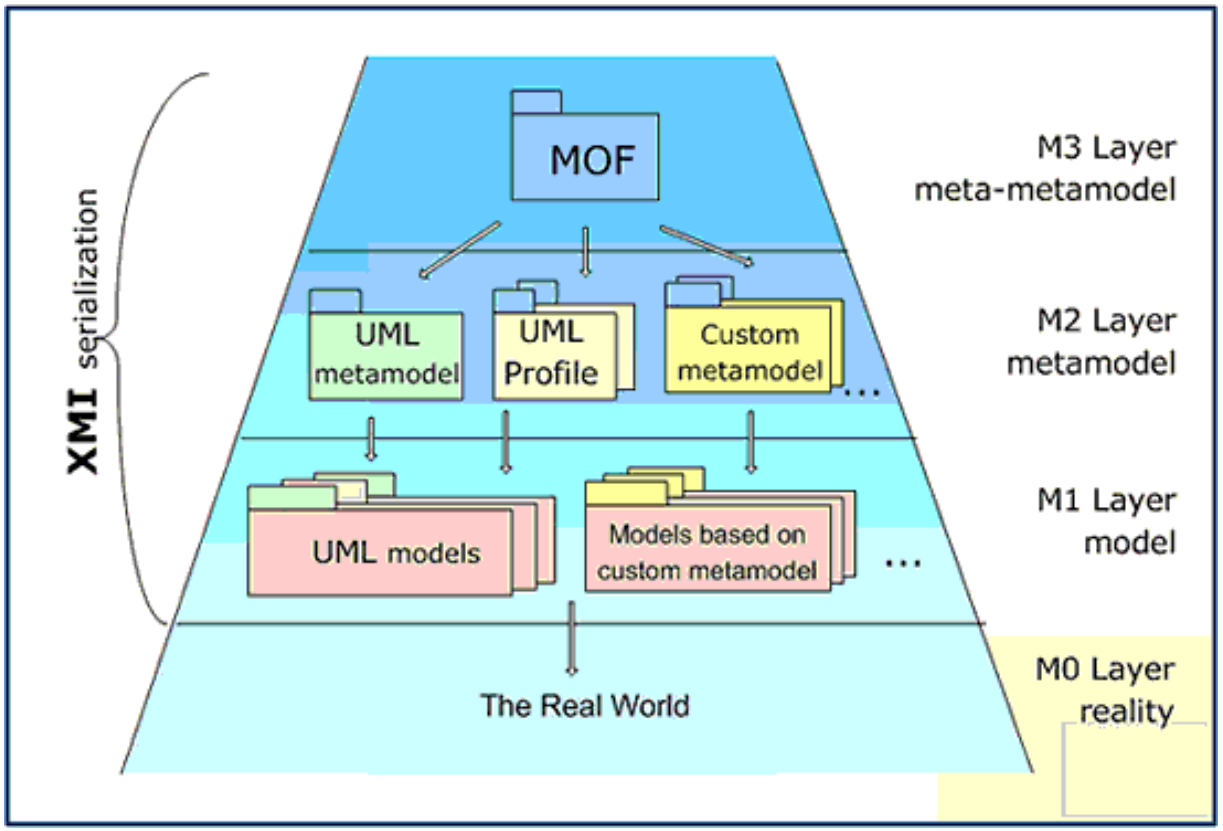
- Pourquoi ?
  - Adapté à un domaine particulier, facilité d'utilisation, efficacité.
  - Des exemples : diagramme de classes, diagramme de cas d'utilisation, diagramme relationnel, diagramme de graphes orientés, diagramme de composants. . .
- Comment ?
  - Par la définition d'un **métamodèle**.
  - Un modèle est conforme à un métamodèle

Construire un nouveau type de modèle

- Pourquoi ?
  - Adapté à un domaine particulier, facilité d'utilisation, efficacité.
  - Des exemples : diagramme de classes, diagramme de cas d'utilisation, diagramme relationnel, diagramme de graphes orientés, diagramme de composants. . .
- Comment ?
  - Par la définition d'un **métamodèle**.
  - Un modèle est conforme à un métamodèle
  - Un métamodèle est un modèle

L'espace des modèles

- L'architecture MDE (D. Djuric, JOT 2006)



Solution OMG pour les métamodèles

- MOF : Acronyme pour Meta-Object Facility (OMG)

Solution OMG pour les métamodèles

- MOF : Acronyme pour Meta-Object Facility (OMG)
- Différentes variantes du MOF : eMOF, cMOF, ...

Solution OMG pour les métamodèles

- MOF : Acronyme pour Meta-Object Facility (OMG)
- Différentes variantes du MOF : eMOF, cMOF, ...
- XMI : langage XML

Solution OMG pour les métamodèles

- MOF : Acronyme pour Meta-Object Facility (OMG)
- Différentes variantes du MOF : eMOF, cMOF, ...
- XMI : langage XML
  - Standard OMG

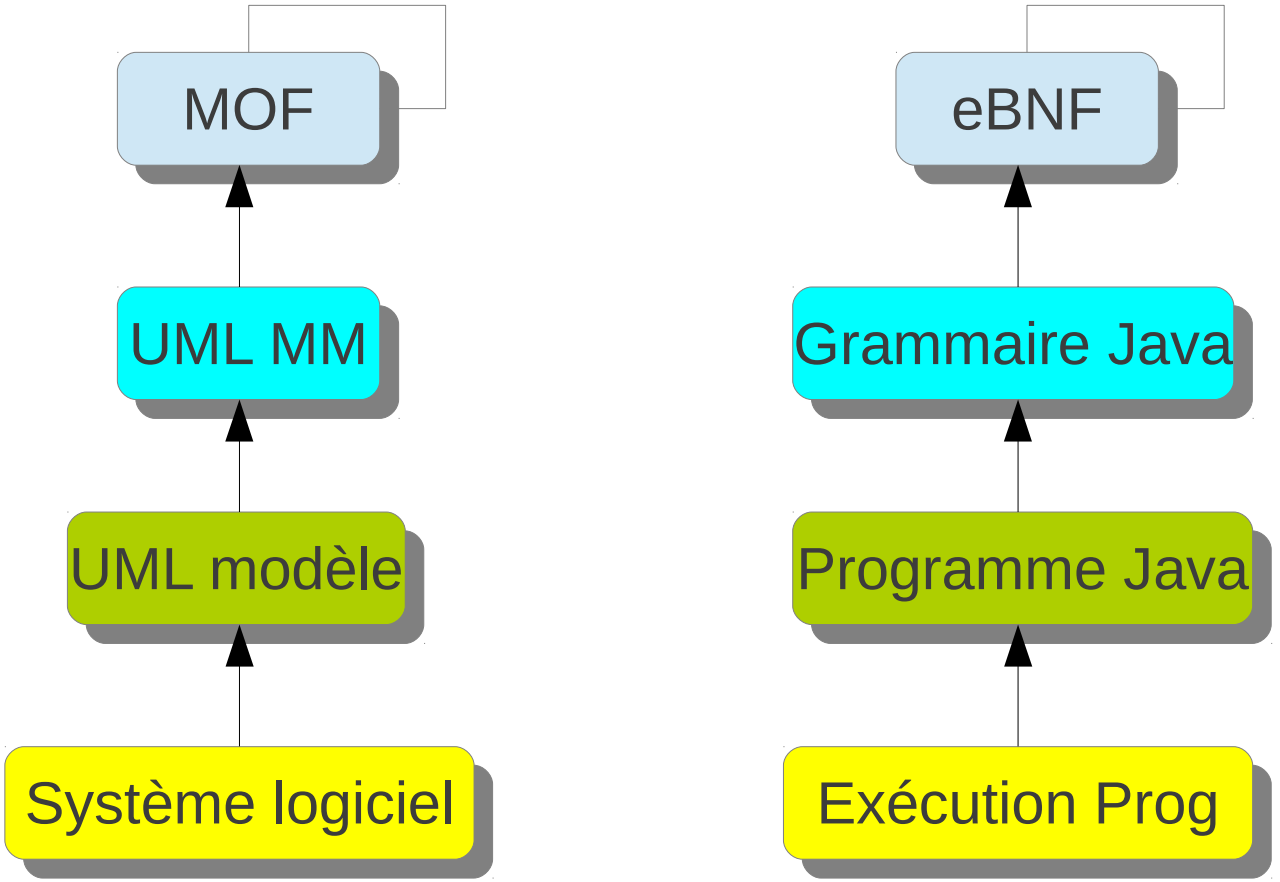
Solution OMG pour les métamodèles

- MOF : Acronyme pour Meta-Object Facility (OMG)
- Différentes variantes du MOF : eMOF, cMOF, ...
- XMI : langage XML
  - Standard OMG
  - Permet de sérialiser des modèles et métamodèles

Solution OMG pour les métamodèles

- MOF : Acronyme pour Meta-Object Facility (OMG)
- Différentes variantes du MOF : eMOF, cMOF, ...
- XMI : langage XML
  - Standard OMG
  - Permet de sérialiser des modèles et métamodèles
  - Échange de modèles et métamodèles

Modèles Vs Langages (S.Mosser, F. Chauvel)



Eclipse et la métamodélisation

- Une version Eclipse ”Modeling Tools”

Eclipse et la métamodélisation

- Une version Eclipse ”Modeling Tools”
- Le plugin essentiel EMF (Eclipse Modeling Framework) :

Eclipse et la métamodélisation

- Une version Eclipse ”Modeling Tools”
- Le plugin essentiel EMF (Eclipse Modeling Framework) :
  - Contient un métamétamodèle (Ecore), noyau réflexif simple (s’appuie sur eMOF)

Eclipse et la métamodélisation

- Une version Eclipse ”Modeling Tools”
- Le plugin essentiel EMF (Eclipse Modeling Framework) :
  - Contient un métamétamodèle (Ecore), noyau réflexif simple (s’appuie sur eMOF)
  - Construction de méta-modèles : édition graphique ou édition arborescente ou édition via code java annoté,

Eclipse et la métamodélisation

- Une version Eclipse ”Modeling Tools”
- Le plugin essentiel EMF (Eclipse Modeling Framework) :
  - Contient un métamétamodèle (Ecore), noyau réflexif simple (s’appuie sur eMOF)
  - Construction de méta-modèles : édition graphique ou édition arborescente ou édition via code java annoté,
  - Génération d’une API Java permettant la manipulation d’instances de ce méta-modèle

Eclipse et la métamodélisation

- Une version Eclipse ”Modeling Tools”
- Le plugin essentiel EMF (Eclipse Modeling Framework) :
  - Contient un métamétamodèle (Ecore), noyau réflexif simple (s’appuie sur eMOF)
  - Construction de méta-modèles : édition graphique ou édition arborescente ou édition via code java annoté,
  - Génération d’une API Java permettant la manipulation d’instances de ce méta-modèle
  - Génération d’un éditeur arborescent

Eclipse et la métamodélisation

- Une version Eclipse ”Modeling Tools”
- Le plugin essentiel EMF (Eclipse Modeling Framework) :
  - Contient un métamétamodèle (Ecore), noyau réflexif simple (s’appuie sur eMOF)
  - Construction de méta-modèles : édition graphique ou édition arborescente ou édition via code java annoté,
  - Génération d’une API Java permettant la manipulation d’instances de ce méta-modèle
  - Génération d’un éditeur arborescent
  - Intègre un framework de vérifications de contraintes (soit OCL, soit Java)

Eclipse et la métamodélisation

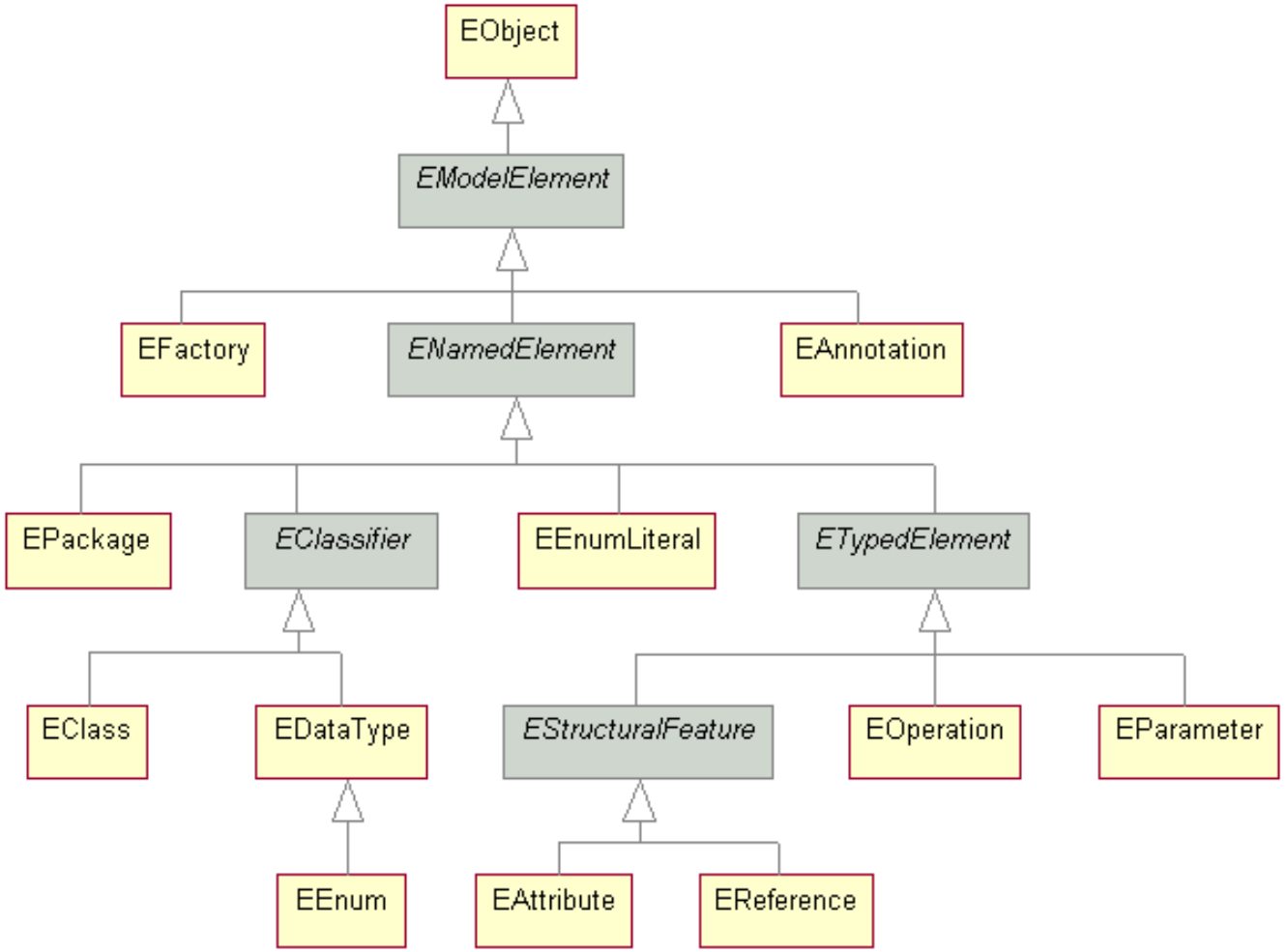
- Une version Eclipse ”Modeling Tools”
- Le plugin essentiel EMF (Eclipse Modeling Framework) :
  - Contient un métamétamodèle (Ecore), noyau réflexif simple (s’appuie sur eMOF)
  - Construction de méta-modèles : édition graphique ou édition arborescente ou édition via code java annoté,
  - Génération d’une API Java permettant la manipulation d’instances de ce méta-modèle
  - Génération d’un éditeur arborescent
  - Intègre un framework de vérifications de contraintes (soit OCL, soit Java)
  - Production de code sous forme de plugins Eclipse

Eclipse et la métamodélisation

- Une version Eclipse ”Modeling Tools”
- Le plugin essentiel EMF (Eclipse Modeling Framework) :
  - Contient un métamétamodèle (Ecore), noyau réflexif simple (s’appuie sur eMOF)
  - Construction de méta-modèles : édition graphique ou édition arborescente ou édition via code java annoté,
  - Génération d’une API Java permettant la manipulation d’instances de ce méta-modèle
  - Génération d’un éditeur arborescent
  - Intègre un framework de vérifications de contraintes (soit OCL, soit Java)
  - Production de code sous forme de plugins Eclipse
- et bien d’autres plugins associés (OCL, GMF, Acceleo, . . .)

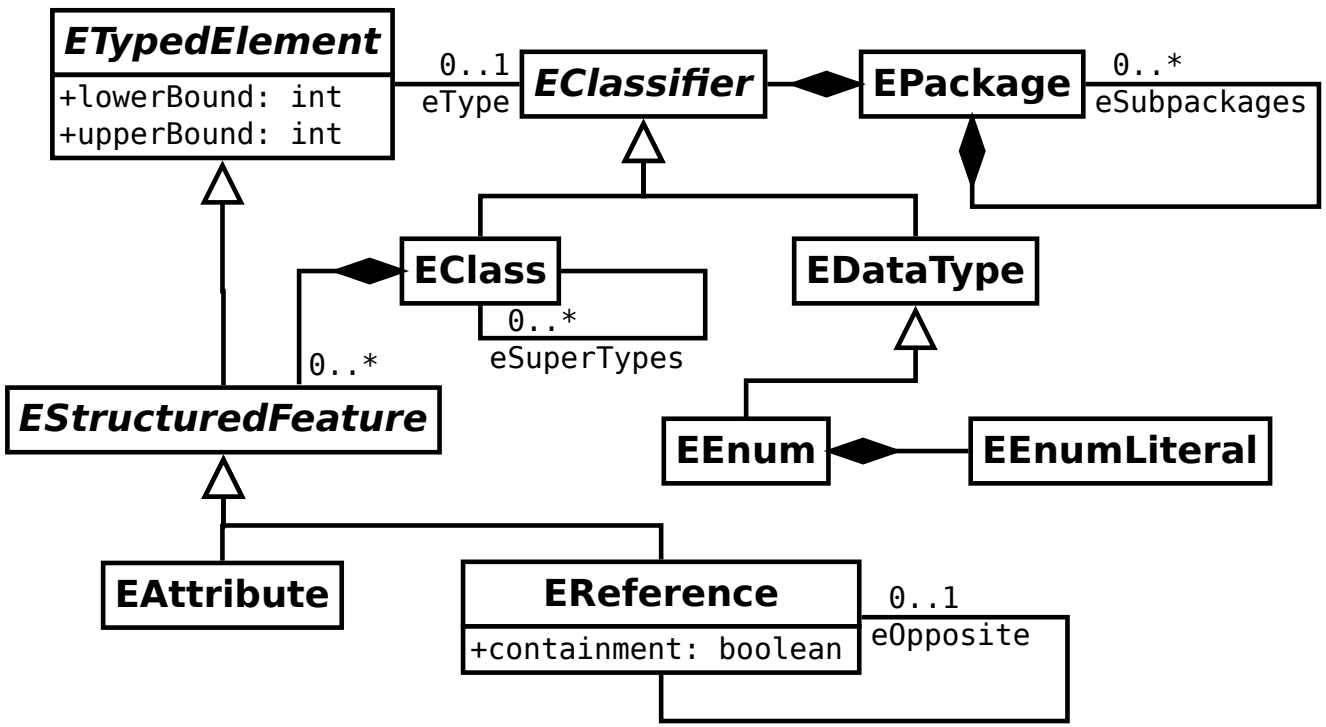
Le métamétamodèle Ecore (1/2)

- Hiérarchie Ecore (E. Cariou, Univ. Pau)



Le métamétamodèle Ecore (2/2)

- Vue simplifiée Ecore

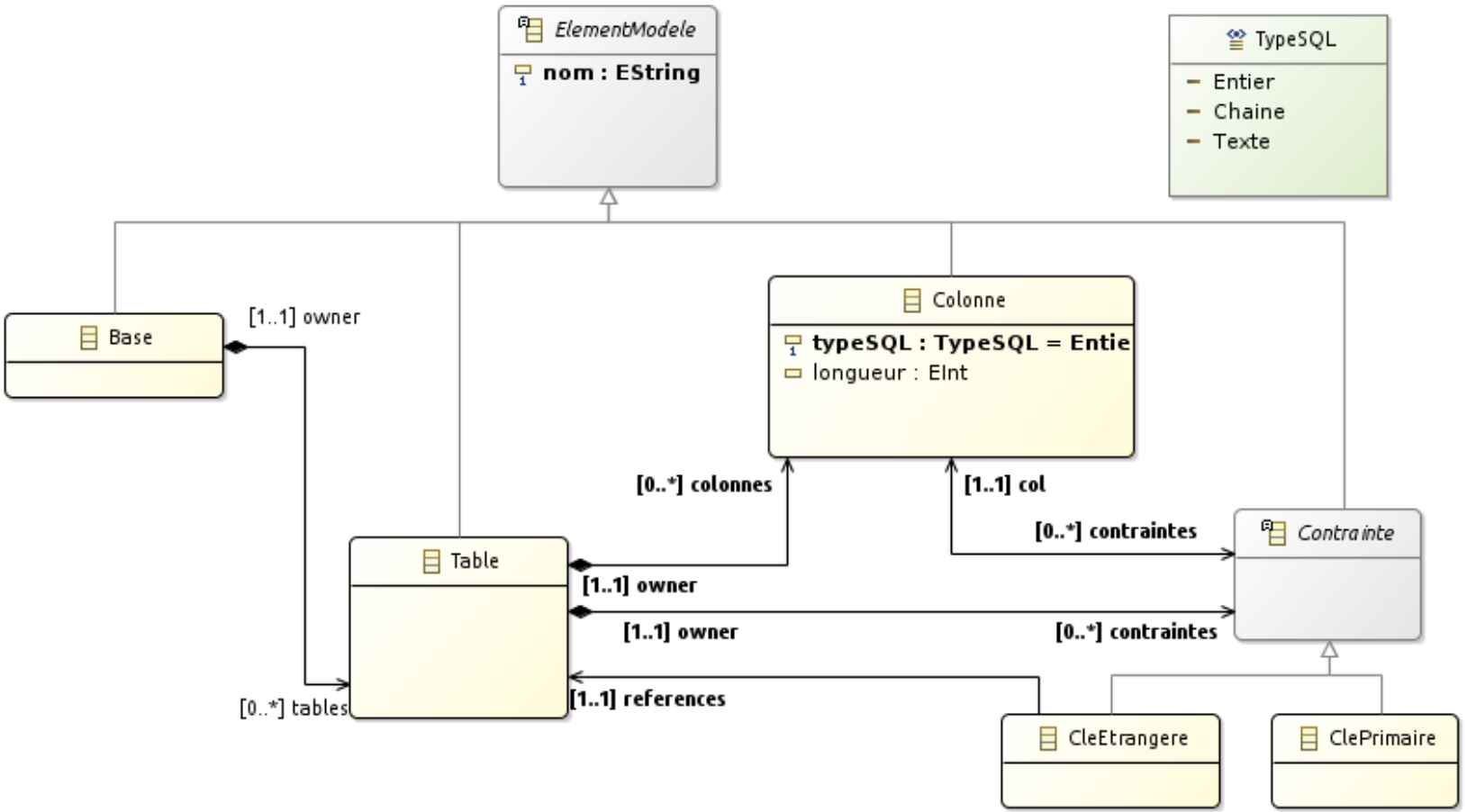


Un exemple fil rouge

Le modèle relationnel (sujet TP 2008-2009)

Une base de données relationnelle contient un ensemble de tables. Une table comporte plusieurs colonnes. Bases, tables et colonnes se caractérisent par un nom. Une colonne correspond à un type SQL particulier. Pour simplifier, on ne s'intéresse qu'aux types SQL Integer, Text et Varchar(n). Une table contient des contraintes nommées qui sont de deux types : Primary Key et Foreign Key. Pour simplifier, une clé primaire n'est formée que d'une seule colonne.

Un métamodèle possible (schéma "UML-like")



Particularités de construction d'un métamodèle

- Possibilité d'utiliser un éditeur graphique *ressemblant* à UML

Particularités de construction d'un métamodèle

- Possibilité d'utiliser un éditeur graphique *ressemblant* à UML
- Mais, un métamodèle n'est pas un modèle UML :

Particularités de construction d'un métamodèle

- Possibilité d'utiliser un éditeur graphique *ressemblant* à UML
- Mais, un métamodèle n'est pas un modèle UML :
  - Il n'y a **pas** d'associations (ni de propriétés d'association), mais des références (`EReference`)
    $\longrightarrow$ voir diapositive suivante

Particularités de construction d'un métamodèle

- Possibilité d'utiliser un éditeur graphique *ressemblant* à UML
- Mais, un métamodèle n'est pas un modèle UML :
  - Il n'y a **pas** d'associations (ni de propriétés d'association), mais des références (`EReference`)
    $\longrightarrow$ voir diapositive suivante
  - MOF vs UML : très peu de concepts (pas d'interfaces, etc)

Particularités de construction d'un métamodèle

- Possibilité d'utiliser un éditeur graphique *ressemblant* à UML
- Mais, un métamodèle n'est pas un modèle UML :
  - Il n'y a **pas** d'associations (ni de propriétés d'association), mais des références (`EReference`)
    $\longrightarrow$ voir diapositive suivante
  - MOF vs UML : très peu de concepts (pas d'interfaces, etc)
- Importance de la référence de type `containment` (équivalent composition UML) pour la création des objets :

Particularités de construction d'un métamodèle

- Possibilité d'utiliser un éditeur graphique *ressemblant* à UML
- Mais, un métamodèle n'est pas un modèle UML :
  - Il n'y a **pas** d'associations (ni de propriétés d'association), mais des références (`EReference`)
    $\longrightarrow$ voir diapositive suivante
  - MOF vs UML : très peu de concepts (pas d'interfaces, etc)
- Importance de la référence de type `containment` (équivalent composition UML) pour la création des objets :
  - Sérialisation arborescente du modèle :

Particularités de construction d'un métamodèle

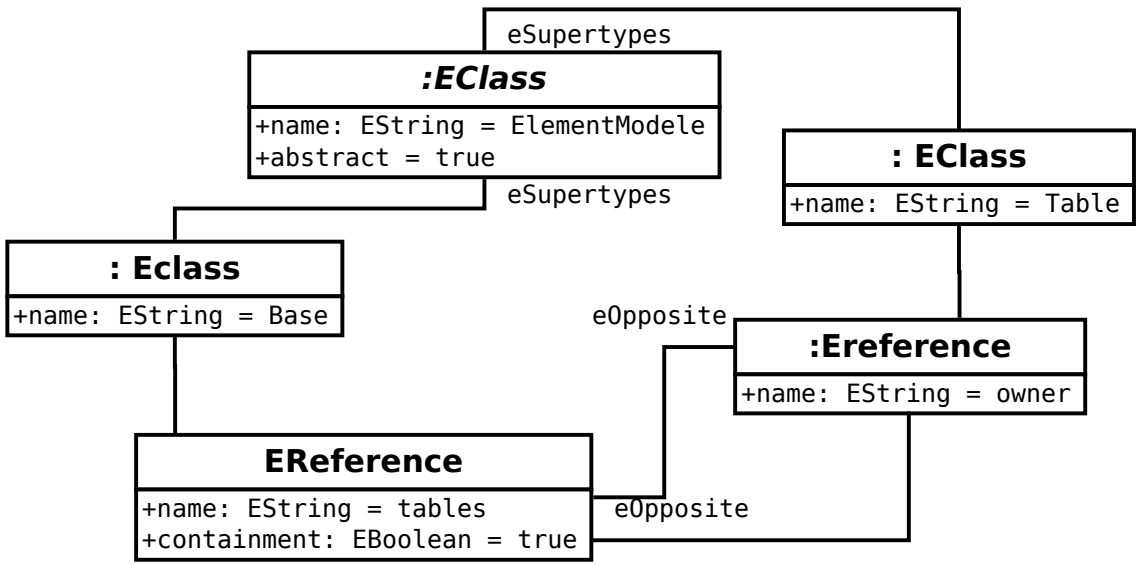
- Possibilité d'utiliser un éditeur graphique *ressemblant* à UML
- Mais, un métamodèle n'est pas un modèle UML :
  - Il n'y a **pas** d'associations (ni de propriétés d'association), mais des références (`EReference`)
    $\longrightarrow$ voir diapositive suivante
  - MOF vs UML : très peu de concepts (pas d'interfaces, etc)
- Importance de la référence de type `containment` (équivalent composition UML) pour la création des objets :
  - Sérialisation arborescente du modèle :
    - Importance de la classe racine

Particularités de construction d'un métamodèle

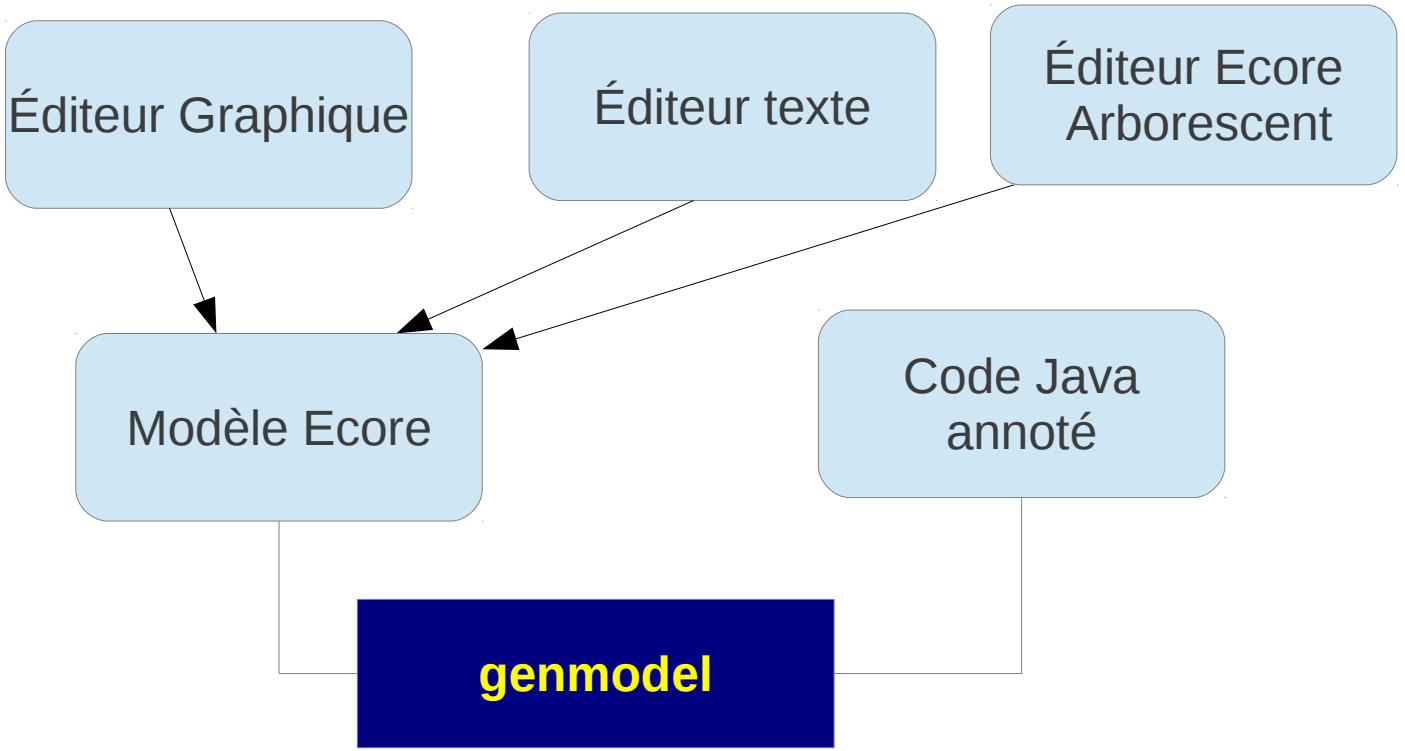
- Possibilité d'utiliser un éditeur graphique *ressemblant* à UML
- Mais, un métamodèle n'est pas un modèle UML :
  - Il n'y a **pas** d'associations (ni de propriétés d'association), mais des références (`EReference`)
    $\longrightarrow$ voir diapositive suivante
  - MOF vs UML : très peu de concepts (pas d'interfaces, etc)
- Importance de la référence de type `containment` (équivalent composition UML) pour la création des objets :
  - Sérialisation arborescente du modèle :
    - Importance de la classe racine
    - Lien `containment` $\equiv$ création d'un fils dans l'arborescence

Extrait du diagramme objet

- Association Versus EReference :



Construction de métamodèle sous Eclipse



Annotations Java

- Annotations Javadoc-like (@model)
- Définition d'interfaces Java, exemple :

```
package DB;

import org.eclipse.emf.common.util.EList;
/**
 * @model
 */
public interface Base extends ElementModele{
  /**
   * @model containment="true" opposite="owner"
   */
  EList<Table> getTables() ;
}
```

Code Java généré

- Une classe pour chaque métaclasse,

Code Java généré

- Une classe pour chaque métaclasse,
- Génération de fonctions accesseurs pour attributs et références.

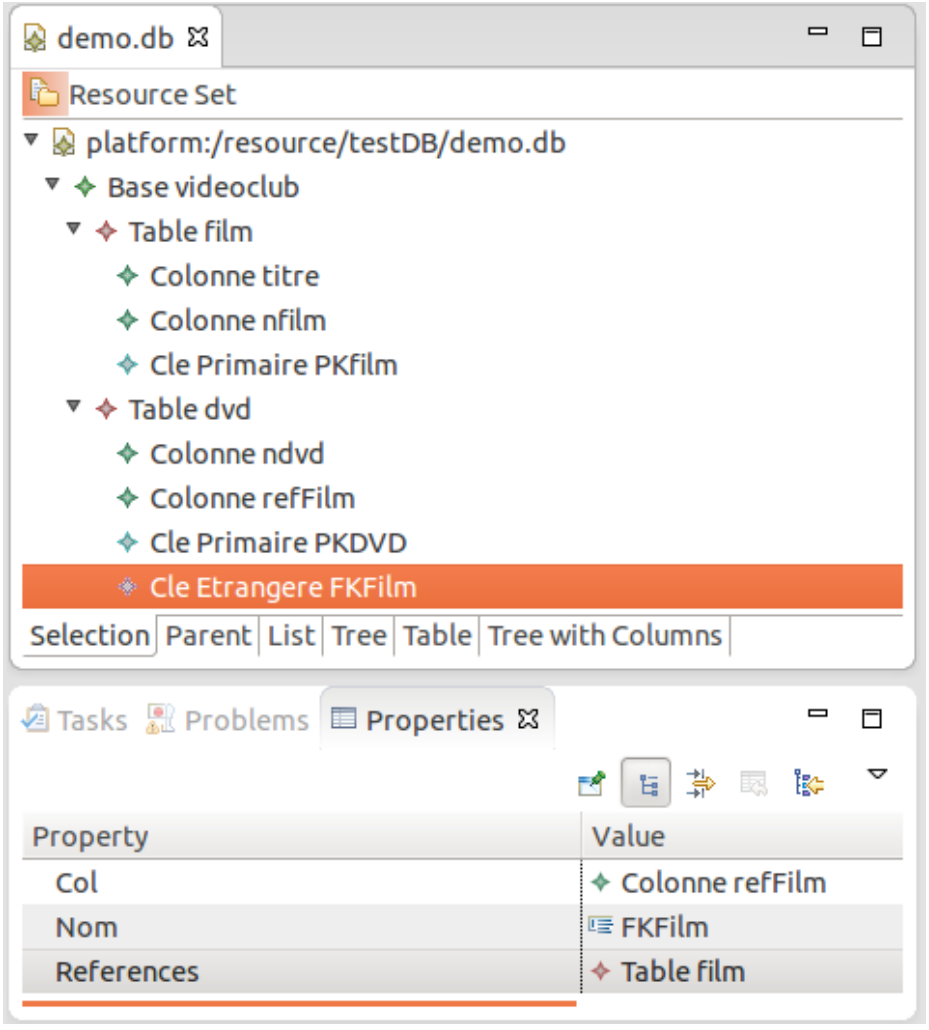
Code Java généré

- Une classe pour chaque métaclasse,
- Génération de fonctions accesseurs pour attributs et références.
- Gestion des contraintes d'intégrité référentielles
  (application du DP Observateur)

Code Java généré

- Une classe pour chaque métaclasse,
- Génération de fonctions accesseurs pour attributs et références.
- Gestion des contraintes d'intégrité référentielles
  (application du DP Observateur)
- Gestion de l'instanciation des objets
  (application du DP Fabrique)

Editeur arborescent généré



Programmation Java

- Illustration des Design Patterns :

```
DBFactory fabrique = DBFactory.eINSTANCE ;
Base videoclub = fabrique.createBase() ;
videoclub.setNom("videoclub");
Table film = fabrique.createTable() ;
film.setNom("film"); film.setOwner(videoclub);
Table dvd = fabrique.createTable() ;
dvd.setNom("dvd"); dvd.setOwner(videoclub);
for (Table table : videoclub.getTables())
  System.out.print("Table␣:␣"+table.getNom()) ;
```

Sérialisation XMI en Java

- Eclipse, notion de `Ressource` :

```
DBPackage.eINSTANCE.eClass();
Resource.Factory.Registry reg = Resource.Factory.Registry.INSTANCE;
Map<String, Object> m = reg.getExtensionToFactoryMap();
m.put("db", new XMIResourceFactoryImpl());
ResourceSet resSet=new ResourceSetImpl();
Resource res = resSet.getResource(URI.createURI("./demo.db"),true);
Base base = (Base)res.getContents().get(0);
for (Table t : base.getTables()) {
  System.out.print("Table : "+t.getNom()) ;
  System.out.print(", cle primaire : ") ;
  for (Contrainte cte : t.getContraintes()) {
        if (cte instanceof ClePrimaire)
          System.out.println(cte.getCol().getNom()) ;
  }
}
```

Exemple de fichier XMI

```
<DB:Base xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:DB="http:///DB.ecore" nom="videoclub">
 <tables nom="film">
  <colonnes nom="titre" typeSQL="Chaine" longueur="20"/>
  <colonnes nom="nfilm" contraintes="//@tables.0/@contraintes.0"/>
  <contraintes xsi:type="DB:ClePrimaire" nom="PKfilm"
               col="//@tables.0/@colonnes.1"/>
 </tables>
 <tables nom="dvd">
   <colonnes nom="ndvd" contraintes="//@tables.1/@contraintes.0"/>
   <colonnes nom="refFilm" contraintes="//@tables.1/@contraintes.1"/>
   <contraintes xsi:type="DB:ClePrimaire" nom="PKDVD"
                col="//@tables.1/@colonnes.0"/>
   <contraintes xsi:type="DB:CleEtrangere" nom="FKFilm"
                col="//@tables.1/@colonnes.1" references="//@tables.0"/>
 </tables>
</DB:Base>
```

Validité des modèles

Constat

Validité des modèles

Constat

- Un modèle conforme à un métamodèle décrit par Ecore peut être non valide

Validité des modèles

Constat

- Un modèle conforme à un métamodèle décrit par Ecore peut être non valide
- Le pouvoir d'expression des métamodèles est insuffisant.

Validité des modèles

Constat

- Un modèle conforme à un métamodèle décrit par Ecore peut être non valide
- Le pouvoir d'expression des métamodèles est insuffisant.

Solution

Compléter le métamodèle par des règles de cohérence.

Validité des modèles

Constat

- Un modèle conforme à un métamodèle décrit par Ecore peut être non valide
- Le pouvoir d'expression des métamodèles est insuffisant.

Solution

Compléter le métamodèle par des règles de cohérence.

Illustration UML

UML est spécifié à l'aide de son métamodèle et de règles écrites en OCL.

Les incohérences possibles de l'exemple

Contraintes modèle relationnel

Les incohérences possibles de l'exemple

Contraintes modèle relationnel

- La base, les tables, les colonnes, les contraintes doivent avoir un nom fixé (déjà géré : `lowerBound=1`)

Les incohérences possibles de l'exemple

Contraintes modèle relationnel

- La base, les tables, les colonnes, les contraintes doivent avoir un nom fixé (déjà géré : `lowerBound=1`)
- Les tables d'une même base ont un nom différent

Les incohérences possibles de l'exemple

Contraintes modèle relationnel

- La base, les tables, les colonnes, les contraintes doivent avoir un nom fixé (déjà géré : `lowerBound=1`)
- Les tables d'une même base ont un nom différent
- Les colonnes d'une même table ont un nom différent

Les incohérences possibles de l'exemple

Contraintes modèle relationnel

- La base, les tables, les colonnes, les contraintes doivent avoir un nom fixé (déjà géré : `lowerBound=1`)
- Les tables d'une même base ont un nom différent
- Les colonnes d'une même table ont un nom différent
- Une table doit posséder une clé primaire et une seule

Les incohérences possibles de l'exemple

Contraintes modèle relationnel

- La base, les tables, les colonnes, les contraintes doivent avoir un nom fixé (déjà géré : `lowerBound=1`)
- Les tables d'une même base ont un nom différent
- Les colonnes d'une même table ont un nom différent
- Une table doit posséder une clé primaire et une seule
- Une clé étrangère doit référencer une table (déjà géré)

Les incohérences possibles de l'exemple

Contraintes modèle relationnel

- La base, les tables, les colonnes, les contraintes doivent avoir un nom fixé (déjà géré : `lowerBound=1`)
- Les tables d'une même base ont un nom différent
- Les colonnes d'une même table ont un nom différent
- Une table doit posséder une clé primaire et une seule
- Une clé étrangère doit référencer une table (déjà géré)
- La clé primaire d'une table doit référencer une colonne de la même table

Les incohérences possibles de l'exemple

Contraintes modèle relationnel

- La base, les tables, les colonnes, les contraintes doivent avoir un nom fixé (déjà géré : `lowerBound=1`)
- Les tables d'une même base ont un nom différent
- Les colonnes d'une même table ont un nom différent
- Une table doit posséder une clé primaire et une seule
- Une clé étrangère doit référencer une table (déjà géré)
- La clé primaire d'une table doit référencer une colonne de la même table
- La colonne d'une clé étrangère doit être du même type que la clé primaire de la table référencée

Le langage OCL

- Acronyme pour **O**bject **C**onstraint **Language**

Le langage OCL

- Acronyme pour **O**bject **C**onstraint **Language**
- Langage créé par IBM à l'origine dédié à la spécification de modèles UML

Le langage OCL

- Acronyme pour **O**bject **C**onstraint **Language**
- Langage créé par IBM à l'origine dédié à la spécification de modèles UML
- Langage formel simple (pas d'ambigüités par rapport au langage naturel)

Le langage OCL

- Acronyme pour **O**bject **C**onstraint **Language**
- Langage créé par IBM à l'origine dédié à la spécification de modèles UML
- Langage formel simple (pas d'ambigüités par rapport au langage naturel)
- Langage généralisé à l'expression de tout type de modèles (ex : Ecore)

Le langage OCL

- Acronyme pour **O**bject **C**onstraint **Language**
- Langage créé par IBM à l'origine dédié à la spécification de modèles UML
- Langage formel simple (pas d'ambigüités par rapport au langage naturel)
- Langage généralisé à l'expression de tout type de modèles (ex : Ecore)
- Utilisable à tout niveau (modèle, métamodèle)

Les contraintes OCL

- Une contrainte retourne une valeur booléenne

Les contraintes OCL

- Une contrainte retourne une valeur booléenne
- Permet de définir des **invariants** :

Les contraintes OCL

- Une contrainte retourne une valeur booléenne
- Permet de définir des **invariants** :
  - est définie dans le contexte d'une classe

Les contraintes OCL

- Une contrainte retourne une valeur booléenne
- Permet de définir des **invariants** :
  - est définie dans le contexte d'une classe
  - s'applique sur un objet, instance de cette classe (variable `self`)

Les contraintes OCL

- Une contrainte retourne une valeur booléenne
- Permet de définir des **invariants** :
  - est définie dans le contexte d'une classe
  - s'applique sur un objet, instance de cette classe (variable `self`)
- Permet de définir des **pré ou post conditions** :

Les contraintes OCL

- Une contrainte retourne une valeur booléenne
- Permet de définir des **invariants** :
  - est définie dans le contexte d'une classe
  - s'applique sur un objet, instance de cette classe (variable `self`)
- Permet de définir des **pré ou post conditions** :
  - est définie dans le contexte d'une opération

Les contraintes OCL

- Une contrainte retourne une valeur booléenne
- Permet de définir des **invariants** :
  - est définie dans le contexte d'une classe
  - s'applique sur un objet, instance de cette classe (variable `self`)
- Permet de définir des **pré ou post conditions** :
  - est définie dans le contexte d'une opération
  - s'exécute avant et après l'exécution de l'opération

Les contraintes OCL

- Une contrainte retourne une valeur booléenne
- Permet de définir des **invariants** :
  - est définie dans le contexte d'une classe
  - s'applique sur un objet, instance de cette classe (variable `self`)
- Permet de définir des **pré ou post conditions** :
  - est définie dans le contexte d'une opération
  - s'exécute avant et après l'exécution de l'opération
  - programmation par contrats

Les types primitifs d'OCL

- Les types de base sont : Integer, Real, Boolean et String (comme UML)

Les types primitifs d'OCL

- Les types de base sont : Integer, Real, Boolean et String (comme UML)
- Les opérateurs qui s'appliquent sur ces types :

Opérateurs relationnels	$=, <>, >, >=, <=$
Opérateurs logiques	and, or, xor, not
Opérateurs mathématiques	$+, -, /, *, =$, min, max,...
Opérateurs chaînes de caractères	concat, toUpper, substring,...

Navigation dans le modèle

- Parcours via les rôles (`EReference`)
- Accès aux ”feature” (attributs, opérations)
- Cardinalité 1, opérateur ’.’
- Cardinalité ’*’, opérateur ’$\rightarrow$’

Un exemple

```
context Colonne inv :
  self.owner.owner.nom <> null
```

Opérations booléennes et dénombrement sur les collections

Nom	**Signification**
Opérations booléennes	
includes(object :T) :Boolean	$\exists$ de `object`
excludes(object :T) :Boolean	non $\exists$ de `object`
includesAll(c :Collection(T)) :Boolean	$\exists$ de tous les éléments de `c`
excludesAll(c :Collection(T)) :Boolean	non $\exists$ de tous les éléments de `c`
isEmpty() :Boolean	liste vide
notEmpty() :Boolean	liste non vide
Opérations de dénombrement	
size() :Integer	nbre d'éléments
count(object : T) :Integer	nbre d'occurences de `object`

Opérations sur les metatypes

Nom	Signification
oclIsTypeOf(t :OclType) :Boolean	vrai si types identiques
oclIsKindOf(t :OclType) :Boolean	vrai si types identiques ou si t est un super-type
oclAsTypeOf(t :OclType) :T	cast

Opérateur select

- Permet de sélectionner des instances d'une collection en fonction d'une expression booléenne.
- Syntaxes :
  - `collection->select(elem : T| expr)`
  - `collection->select(elem | expr)`
  - `collection->select(elem | expr)`

Un exemple pour une instance de `Base`

```
self.tables->select(t | t.contraintes->notEmpty())
```

retourne une collection des tables film et dvd

Opérateur exists

- Retourne vrai si l'expression est vraie pour au moins un élément de la collection.
- Syntaxes :
  - `collection->exists(elem : T| expr)`
  - `collection->exists(elem | expr)`
  - `collection->exists(elem | expr)`

Un exemple pour une instance de `Base`

```
self.tables->exists(nom='film')
```

Opérateur forAll

- Retourne vrai si l'expression est vraie pour tous les éléments de la collection.
- Syntaxes :
  - `collection->forAll(elem : T| expr)`
  - `collection->forAll(elem | expr)`
  - `collection->forAll(elem | expr)`

Un exemple pour une instance de `Base`

```
self.tables->forAll(t1, t2 | t1.nom=t2.nom implies t1=t2)
```

Opérateur collect

- Retourne la collection des valeurs (Bag) résultant de l'évaluation de l'expression appliquée à tous les éléments de la collection
- Syntaxes :
  - `collection->collect(elem : T| expr)`
  - `collection->collect(elem | expr)`
  - `collection->collect(elem | expr)`

Un exemple pour une instance de `Base`

```
self.tables->collect(contraintes)

// retourne les 3 contraintes PKfilm, PKDVD et FKFilm
```

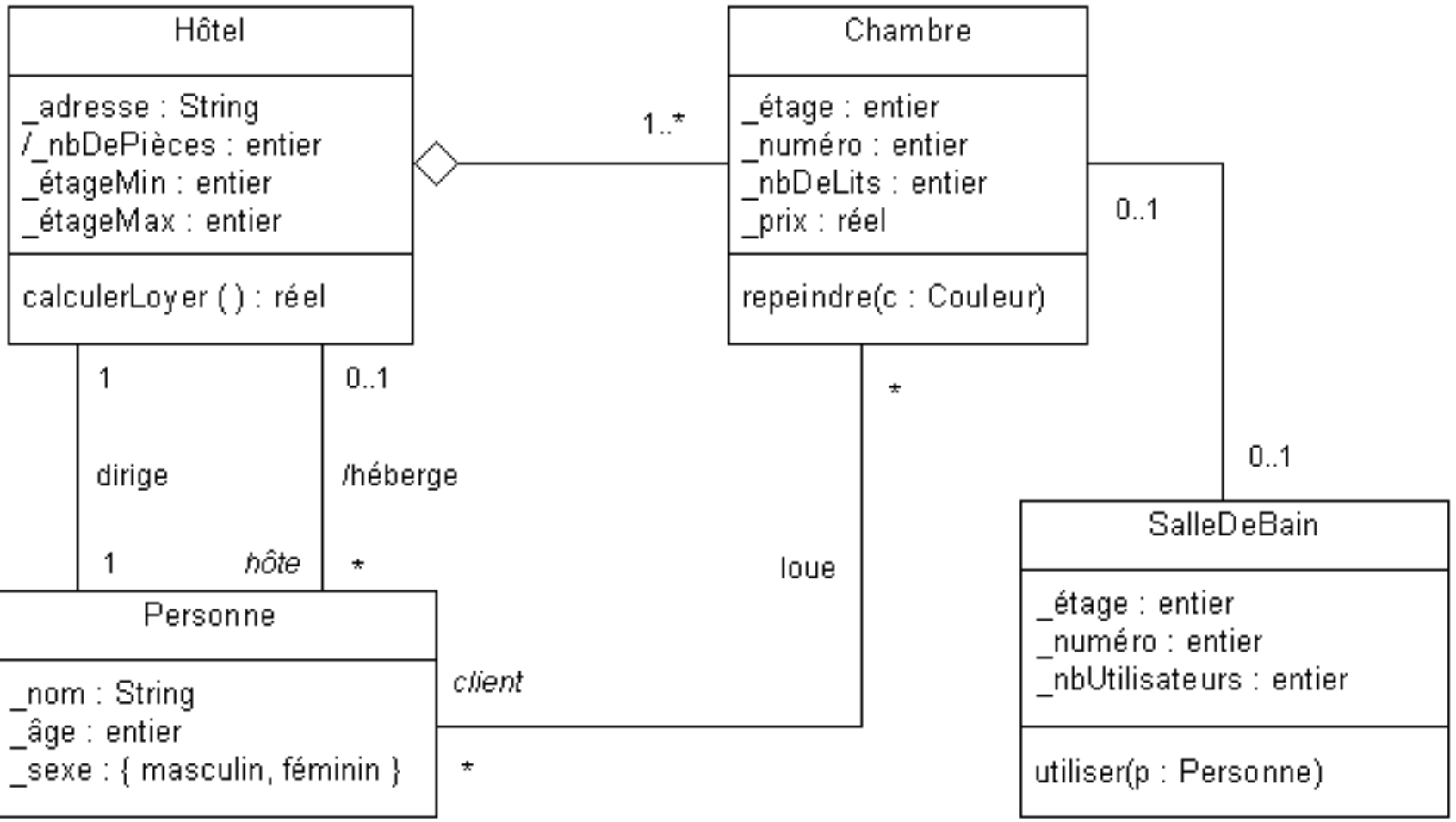
Exemple modèle relationnel

- Le type de la clé étrangère est conforme au type de la clé primaire qu'elle référence
- La vérification de l'existence d'une clé primaire est réalisée dans une autre contrainte

Contrôle des types de clés

```
Context CleEtrangere inv:
  self.references.contraintes
  ->select(oclIsKindOf(ClePrimaire))
  ->first().col.typeSQL=self.col.typeSQL
```

Exemple OCL, source JOOP 1999, uml.free.fr



Contraintes OCL (1/3)

* Un hôtel ne contient jamais d'étage numéro 13 (superstition oblige).

```
context Chambre inv:
  self._etage <> 13
```

* Le nombre de personnes par chambre doit être inférieur ou égal au nombre de lits dans la chambre louée. Les enfants (accompagnés) de moins de 4 ans ne "comptent pas" dans cette règle de calcul (à hauteur d'un enfant de moins de 4 ans maximum par chambre).

```
context Chambre inv:
  client->size <= _nbDeLits or
  (client->size = _nbDeLits + 1 and
                  client->exists(p : Personne | p._age < 4))
```

Contraintes OCL (2/3)

* L'étage de chaque chambre est compris entre le premier et le dernier étage de l'hôtel.

```
context Hotel inv:
 self.chambre->forAll(c : Chambre | c._etage <= self._etageMax and
                                    c._etage >= self._etageMin)
```

* Chaque étage possède au moins une chambre (sauf l'étage 13, qui n'existe pas...).

```
context Hotel inv:
 Sequence{_etageMin.._etageMax}->forAll(i : Integer |
   if i <> 13 then
     self.chambre->select(c : Chambre | c._etage = i)->notEmpty)
   endif)
```

Contraintes OCL (3/3)

* On ne peut repeindre une chambre que si elle n'est pas louée. Une fois repeinte, une chambre coûte 10 pour cent de plus.

```
context Chambre::repeindre(c : Couleur)
  pre: client->isEmpty
  post: _prix = _prix@pre * 1.1
```

* Une salle de bain privative ne peut être utilisée que par les personnes qui louent la chambre contenant la salle de bain et une salle de bain sur le palier ne peut être utilisée que par les clients qui logent sur le même palier.

```
context SalleDeBain::utiliser(p : Personne)
pre: if chambre->notEmpty then chambre.client->includes(p)
     else p.chambre._etage = self._etage
     endif
post: _nbUtilisateurs = _nbUtilisateurs@pre + 1
```

Eclipse et les contraintes

- Le framework EMF intègre un moteur de contraintes

Eclipse et les contraintes

- Le framework EMF intègre un moteur de contraintes
- Deux possibilités d'implémentation des constraintes :

Eclipse et les contraintes

- Le framework EMF intègre un moteur de contraintes
- Deux possibilités d'implémentation des constraintes :
  - Contraintes écrites en Java : utilisation de l'API Java générée par EMF

Eclipse et les contraintes

- Le framework EMF intègre un moteur de contraintes
- Deux possibilités d'implémentation des constraintes :
  - Contraintes écrites en Java : utilisation de l'API Java générée par EMF
  - Utilisation d'OCL

Eclipse et les contraintes

- Le framework EMF intègre un moteur de contraintes
- Deux possibilités d'implémentation des constraintes :
  - Contraintes écrites en Java : utilisation de l'API Java générée par EMF
  - Utilisation d'OCL
- L'éditeur OCLinEcore :

Eclipse et les contraintes

- Le framework EMF intègre un moteur de contraintes
- Deux possibilités d'implémentation des constraintes :
  - Contraintes écrites en Java : utilisation de l'API Java générée par EMF
  - Utilisation d'OCL
- L'éditeur OCLinEcore :
  - Langage qui permet de saisir des modèles Ecore

Eclipse et les contraintes

- Le framework EMF intègre un moteur de contraintes
- Deux possibilités d'implémentation des constraintes :
  - Contraintes écrites en Java : utilisation de l'API Java générée par EMF
  - Utilisation d'OCL
- L'éditeur OCLinEcore :
  - Langage qui permet de saisir des modèles Ecore
  - Langage généré par le plugin XText

Eclipse et les contraintes

- Le framework EMF intègre un moteur de contraintes
- Deux possibilités d'implémentation des constraintes :
  - Contraintes écrites en Java : utilisation de l'API Java générée par EMF
  - Utilisation d'OCL
- L'éditeur OCLinEcore :
  - Langage qui permet de saisir des modèles Ecore
  - Langage généré par le plugin XText
  - Permet d'insérer du code OCL

Le métamodèle relationnel en mode texte

```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore#/';

package DB : DB = 'http:///DB.ecore'
{
  abstract class ElementModele {
        attribute nom : String;
  }
  class Base extends ElementModele {
        property tables#owner : Table[*] { ordered composes !resolve };
        invariant uniciteNomTables:
        self.tables->forAll(t1, t2 | t1.nom = t2.nom implies t1=t2) ;
  }
  ...
```

Avantages des métamodèles

- Parfaitement adapté au domaine métier

Avantages des métamodèles

- Parfaitement adapté au domaine métier
- Pas de concepts inutiles

Avantages des métamodèles

- Parfaitement adapté au domaine métier
- Pas de concepts inutiles
- Réduction des erreurs de conception

Inconvénients des métamodèles

- Etablir un métamodèle d'un domaine métier n'est pas un processus simple

Inconvénients des métamodèles

- Etablir un métamodèle d'un domaine métier n'est pas un processus simple
- Un métamodèle ne fournit pas un langage graphique (diagramme)

Inconvénients des métamodèles

- Etablir un métamodèle d'un domaine métier n'est pas un processus simple
- Un métamodèle ne fournit pas un langage graphique (diagramme)
  - Solutions Eclipse :

Inconvénients des métamodèles

- Etablir un métamodèle d'un domaine métier n'est pas un processus simple
- Un métamodèle ne fournit pas un langage graphique (diagramme)
  - Solutions Eclipse :
    - GMF : **G**raphic **M**odeling **F**ramework
      Un méta-outil pour associer un modèle graphique à un métamodèle EMF

Inconvénients des métamodèles

- Etablir un métamodèle d'un domaine métier n'est pas un processus simple
- Un métamodèle ne fournit pas un langage graphique (diagramme)
  - Solutions Eclipse :
    - GMF : **G**raphic **M**odeling **F**ramework
      Un méta-outil pour associer un modèle graphique à un métamodèle EMF
    - XText : un méta-outil pour associer un langage (grammaire) à un métamodèle EMF

Inconvénients des métamodèles

- Etablir un métamodèle d'un domaine métier n'est pas un processus simple
- Un métamodèle ne fournit pas un langage graphique (diagramme)
  - Solutions Eclipse :
    - GMF : **G**raphic **M**odeling **F**ramework
      Un méta-outil pour associer un modèle graphique à un métamodèle EMF
    - XText : un méta-outil pour associer un langage (grammaire) à un métamodèle EMF
  - Mais processus de construction encore moins simple

Inconvénients des métamodèles

- Etablir un métamodèle d'un domaine métier n'est pas un processus simple
- Un métamodèle ne fournit pas un langage graphique (diagramme)
  - Solutions Eclipse :
    - GMF : **G**raphic **M**odeling **F**ramework
      Un méta-outil pour associer un modèle graphique à un métamodèle EMF
    - XText : un méta-outil pour associer un langage (grammaire) à un métamodèle EMF
  - Mais processus de construction encore moins simple

La question innocente

Pourquoi ne pas utiliser UML ou toute autre notation graphique ?

Qu’est-ce qu’un profil UML ?

- Mécanisme `standard` d’extension d’UML

Qu'est-ce qu'un profil UML ?

- Mécanisme `standard` d'extension d'UML
- On n'ajoute pas de métaclasses mais des annotations aux métaclasses UML existantes (customisation).

Qu'est-ce qu'un profil UML ?

- Mécanisme `standard` d'extension d'UML
- On n'ajoute pas de métaclasses mais des annotations aux métaclasses UML existantes (customisation).

Un profil UML, c'est

Un ensemble cohérent de stéréotypes et contraintes qui sont appliqués à différentes métaclasses UML.

Avantages et inconvénients d'UML

- Avantages :

Avantages et inconvénients d'UML

- Avantages :
  - Les habitués UML s'y retrouvent

Avantages et inconvénients d'UML

- Avantages :
  - Les habitués UML s'y retrouvent
  - Pas besoin de définir une notation graphique, c'est celle d'UML

Avantages et inconvénients d'UML

- Avantages :
  - Les habitués UML s'y retrouvent
  - Pas besoin de définir une notation graphique, c'est celle d'UML
  - Les ateliers UML sont déjà compatibles

Avantages et inconvénients d'UML

- Avantages :
  - Les habitués UML s'y retrouvent
  - Pas besoin de définir une notation graphique, c'est celle d'UML
  - Les ateliers UML sont déjà compatibles
- Inconvénients :

Avantages et inconvénients d'UML

- Avantages :
  - Les habitués UML s'y retrouvent
  - Pas besoin de définir une notation graphique, c'est celle d'UML
  - Les ateliers UML sont déjà compatibles
- Inconvénients :
  - Nécessité de restreindre UML (via des contraintes)

Avantages et inconvénients d'UML

- Avantages :
  - Les habitués UML s'y retrouvent
  - Pas besoin de définir une notation graphique, c'est celle d'UML
  - Les ateliers UML sont déjà compatibles
- Inconvénients :
  - Nécessité de restreindre UML (via des contraintes)
  - Identifier chaque concept UML le plus proche de chaque concept métier

Avantages et inconvénients d'UML

- Avantages :
  - Les habitués UML s'y retrouvent
  - Pas besoin de définir une notation graphique, c'est celle d'UML
  - Les ateliers UML sont déjà compatibles
- Inconvénients :
  - Nécessité de restreindre UML (via des contraintes)
  - Identifier chaque concept UML le plus proche de chaque concept métier
  - Pas de possibilité de créer une nouvelle métaclasse

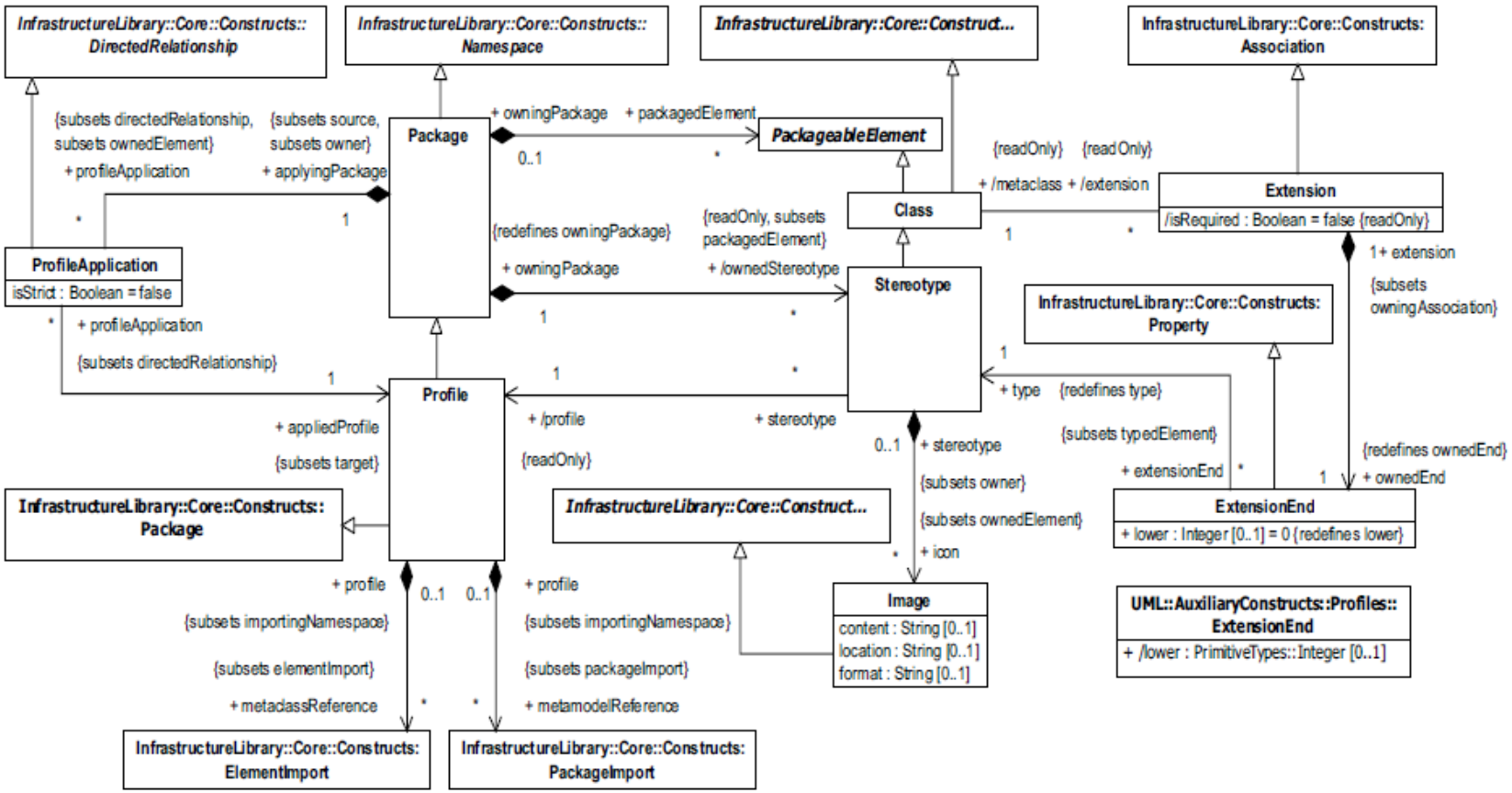
Avantages et inconvénients d'UML

- Avantages :
  - Les habitués UML s'y retrouvent
  - Pas besoin de définir une notation graphique, c'est celle d'UML
  - Les ateliers UML sont déjà compatibles
- Inconvénients :
  - Nécessité de restreindre UML (via des contraintes)
  - Identifier chaque concept UML le plus proche de chaque concept métier
  - Pas de possibilité de créer une nouvelle métaclasse
  - Restreindre les caractéristiques du concept sélectionné (via des contraintes)

Avantages et inconvénients d'UML

- Avantages :
  - Les habitués UML s'y retrouvent
  - Pas besoin de définir une notation graphique, c'est celle d'UML
  - Les ateliers UML sont déjà compatibles
- Inconvénients :
  - Nécessité de restreindre UML (via des contraintes)
  - Identifier chaque concept UML le plus proche de chaque concept métier
  - Pas de possibilité de créer une nouvelle métaclasse
  - Restreindre les caractéristiques du concept sélectionné (via des contraintes)
  - Au final, beaucoup plus de contraintes à spécifier

Mécanisme d'extension UML



Processus construction profil UML (1/3)

- Phase 1 : attacher les concepts du modèle cible au modèle UML

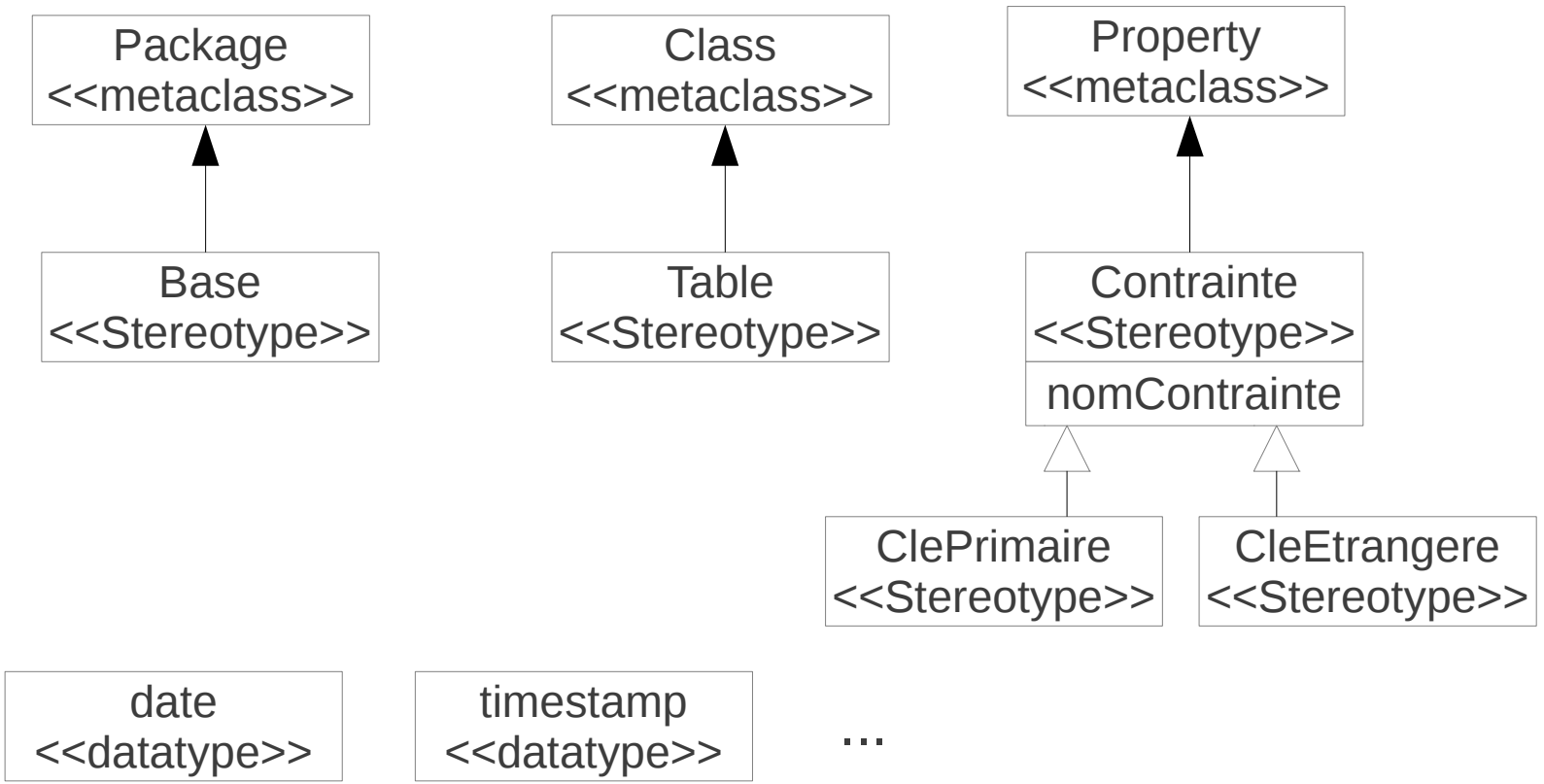
Processus construction profil UML (1/3)

- Phase 1 : attacher les concepts du modèle cible au modèle UML

Profil Bases de données

Concept modèle	Méta-classe UML	Extension
Base	Package	Stéréotype
Table	Class	Stéréotype
Colonne	Property	
ClePrimaire	Property	Stéréotype
CleEtrangere	Property	Stéréotype

Notation Graphique Profil UML



Processus construction profil UML (2/3)

- Phase 2 : restreindre le métamodèle UML
- Utilisation du langage OCL

Contrôle du contenu d'une base de données

```
Context Base inv:
  self.base_Package.allOwnedElements()->forAll( e |
    e.oclIsTypeOf(uml::Class)
  )
```

- Une classe ne contient que des attributs (`Property`), restriction des types d'attributs...
- parcours métamodèle : lien d'agrégation `base_`*`nomMetaClasse`*

Processus construction profil UML (3/3)

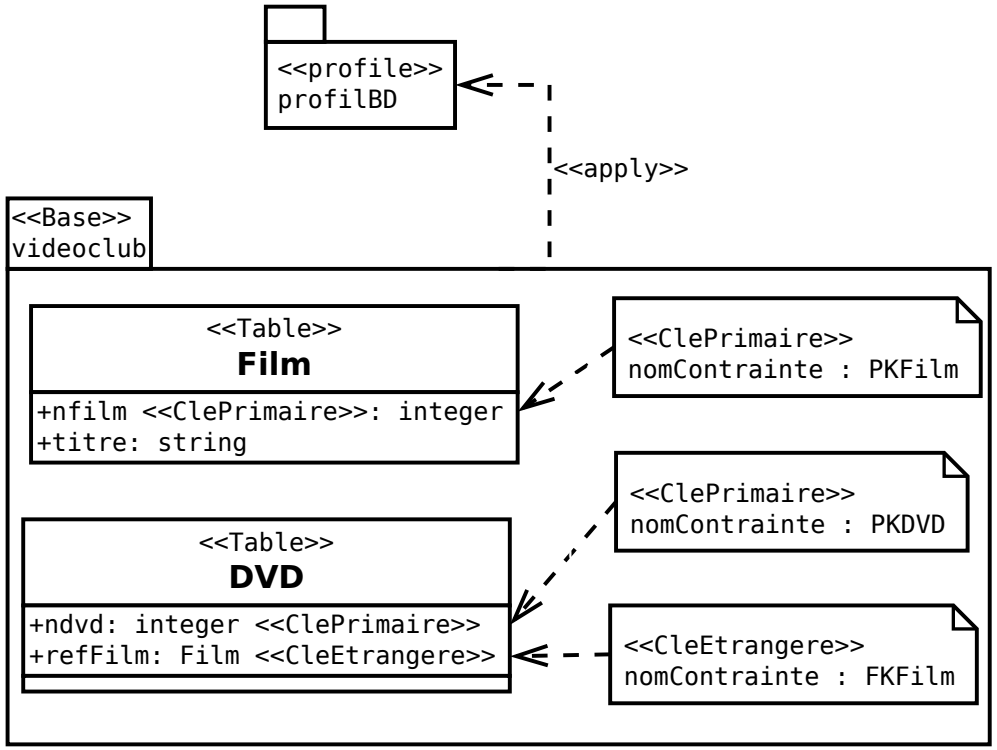
- Phase 3 : Cohérence du profil
- Utilisation du langage OCL

Exemple clé étrangère

```
Context CleEtrangere inv:
  self.base_Property.type.oclIsTypeOf(uml::Class)
```

Application profil

- Définition profil et utilisation du profil sont au même niveau



Discussion Exemple

- De nombreux choix :

Discussion Exemple

- De nombreux choix :
  - CleEtrangere associé à `Association` ou `Property` ?

Discussion Exemple

- De nombreux choix :
  - CleEtrangere associé à `Association` ou `Property` ?
  - Pourquoi un stéréotype associé à `Class` ?

Discussion Exemple

- De nombreux choix :
  - CleEtrangere associé à `Association` ou `Property` ?
  - Pourquoi un stéréotype associé à `Class` ?
- Prolifération de contraintes :
  Pas d'héritage entre classes, une classe ne contient pas d'opérations, ...

L'industrie des profils

- Des profils standardisés existent : profil CORBA 2, profil EJB

L'industrie des profils

- Des profils standardisés existent : profil CORBA 2, profil EJB
- Pas de profil standard pour le modèle relationnel

L'industrie des profils

- Des profils standardisés existent : profil CORBA 2, profil EJB
- Pas de profil standard pour le modèle relationnel
- EMF Eclipse et Profil UML :

L'industrie des profils

- Des profils standardisés existent : profil CORBA 2, profil EJB
- Pas de profil standard pour le modèle relationnel
- EMF Eclipse et Profil UML :
  - Métamodèle UML2 écrit sous Ecore

L'industrie des profils

- Des profils standardisés existent : profil CORBA 2, profil EJB
- Pas de profil standard pour le modèle relationnel
- EMF Eclipse et Profil UML :
  - Métamodèle UML2 écrit sous Ecore
  - Outil de conversion "profil UML" vers modèle ECore

L'industrie des profils

- Des profils standardisés existent : profil CORBA 2, profil EJB
- Pas de profil standard pour le modèle relationnel
- EMF Eclipse et Profil UML :
  - Métamodèle UML2 écrit sous Ecore
  - Outil de conversion "profil UML" vers modèle ECore
  - Exploitation du framework de validation de modèle pour exploiter les contraintes sur les stéréotypes.

Transformation de modèles UML

- Transformation de modèle vers modèle :

Transformation de modèles UML

- Transformation de modèle vers modèle :
  - Endogène : modèle cible et source sont conformes au même métamodèle
    Ex : générer des méthodes accesseurs pour des attributs d'une classe

Transformation de modèles UML

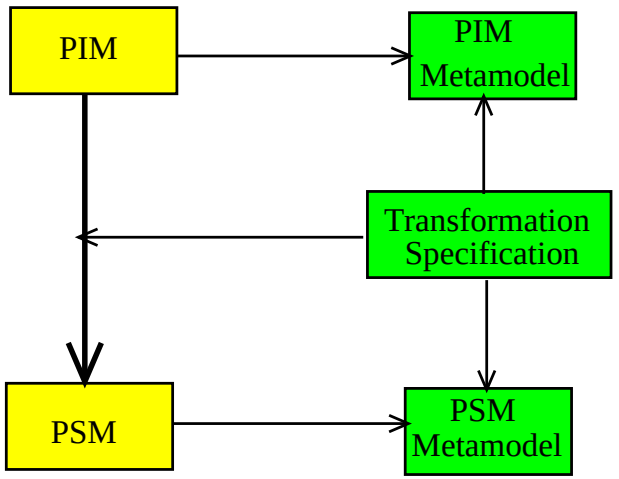
- Transformation de modèle vers modèle :
  - Endogène : modèle cible et source sont conformes au même métamodèle
    Ex : générer des méthodes accesseurs pour des attributs d'une classe
  - Exogène : modèle cible et source sont conformes à des métamodèles différents
    Ex : passage d'un modèle UML à un modèle Relationnel

Transformation de modèles UML

- Transformation de modèle vers modèle :
  - Endogène : modèle cible et source sont conformes au même métamodèle
    Ex : générer des méthodes accesseurs pour des attributs d'une classe
  - Exogène : modèle cible et source sont conformes à des métamodèles différents
    Ex : passage d'un modèle UML à un modèle Relationnel
- Transformation de modèle vers texte
  Ex : passage d'un modèle UML vers code Java

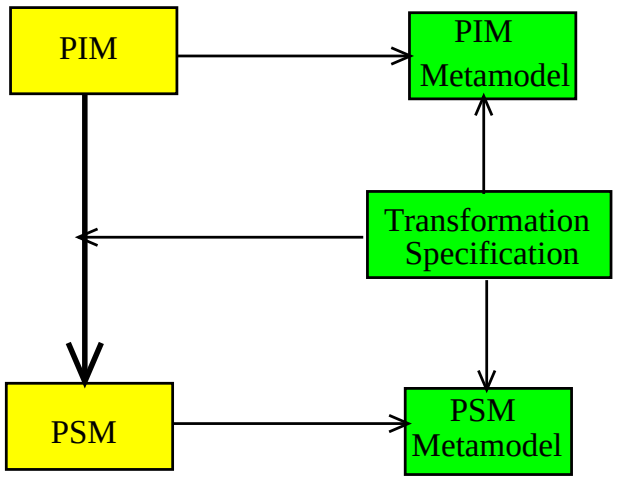
Transformation de modèles vers modèles (1/2)

- Mode universel



Transformation de modèles vers modèles (1/2)

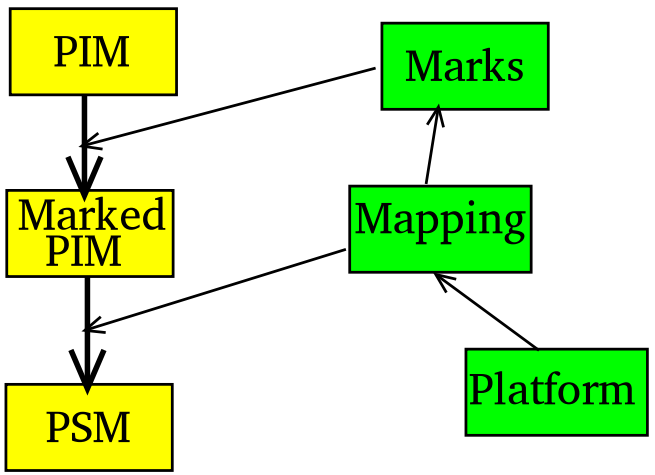
- Mode universel



- Technologies associées : XMI , XSLT, XQuery, QVT (standard OMG)

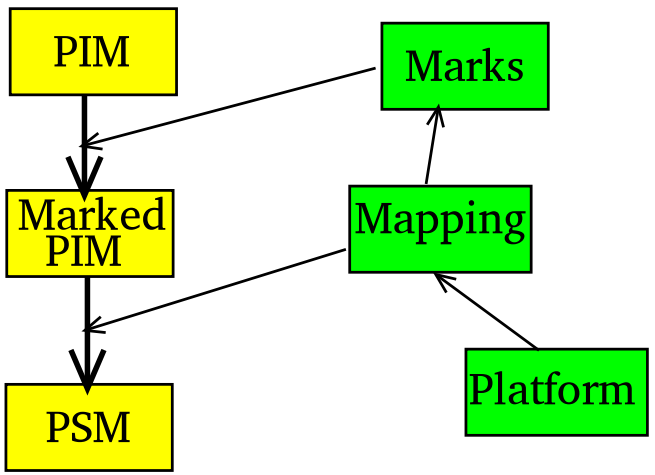
Transformation de modèles UML (2/2)

- Plutôt approche par profil



Transformation de modèles UML (2/2)

- Plutôt approche par profil



- Technologies associées : profils UML, XMI, XSLT, XQuery, QVT (standard OMG)

Processus de transformation de modèles

- Peut nécessiter plusieurs étapes

Processus de transformation de modèles

- Peut nécessiter plusieurs étapes
- Peut nécessiter d'ajouter de la sémantique aux modèles intermédiaires.

Processus de transformation de modèles

- Peut nécessiter plusieurs étapes
- Peut nécessiter d'ajouter de la sémantique aux modèles intermédiaires.

De UML vers SQL

Passage d'un schéma UML à du code SQL, 3 ou 4 étapes :

Processus de transformation de modèles

- Peut nécessiter plusieurs étapes
- Peut nécessiter d'ajouter de la sémantique aux modèles intermédiaires.

De UML vers SQL

Passage d'un schéma UML à du code SQL, 3 ou 4 étapes :

1. Modèle UML

Processus de transformation de modèles

- Peut nécessiter plusieurs étapes
- Peut nécessiter d'ajouter de la sémantique aux modèles intermédiaires.

De UML vers SQL

Passage d'un schéma UML à du code SQL, 3 ou 4 étapes :

1. Modèle UML
2. Ajout du stéréotype : `Identifiant` (Marked PIM)

Processus de transformation de modèles

- Peut nécessiter plusieurs étapes
- Peut nécessiter d'ajouter de la sémantique aux modèles intermédiaires.

De UML vers SQL

Passage d'un schéma UML à du code SQL, 3 ou 4 étapes :

1. Modèle UML
2. Ajout du stéréotype : `Identifiant` (Marked PIM)
3. Transformation vers modèle relationnel (Modèle vers Modèle)

Processus de transformation de modèles

- Peut nécessiter plusieurs étapes
- Peut nécessiter d'ajouter de la sémantique aux modèles intermédiaires.

De UML vers SQL

Passage d'un schéma UML à du code SQL, 3 ou 4 étapes :

1. Modèle UML
2. Ajout du stéréotype : `Identifiant` (Marked PIM)
3. Transformation vers modèle relationnel (Modèle vers Modèle)
4. Génération code SQL (Modèle vers Modèle)

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF
- Query

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF
- Query
  - Langage de requêtes (extension OCL)

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF
- Query
  - Langage de requêtes (extension OCL)
  - Permet de filtrer et sélectionner des éléments d'un modèle

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF
- Query
  - Langage de requêtes (extension OCL)
  - Permet de filtrer et sélectionner des éléments d'un modèle
- Views

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF
- Query
  - Langage de requêtes (extension OCL)
  - Permet de filtrer et sélectionner des éléments d'un modèle
- Views
  - Mécanisme de création de vues

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF
- Query
  - Langage de requêtes (extension OCL)
  - Permet de filtrer et sélectionner des éléments d'un modèle
- Views
  - Mécanisme de création de vues
  - Vue : modèle déduit d'un autre (éventuellement avec son métamodèle associé)

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF
- Query
  - Langage de requêtes (extension OCL)
  - Permet de filtrer et sélectionner des éléments d'un modèle
- Views
  - Mécanisme de création de vues
  - Vue : modèle déduit d'un autre (éventuellement avec son métamodèle associé)
- Transformation

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF
- Query
  - Langage de requêtes (extension OCL)
  - Permet de filtrer et sélectionner des éléments d'un modèle
- Views
  - Mécanisme de création de vues
  - Vue : modèle déduit d'un autre (éventuellement avec son métamodèle associé)
- Transformation
  - Partie opérationnelle, actions de transformation

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF
- Query
  - Langage de requêtes (extension OCL)
  - Permet de filtrer et sélectionner des éléments d'un modèle
- Views
  - Mécanisme de création de vues
  - Vue : modèle déduit d'un autre (éventuellement avec son métamodèle associé)
- Transformation
  - Partie opérationnelle, actions de transformation
  - Permet de créer des vues

Un bref mot sur QVT

- **Q**uery **V**iews **T**ransformation
- Standard de l'OMG
- Ensemble de langages permettant d'exprimer des transformations entre modèles définis avec le MOF
- Query
  - Langage de requêtes (extension OCL)
  - Permet de filtrer et sélectionner des éléments d'un modèle
- Views
  - Mécanisme de création de vues
  - Vue : modèle déduit d'un autre (éventuellement avec son métamodèle associé)
- Transformation
  - Partie opérationnelle, actions de transformation
  - Permet de créer des vues
- Plus loin avec site OMG-MOF, slides B. Combemale (IRISA)

Modèle vers Texte

- Langage MTL (standard OMG-MOF, ”Model to Text Transformation Language”)

Modèle vers Texte

- Langage MTL (standard OMG-MOF, ”Model to Text Transformation Language”)
- Source : un fichier XMI, Cible : un ou plusieurs fichiers texte

Modèle vers Texte

- Langage MTL (standard OMG-MOF, ”Model to Text Transformation Language”)
- Source : un fichier XMI, Cible : un ou plusieurs fichiers texte
- Langage à base de tags, Définition de ”templates” ou ”query”

Modèle vers Texte

- Langage MTL (standard OMG-MOF, ”Model to Text Transformation Language”)
- Source : un fichier XMI, Cible : un ou plusieurs fichiers texte
- Langage à base de tags, Définition de ”templates” ou ”query”
- Exploitation d’OCL pour la navigation de modèles

Modèle vers Texte

- Langage MTL (standard OMG-MOF, ”Model to Text Transformation Language”)
- Source : un fichier XMI, Cible : un ou plusieurs fichiers texte
- Langage à base de tags, Définition de ”templates” ou ”query”
- Exploitation d’OCL pour la navigation de modèles
- Structures de contrôle (if, for, . . .)

Modèle vers Texte

- Langage MTL (standard OMG-MOF, ”Model to Text Transformation Language”)
- Source : un fichier XMI, Cible : un ou plusieurs fichiers texte
- Langage à base de tags, Définition de ”templates” ou ”query”
- Exploitation d’OCL pour la navigation de modèles
- Structures de contrôle (if, for, . . .)
- Eclipse : plugin Acceleo MTL, génération code Java du transformateur

MTL : déclaration de module

- Tag `module` : quel est le métamodèle concerné (ligne 2)
- Tag `comment` : commentaire (ici encoding est traité, ligne 1)

Exemple déclaration module

```
1  [comment encoding = UTF-8 /]
2  [module generate('http:///DB.ecore')]
3  ...
```

MTL : templates (1/2)

- Tag `template` : la notion de fonction
- Tout ce qui n'est pas entre crochets est généré tel quel
- Polymorphisme paramétrique
- Fonctions principales :
  - template annoté main : fonction principale (ligne 2)
  - Indication de la métaclasse racine du fichier XMI (ligne 1)

Exemple fonction principale

```
1 [template public generateBase(aBase : Base)]
2 [comment @main/]
3 [file (aBase.nom+'.sql', false, 'UTF-8')] ... [/file]
4 [/template]
```

MTL : templates (2/2)

- Post-traitement : se réalise avant la génération du texte final (ligne 3, clause post)
- Possibilité de définir des variables locales au template (ligne 2)

Exemple templates

```
1  [template public foo(aBase : Base)
2    { nomBase : String = aBase.nom ; }
3    post (trim()) ]
4  ...
5  [/template]
```

MTL : Instructions if

- Tags : `if`, `else` et `elseif`

Syntaxe instruction if

```
1  [if (condition)]
2    ...
3  [else]
4    ...
5  [/if]
```

MTL : Instructions for

- Tag : `for`
- Définitions de comportements, avant, entre et après les itérations `before()`, `separator()` et `after()` (ligne 4)

Exemple fonction principale

```
1 [for (iterator : Type | expression)]
2   ...
3 [/for]
4 [for (...) separator(', ') after(';') ] ... [/for]
```

MTL : Les Queries

- Récupère des objets, ne génère pas de texte (template)

Exemple fonction principale

```
1 [query public getTables(b : Base) : Set(Table)
2   = c.tables /]
```

Du modèle relationnel vers SQL (1/4)

Exemple fil rouge

```
[comment encoding = UTF-8 /]
[module generate('http:///DB.ecore')]

[template public generateBase(aBase : Base)]
[comment @main/]
[file (aBase.nom+'.sql', false, 'UTF-8')]
[for (t : Table | aBase.tables)]
        [generateTable(t) /]
[/for]
[/file]
[/template]
```

Du modèle relationnel vers SQL (2/4)

Exemple fil rouge

```
[template public generateTable(t : Table) ]
create table [t.nom /] (
[for (c : Colonne | t.colonnes)]
        [generateColonne(c) /]
[/for]
[for (c : Contrainte | t.contraintes) separator(',')]
[generateContrainte(c) /]
[/for]
) ;
[/template]
```

Du modèle relationnel vers SQL (3/4)

Exemple fil rouge

```
[template public generateColonne(c : Colonne)]
[c.nom/] [generateTypeSQL(c)/],
[/template]

[template public generateTypeSQL(c:Colonne) post (trim())]
[if (c.typeSQL = TypeSQL::Entier)] int
[elseif (c.typeSQL = TypeSQL::Texte)] text
[elseif (c.typeSQL = TypeSQL::Chaine)]
  varchar([c.longueur /]) [/if]
[/template]
```

Du modèle relationnel vers SQL (4/4)

Exemple fil rouge (polymorphisme paramétrique)

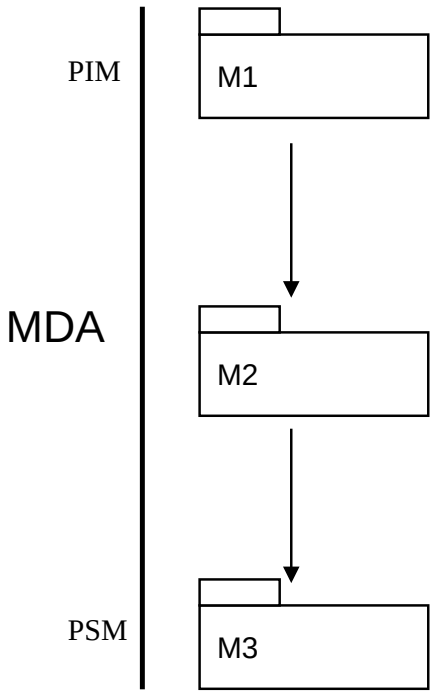
```
[template public generateContrainte(c : Contrainte)
          post (trim())]
[/template]
[template public generateContrainte(c : ClePrimaire)
          post (trim())]
constraint [c.nom /] primary key ([c.col.nom /])
[/template]
[template public generateContrainte(c : CleEtrangere)
          post (trim())]
constraint [c.nom /] foreign Key ([c.col.nom /])
  references [c.references.nom /]
[/template]
```

Le résultat de la transformation

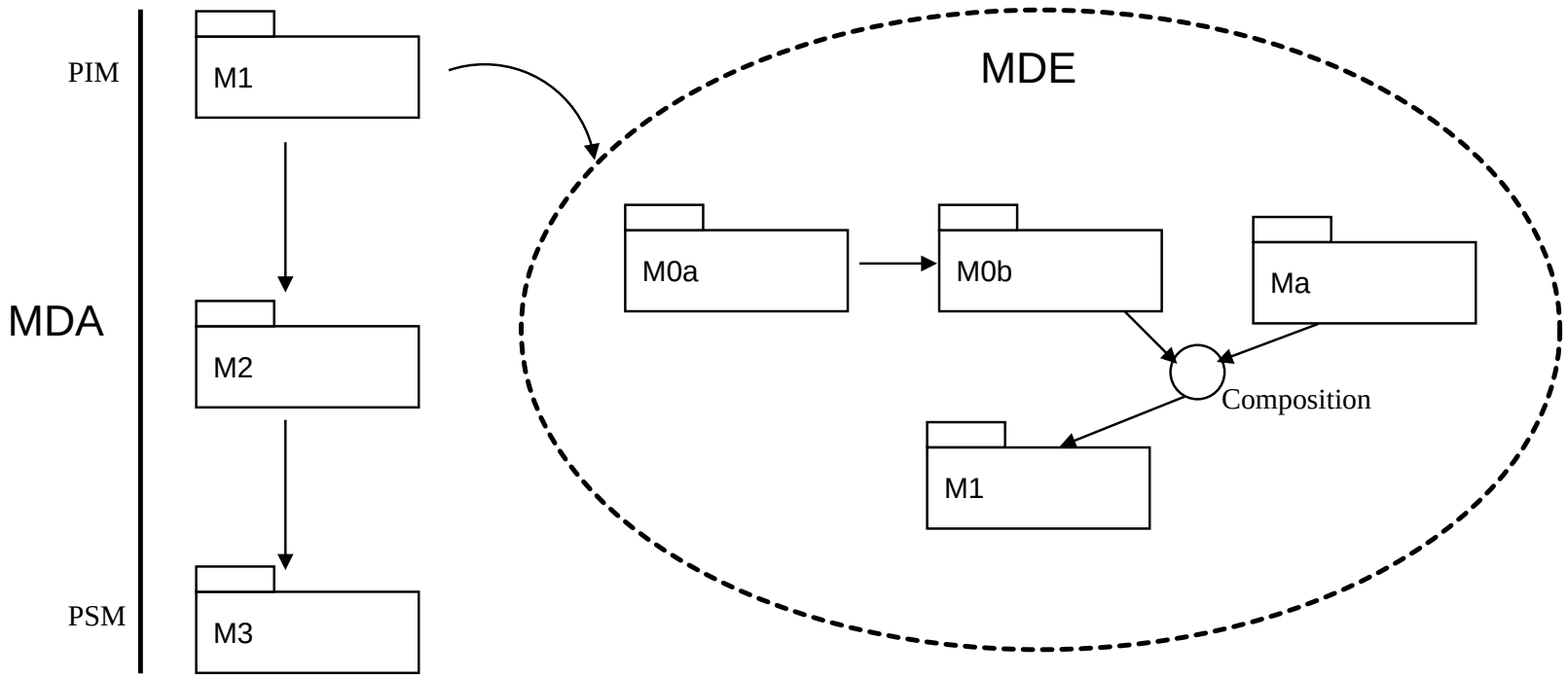
Exemple fil rouge, code généré

```
create table film (
  titre  varchar(20) ,
  nfilm  int ,
  constraint PKfilm primary key (nfilm)
) ;
create table dvd (
  ndvd  int ,
  refFilm  int ,
  constraint PKDVD primary key (ndvd)
  ,constraint FKFilm foreign Key (refFilm) references film
) ;
```

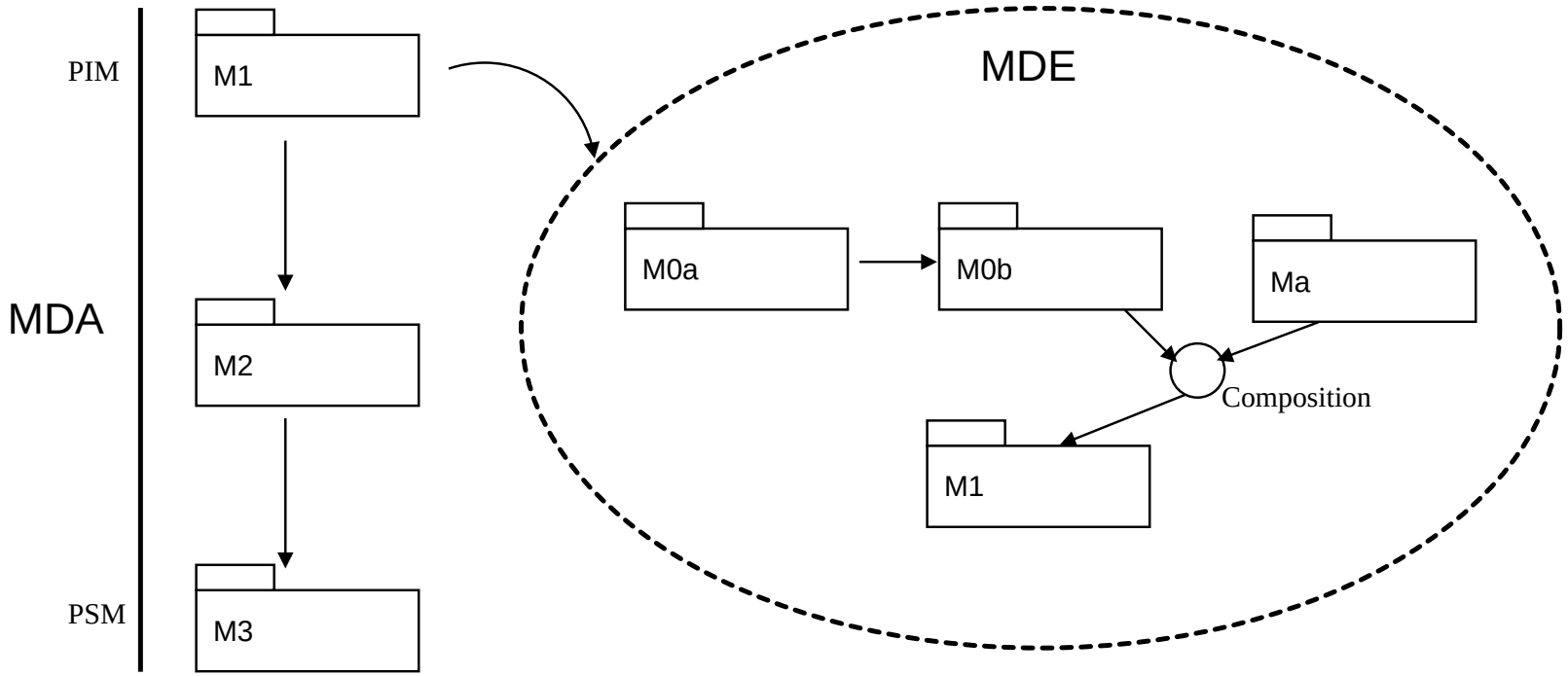
de MDA à MDE



de MDA à MDE



de MDA à MDE



MDE

- Pas seulement une vision descendante (”à la MDA”)

MDE

- Pas seulement une vision descendante (”à la MDA”)
- Les modèles comme des artefacts manipulables.

MDE

- Pas seulement une vision descendante (”à la MDA”)
- Les modèles comme des artefacts manipulables.
- MDE par l’exemple :

MDE

- Pas seulement une vision descendante (”à la MDA”)
- Les modèles comme des artefacts manipulables.
- MDE par l’exemple :
  - Feature Model (ligne de produits)

MDE

- Pas seulement une vision descendante (”à la MDA”)
- Les modèles comme des artefacts manipulables.
- MDE par l’exemple :
  - Feature Model (ligne de produits)
  - Modèles réutilisables

Feature Model

- Variabilité des modèles, comment exprimer toutes les variantes ?

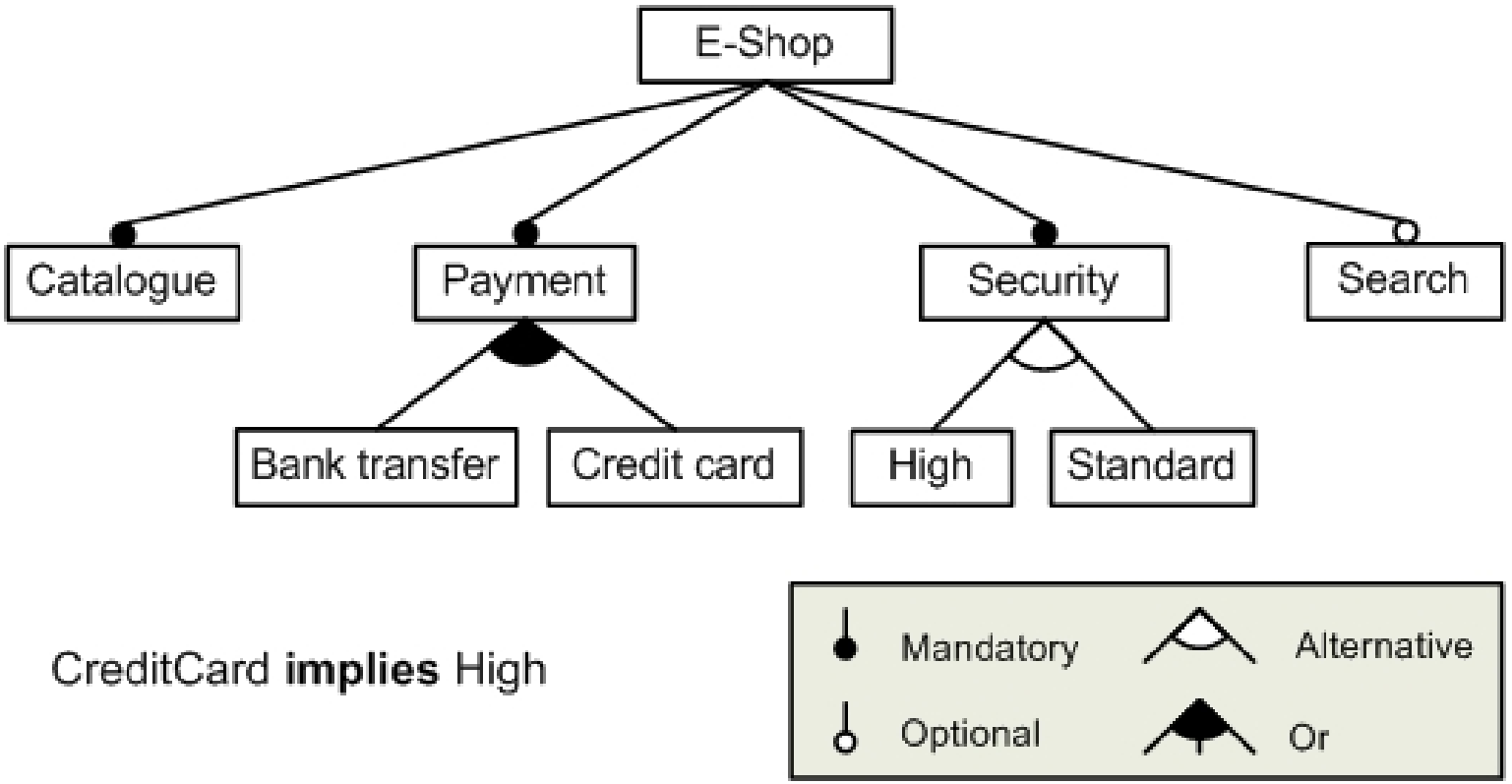
Feature Model

- Variabilité des modèles, comment exprimer toutes les variantes ?
- Est-ce qu'une configuration est conforme ?

Feature Model

- Variabilité des modèles, comment exprimer toutes les variantes ?
- Est-ce qu'une configuration est conforme ?
- Lignes de produits logiciels

Feature Model, un exemple (from Wikipedia)



CreditCard **implies** High

Feature Model, apports MDE

- Un métamodèle de ”feature model”, instanciation de ”feature model”

Feature Model, apports MDE

- Un métamodèle de ”feature model”, instanciation de ”feature model”
- Lien de composition ”estConforme” entre un modèle et son ”feature model”

Feature Model, apports MDE

- Un métamodèle de ”feature model”, instanciation de ”feature model”
- Lien de composition ”estConforme” entre un modèle et son ”feature model”
- Outil de vérification de conformité

Feature Model, apports MDE

- Un métamodèle de ”feature model”, instanciation de ”feature model”
- Lien de composition ”estConforme” entre un modèle et son ”feature model”
- Outil de vérification de conformité
- Outil de complétion de modèle : que faut-il ajouter pour être conforme ?

Feature Model, apports MDE

- Un métamodèle de ”feature model”, instanciation de ”feature model”
- Lien de composition ”estConforme” entre un modèle et son ”feature model”
- Outil de vérification de conformité
- Outil de complétion de modèle : que faut-il ajouter pour être conforme ?
- . . .

Modèles réutilisables

- Verrous scientifiques :

Modèles réutilisables

- Verrous scientifiques :
  - Comment faire des modèles réutilisables ?

Modèles réutilisables

- Verrous scientifiques :
  - Comment faire des modèles réutilisables ?
  - Comment les rendre génériques ?

Modèles réutilisables

- Verrous scientifiques :
  - Comment faire des modèles réutilisables ?
  - Comment les rendre génériques ?
  - Comment exprimer et valider la composition de modèles ?

Modèles réutilisables

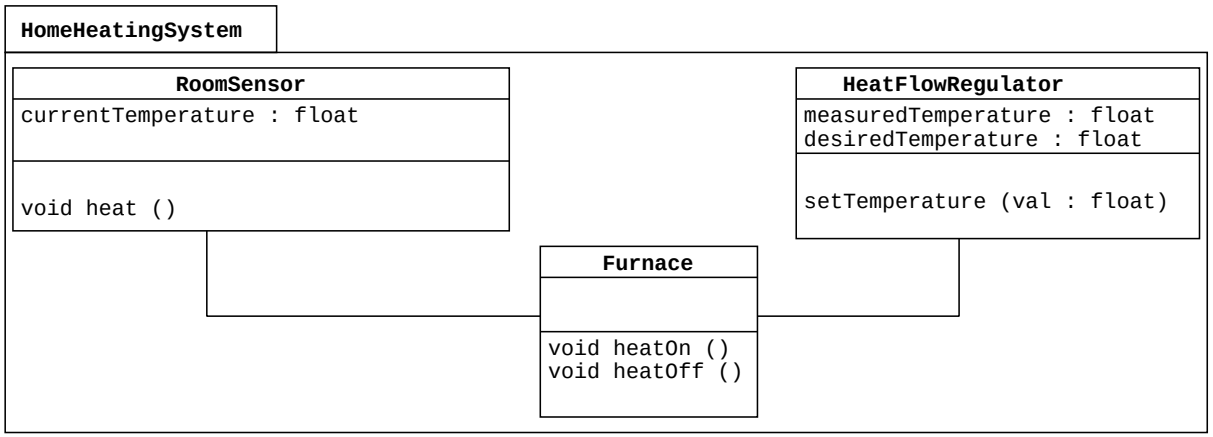
- Verrous scientifiques :
  - Comment faire des modèles réutilisables ?
  - Comment les rendre génériques ?
  - Comment exprimer et valider la composition de modèles ?
  - Peut-on composer les modèles génériques entre eux ?

Modèles réutilisables

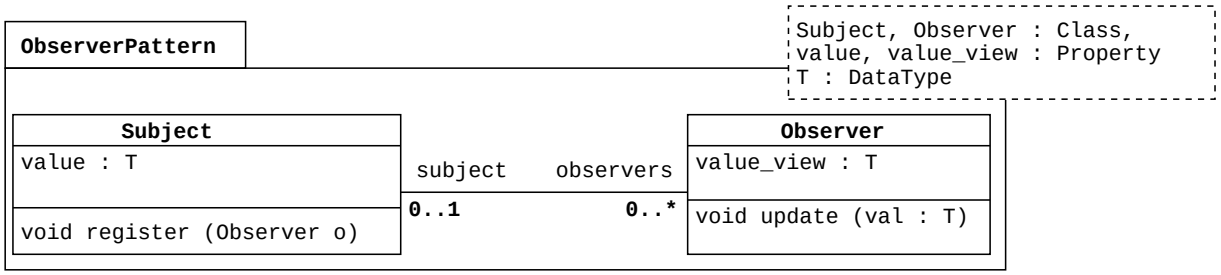
- Verrous scientifiques :
  - Comment faire des modèles réutilisables ?
  - Comment les rendre génériques ?
  - Comment exprimer et valider la composition de modèles ?
  - Peut-on composer les modèles génériques entre eux ?
  - . . .

Une solution : les modèles paramètrables (Cocoa - LIFL)

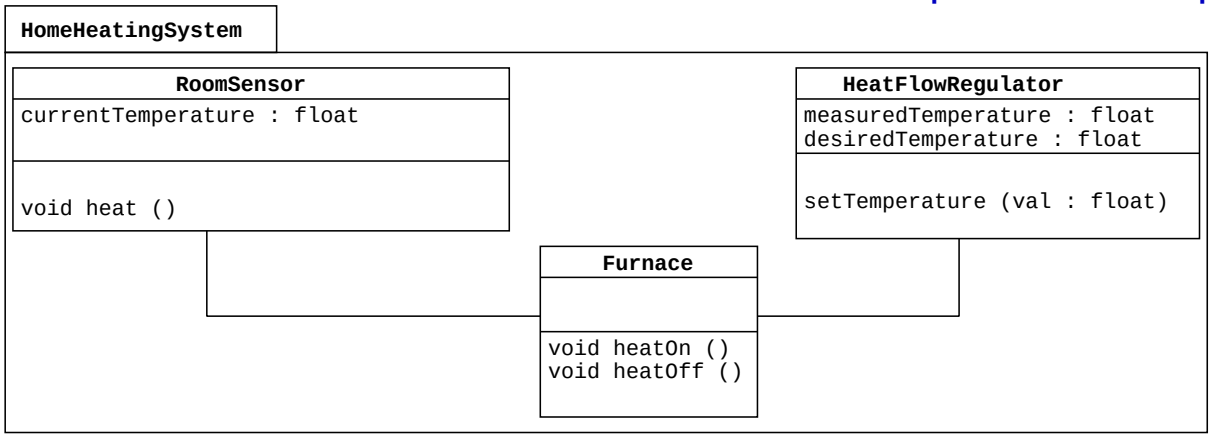
1.
Edition UML



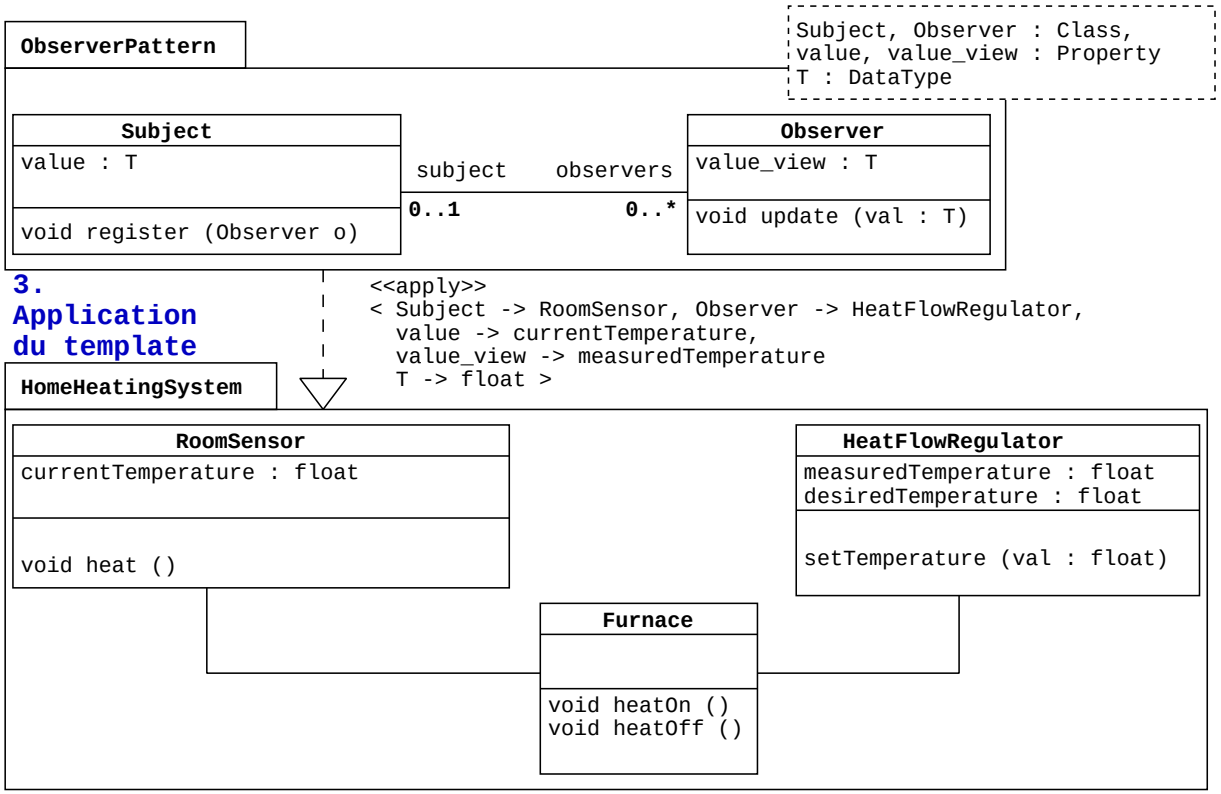
Une solution : les modèles paramètrables (Cocoa - LIFL)



2.
Importation de template

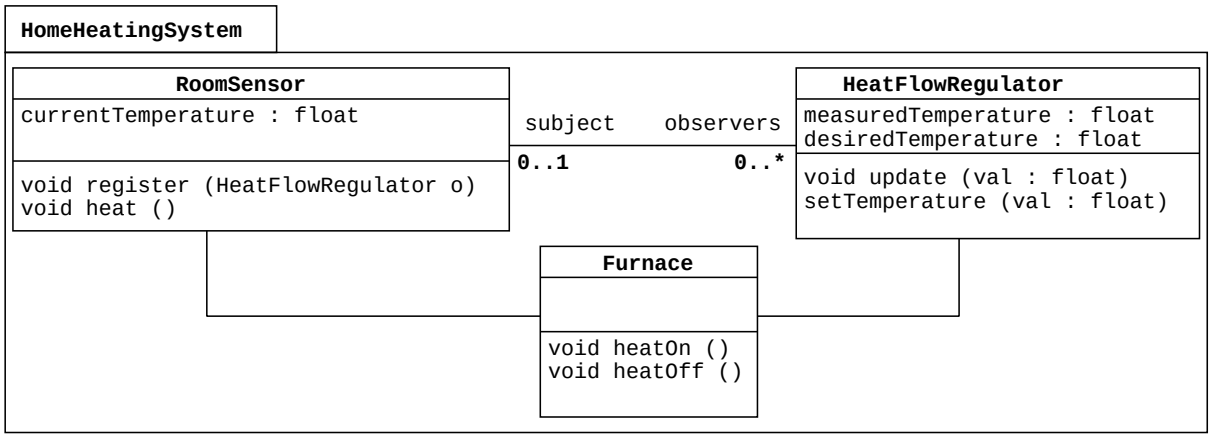


Une solution : les modèles paramètrables (Cocoa - LIFL)

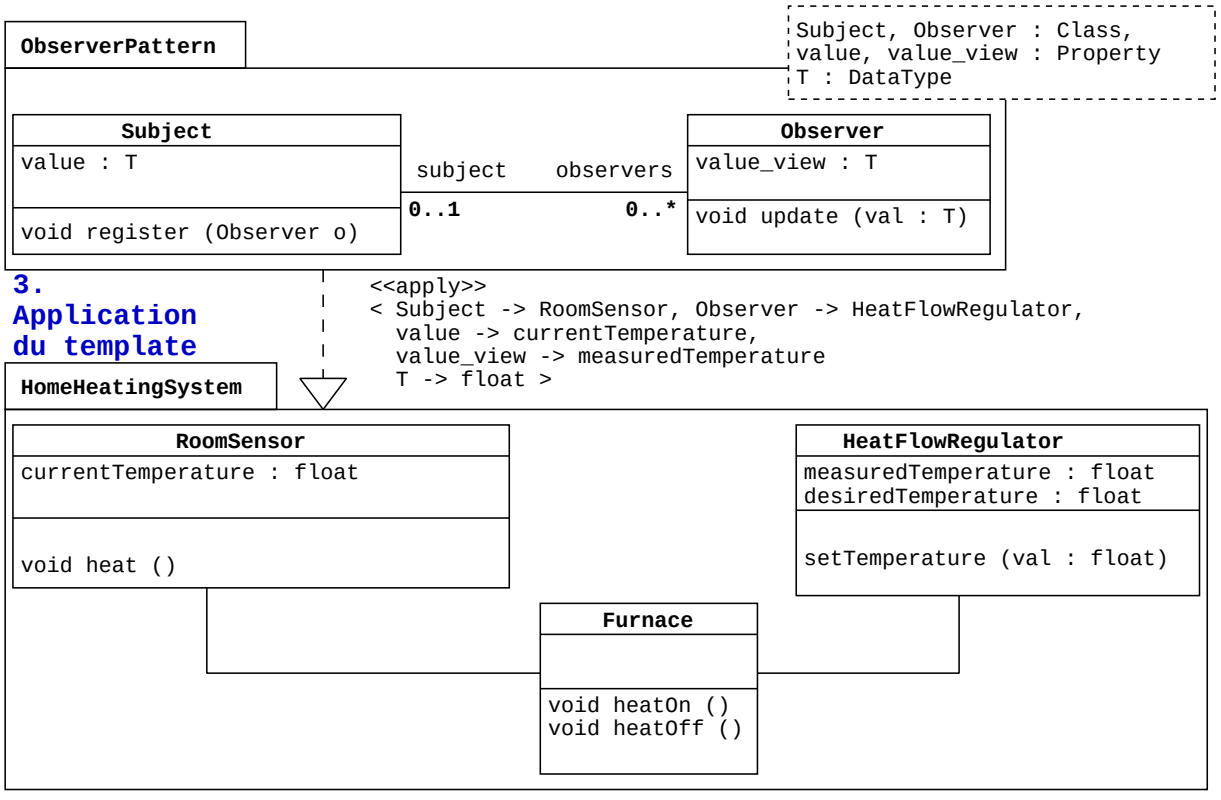


Une solution : les modèles paramètrables (Cocoa - LIFL)

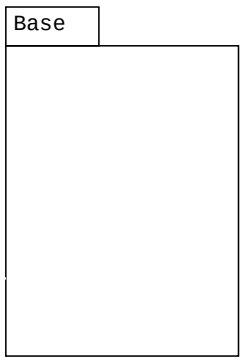
4. Transformation



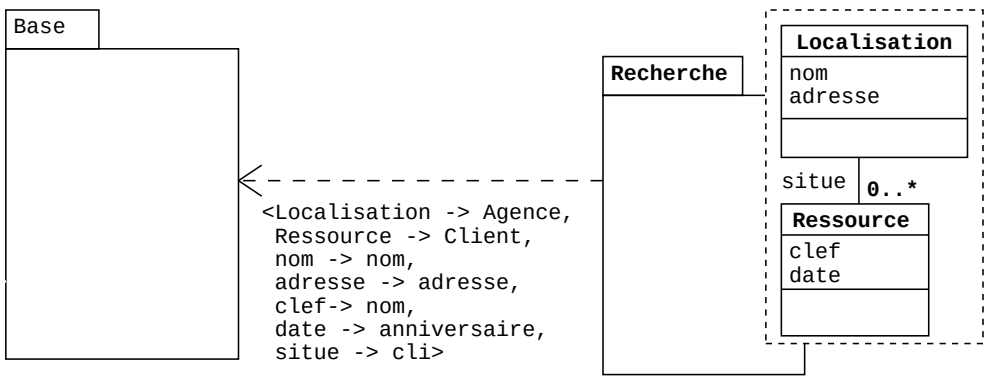
Une solution : les modèles paramètrables (Cocoa - LIFL)



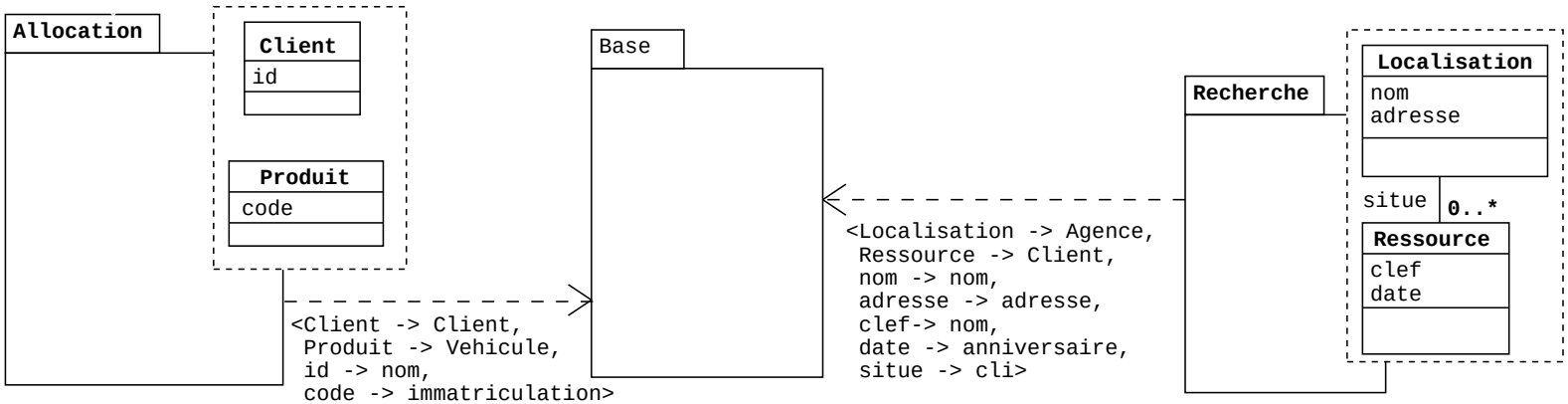
Modèles paramétrables



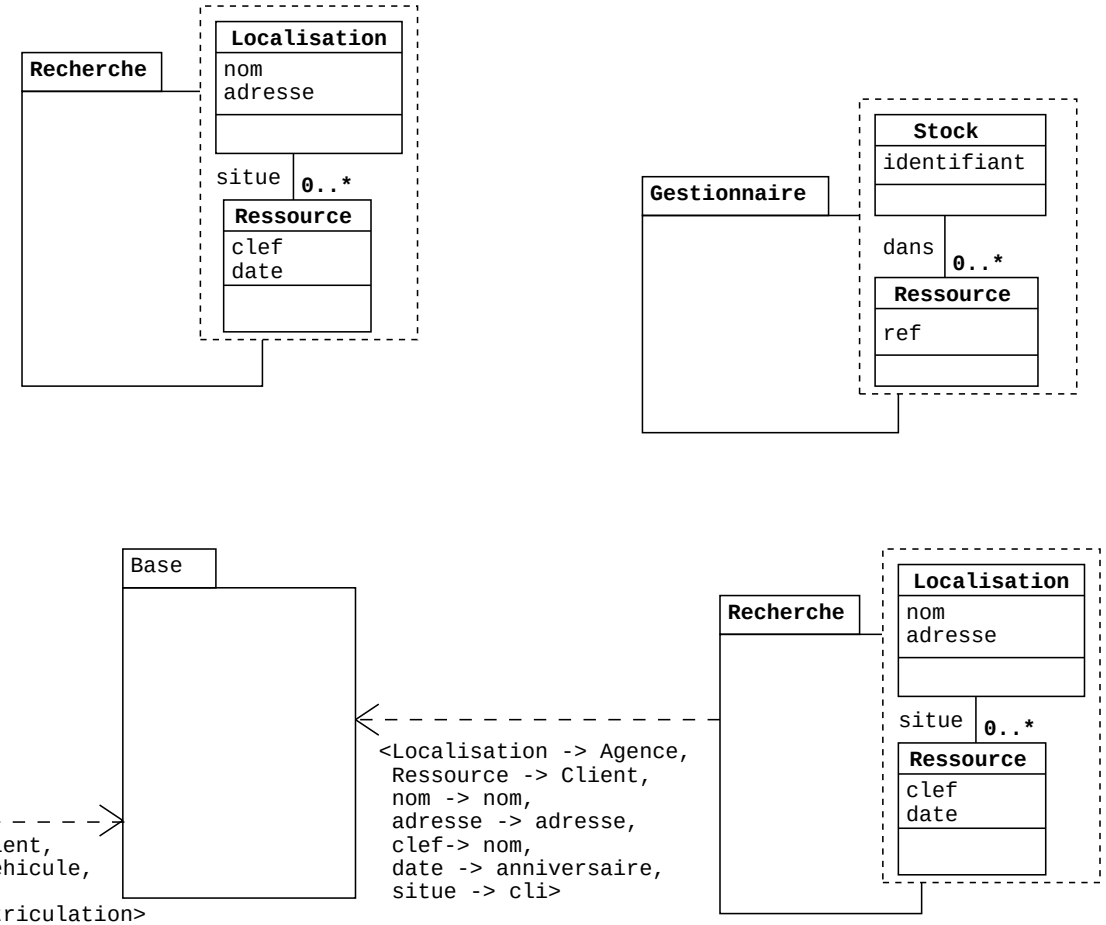
Modèles paramétrables



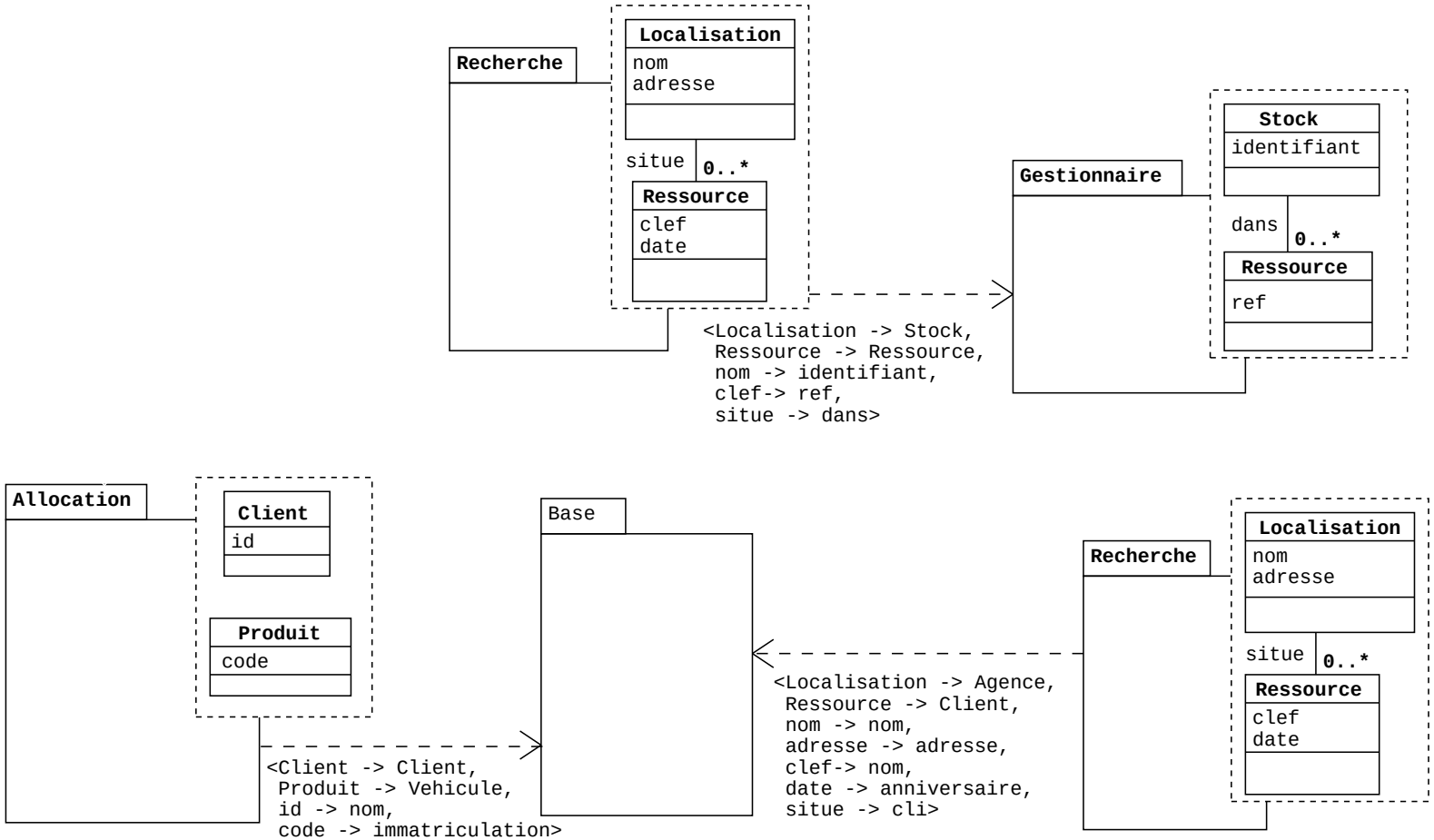
Modèles paramétrables



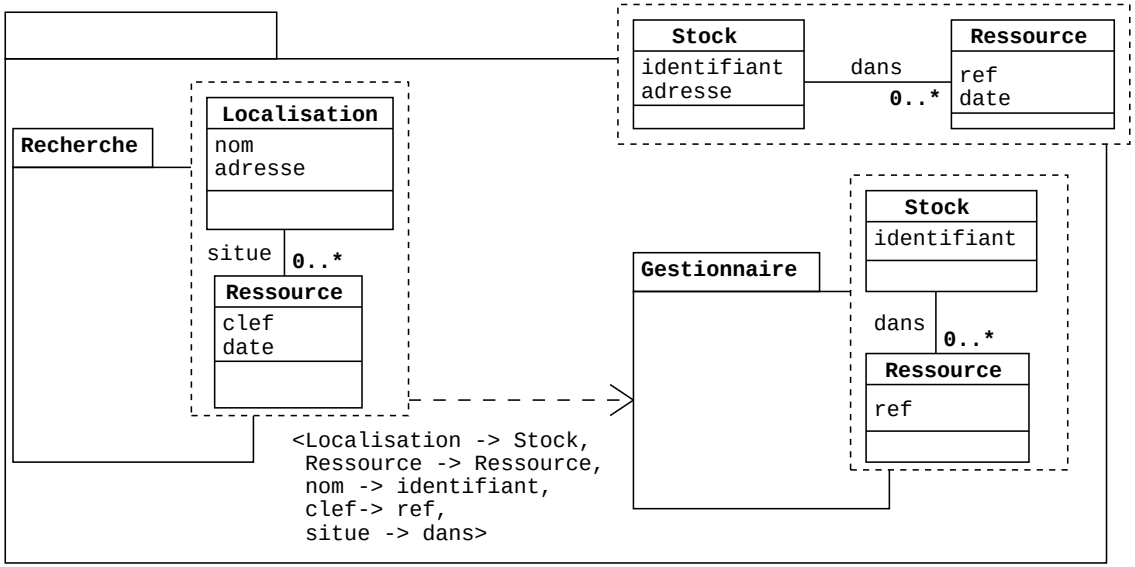
Modèles paramétrables

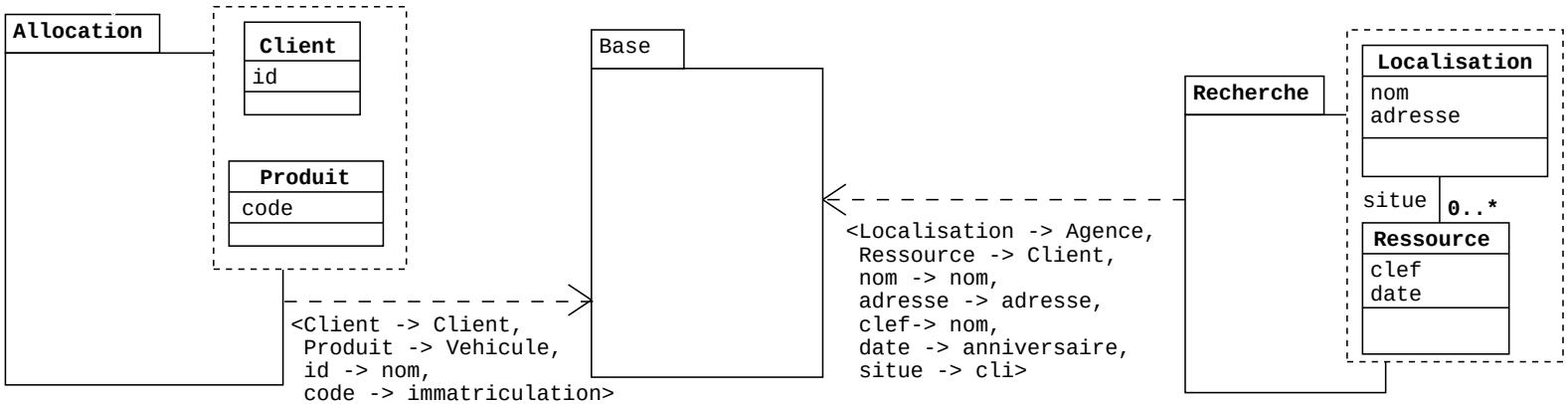


Modèles paramétrables

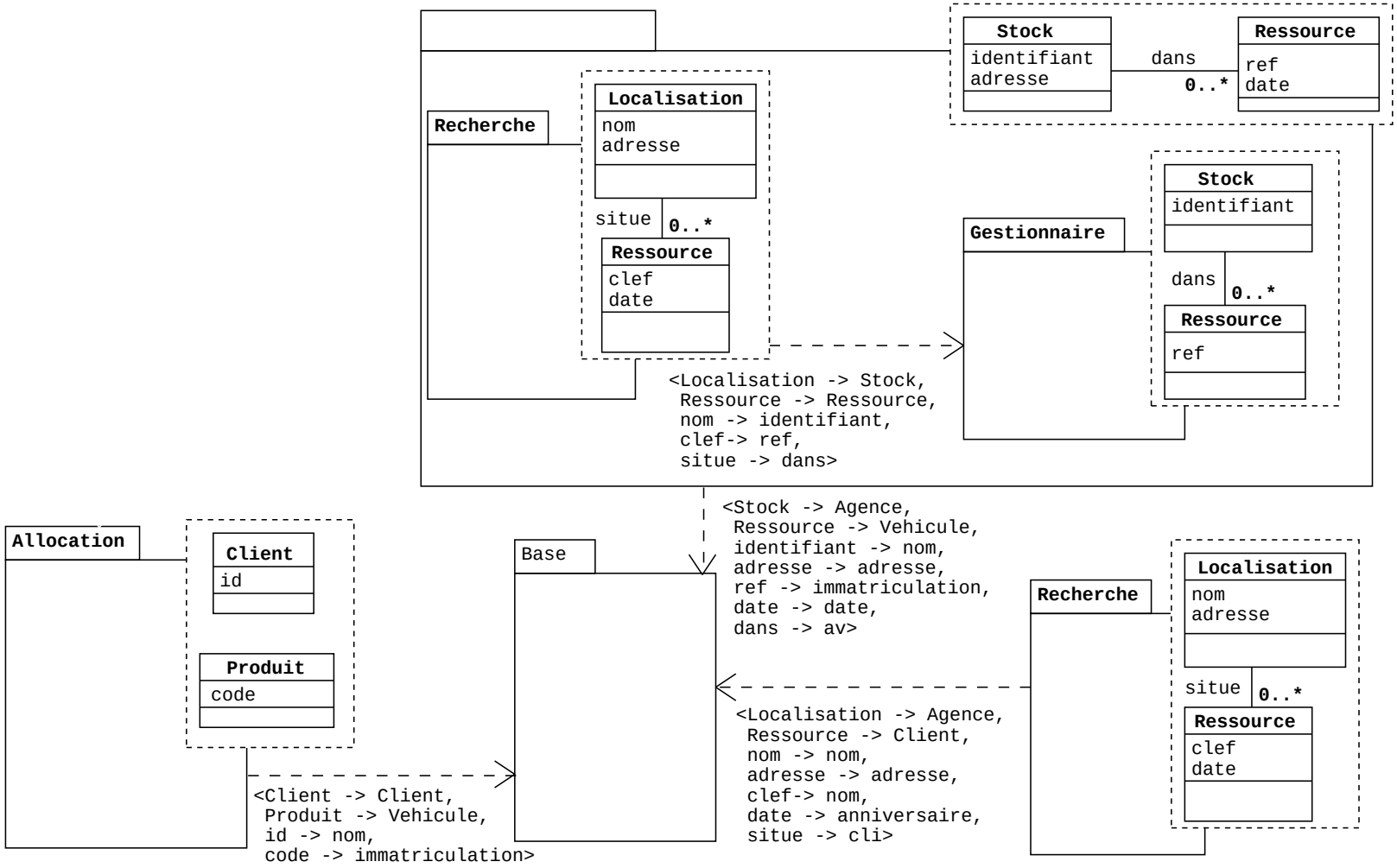


Modèles paramétrables

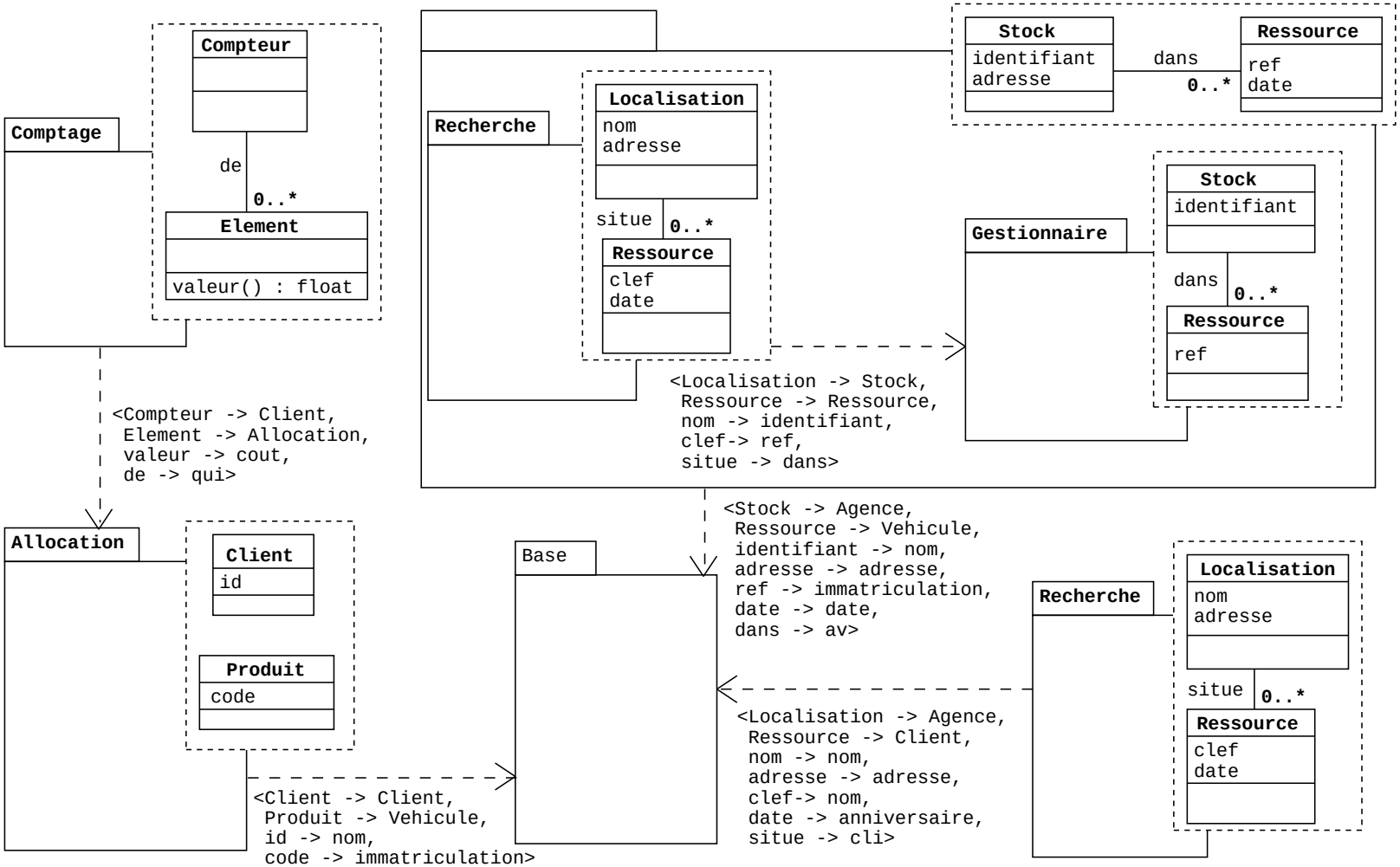




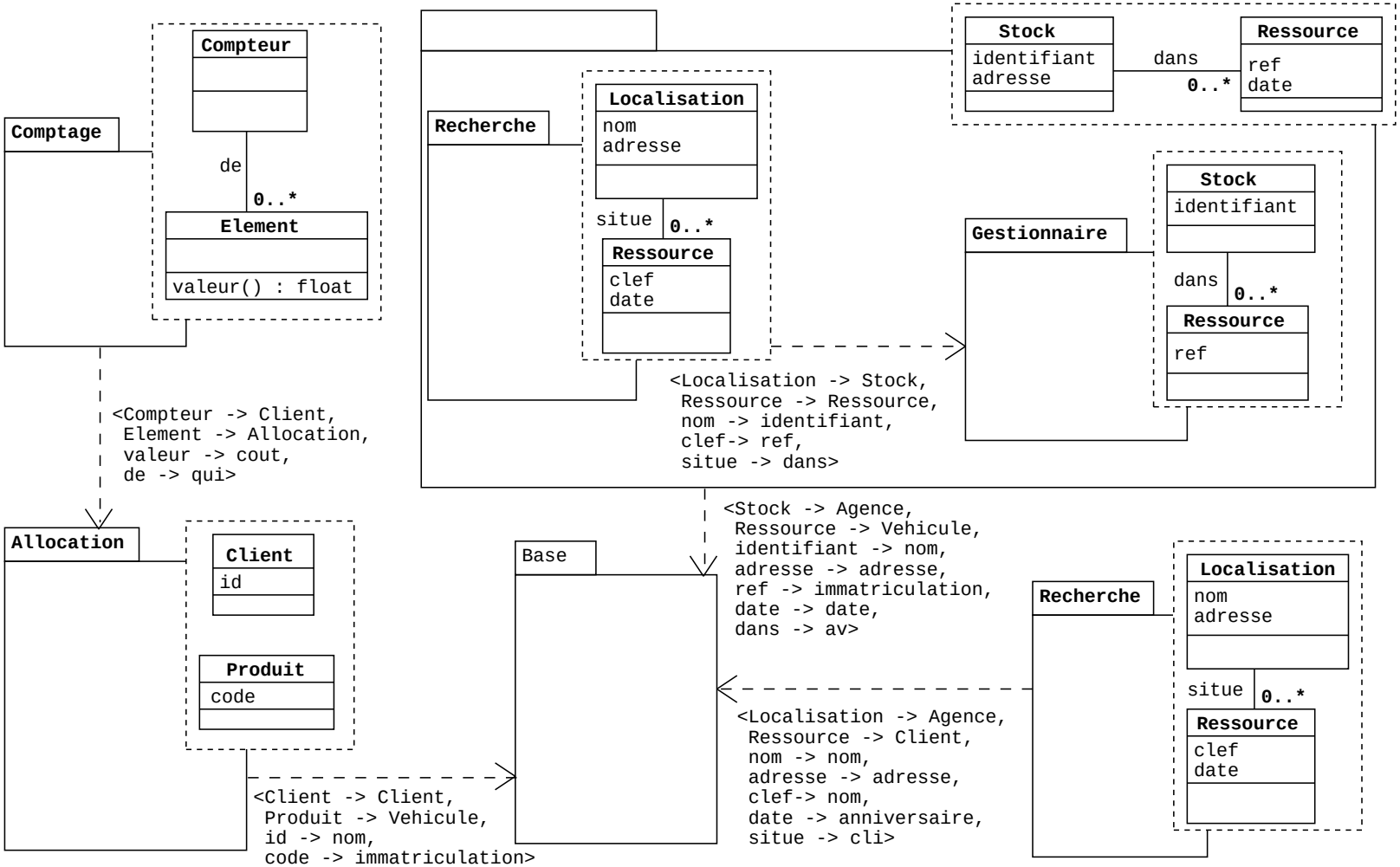
Modèles paramétrables



Modèles paramétrables



Modèles paramétrables



Modèles paramétrables, apports MDE

- Un métamodèle de ”modèles paramétrables” (templates) et/ou un profil UML

Modèles paramétrables, apports MDE

- Un métamodèle de ”modèles paramétrables” (templates) et/ou un profil UML
- Lien de composition entre modèle et template

Modèles paramétrables, apports MDE

- Un métamodèle de ”modèles paramétrables” (templates) et/ou un profil UML
- Lien de composition entre modèle et template
- Outil de vérification de conformité

Modèles paramétrables, apports MDE

- Un métamodèle de ”modèles paramétrables” (templates) et/ou un profil UML
- Lien de composition entre modèle et template
- Outil de vérification de conformité
- Outil de transformation

Modèles paramétrables, apports MDE

- Un métamodèle de ”modèles paramétrables” (templates) et/ou un profil UML
- Lien de composition entre modèle et template
- Outil de vérification de conformité
- Outil de transformation
- Outil de complétion de modèle : que faut-il ajouter pour que la composition soit correcte ?

Modèles paramétrables, apports MDE

- Un métamodèle de ”modèles paramétrables” (templates) et/ou un profil UML
- Lien de composition entre modèle et template
- Outil de vérification de conformité
- Outil de transformation
- Outil de complétion de modèle : que faut-il ajouter pour que la composition soit correcte ?
- Outil de génération (MDA)

Modèles paramétrables, apports MDE

- Un métamodèle de ”modèles paramétrables” (templates) et/ou un profil UML
- Lien de composition entre modèle et template
- Outil de vérification de conformité
- Outil de transformation
- Outil de complétion de modèle : que faut-il ajouter pour que la composition soit correcte ?
- Outil de génération (MDA)
- …