1ère année Master GLSD

Département d'Informatique

Université de Biskra

Année 2019/2020

APPROCHES AVANCÉES DE DÉVELOPPEMENT

Introduction à l'ingénierie des modèles

Dr. Mohamed Lamine Kerdoudi

Email: lamine.kerdoudi@gmail.com

Plan

- Pourquoi l'ingénierie des modèles (IDM)
  - Constat sur l'évolution des technologies
  - Approche MDA de l'OMG
- Pourquoi faire
  - But, problématique de l'IDM
- Définitions générales
  - Modèles, méta-modèles, transformations

Évolution des technologies

- Évolution permanente des technologies logicielles
- **Exemple** : **Systèmes distribués**
  - Faire communiquer et interagir des éléments distants
- Évolution dans ce domaine
  - C et sockets TCP/UDP
  - C et RPC
  - C++ et CORBA
  - Java et RMI
  - Java et EJB
  - C# et Web Services
  - A suivre ...

Évolution des technologies

- Idée afin de limiter le nombre de technologies
  - Normaliser un standard qui sera utilisé par tous
- Pour les systèmes distribués
  - Normaliser un intergiciel (middleware)
  - C'était le but de **CORBA**
- **CORBA** : **C**ommon **O**bject Request **B**roker **A**rchitecture
  - Norme de l'OMG : Object Management Group
    - Consortium d'industriels (et d'académiques) pour développement de standards

Évolution des technologies

- Principes de **CORBA**
  - Indépendant des langages de programmation
  - Indépendant des systèmes d'exploitation
  - Pour interopérabilité de toutes applications, indépendamment des technologies utilisées

- Mais **CORBA** ne s'est pas réellement imposé en pratique
  - D'autres middleware sont apparus : **Java RMI**
  - Plusieurs implémentations de CORBA
    - Ne réalisant pas tous entièrement la norme
  - Les composants logiciels sont arrivés
    - OMG a développé un modèle de composants basé sur CORBA : **CCM (Corba Component Model)**
    - Mais, Sun EJB, MS .Net, Web Services sont là ...

Évolution des technologies

- De plus, « guerre » de la standardisation et/ou de l'universalité
  - Sun : Java
    - Plate-forme d'exécution universelle
    - Avec intergiciel intégré (RMI)
  - OMG : CORBA
    - Intergiciel universel
  - Microsoft et d'autres : Web Services
    - Interopératibilité universelle entre composants
    - Intergiciel = HTTP/XML
- Middleware ou middle war * ?

Évolution des technologies

- Evolutions apportent un gain réel:
  - Communications distantes
    - Socket : envoi d'informations brutes
    - RPC/RMI/CORBA : appel d'opérations sur un élément distant presque comme s'il était local
    - Composants : meilleure description et structuration des interactions (appels d'opérations)
  - Paradigmes de programmation
    - C : procédural
    - Java, C++, C# : objet
      - Encapsulation, réutilisation, héritage, spécialisation ...
    - EJB, CCM : composants
      - Meilleure encapsulation et réutilisation, déploiement ...

Évolution des technologies

- Conclusion sur l'évolution des technologies
  - Nouveaux paradigmes, nouvelles techniques
  - Pour développement toujours plus rapide, plus efficace
  - Rend difficile la standardisation (désuétude rapide d'une technologie)
    - Et aussi car combats pour imposer sa technologie
- Principes de cette évolution
  - Évolution sans fin
  - La meilleure technologie est ... celle à venir

Évolution des technologies

- **Quelles conséquences en pratique de cette évolution permanente ?**
- Si on veut profiter des nouvelles technologies et de leurs avantages :
  - Nécessite d'adapter une application à ces technologies
- **Question : quel est le coût de cette adaptation ?**
  - Généralement très élevé
    - Doit réécrire presque entièrement l'application
    - Car mélange du code métier et du code technique
    - Aucune capitalisation de la logique et des règles métiers

Évolution des technologies

Exemple

- Application de calculs scientifiques distribués sur un réseau de machines
- Passage de C/RPC à Java/EJB
  - ➢ Impossibilité de reprendre le code existant
    - ❑ Paradigme procédural à objet/composant
- Pourtant
  - ➢ Les algorithmes de distribution des calculs et de répartition des charges sur les machines sont indépendants de la technologie de mise en œuvre
    - ❑ Logique métier indépendante de la technologie

Évolution des technologies

- Partant de tous ces constats
  - ➢ Nécessité de découpler clairement la logique métier et de la mise en œuvre technologique

  ➔ C'est un des principes fondamentaux de l'**Ingénierie des Modèles**
    - ❑ Séparation des préoccupations (separation of concerns)
- Besoin de modéliser/spécifier
  - ➢ A un niveau abstrait la partie métier
  - ➢ La plate-forme de mise en œuvre
  - ➢ De projeter ce niveau abstrait sur une plateforme

Model Driven Architecture -MDA-

- Approche Model-Driven Architecture (MDA) de l'OMG
  - Origine de l'ingénierie des modèles
  - Date de fin 2000
- Le MDA est né à partir des constatations que nous venons de voir
  - **Évolution continue des technologies**
- But du MDA
  - **Abstraire les parties métiers de leur mise en œuvre**
  - Basé sur des technologies et standards de l'OMG
    - **UML, MOF, OCL, CWM, QVT ...**

Model Driven Architecture -MDA-

Si je crée une visualisation d'une partie d'un système, est ce que cela signifie que je pratiquais MDA?

Il n'y a pas de réponse définitive!

Model Driven Architecture -MDA-

- Mais, il y a un consensus croit que: MDA est plus étroitement associée aux approches de model-driven dans laquelle :
  - ➢ le code est (semi-) automatiquement généré à partir des modèles plus abstrait,
  - ➢ et qui utilise des langages de spécification standard pour décrire ces modèles et les transformations entre eux.
- MDA est une variante de l'**I**ngénierie **D**irigée par les **M**odèles (**IDM**, ou MDE « Model Driven Engineering »).

- La clé de MDA est l'importance des modèles.

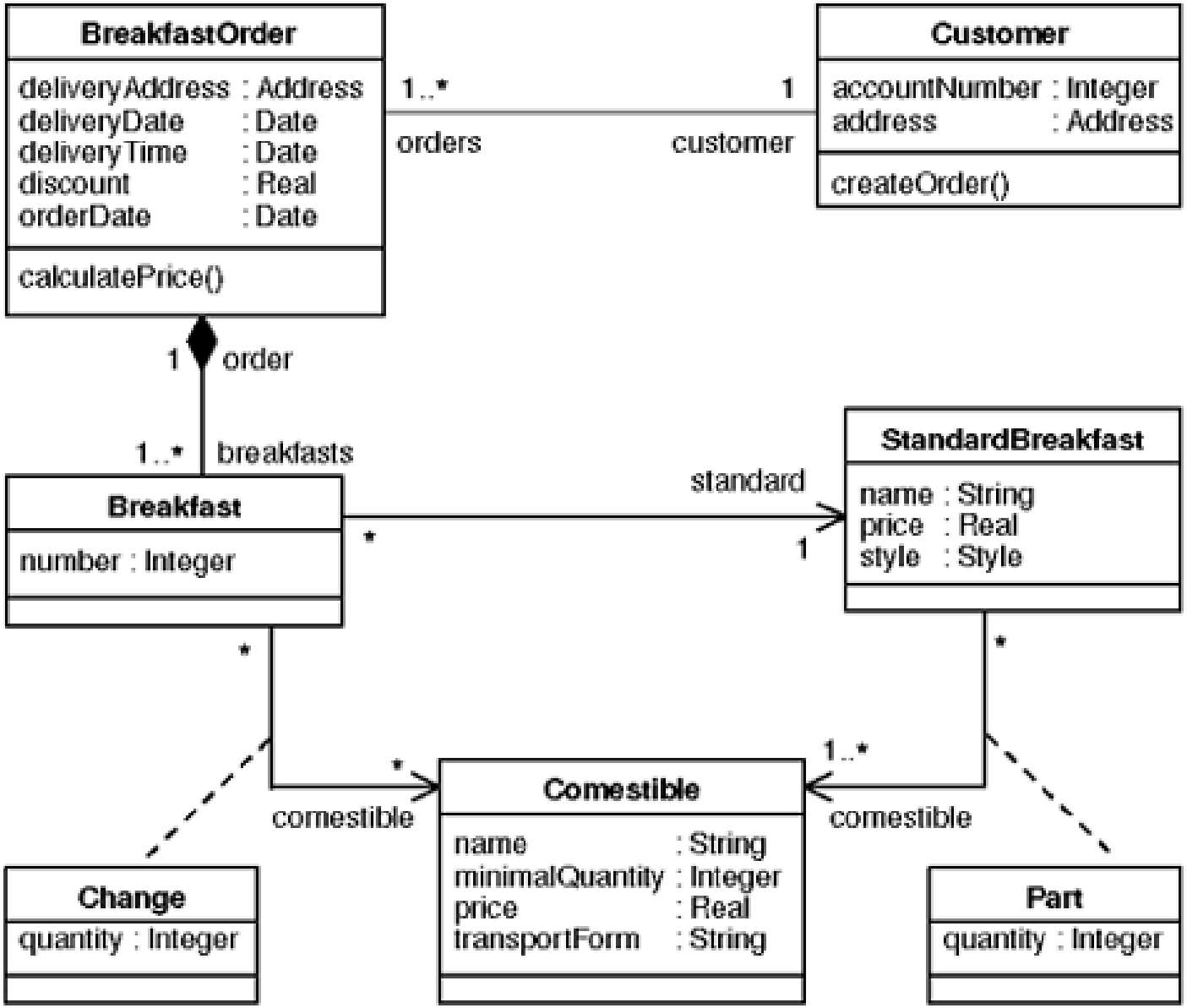
Model Driven Architecture -MDA-

- Le MDA définit 2 principaux niveaux de modèles
  - **PIM : Platform Independent Model**
    - Modèle spécifiant une application indépendamment de la technologie de mise en œuvre
    - Uniquement spécification de la partie métier d'une application
  - **PSM : Platform Specific Model**
    - Modèle spécifiant une application après projection sur une plate-forme technologique donnée
- Autres types de modèles
  - CIM : Computation Independent Model
    - Spécification du système, point de vue extérieur de l'utilisateur

    CIM peut contenir des information métier sur l'organisation, les rôles, les fonctions, les processus et les activités, la documentation, … etc.
  - PDM : Plateform Deployment Model:
    - Modèle d'une plate-forme de déploiement: est un modèle de transformation pour permettre le passage du PIM vers le PSM

Model Driven Architecture -MDA-

- **Exemple de PIM : Breakfast System**

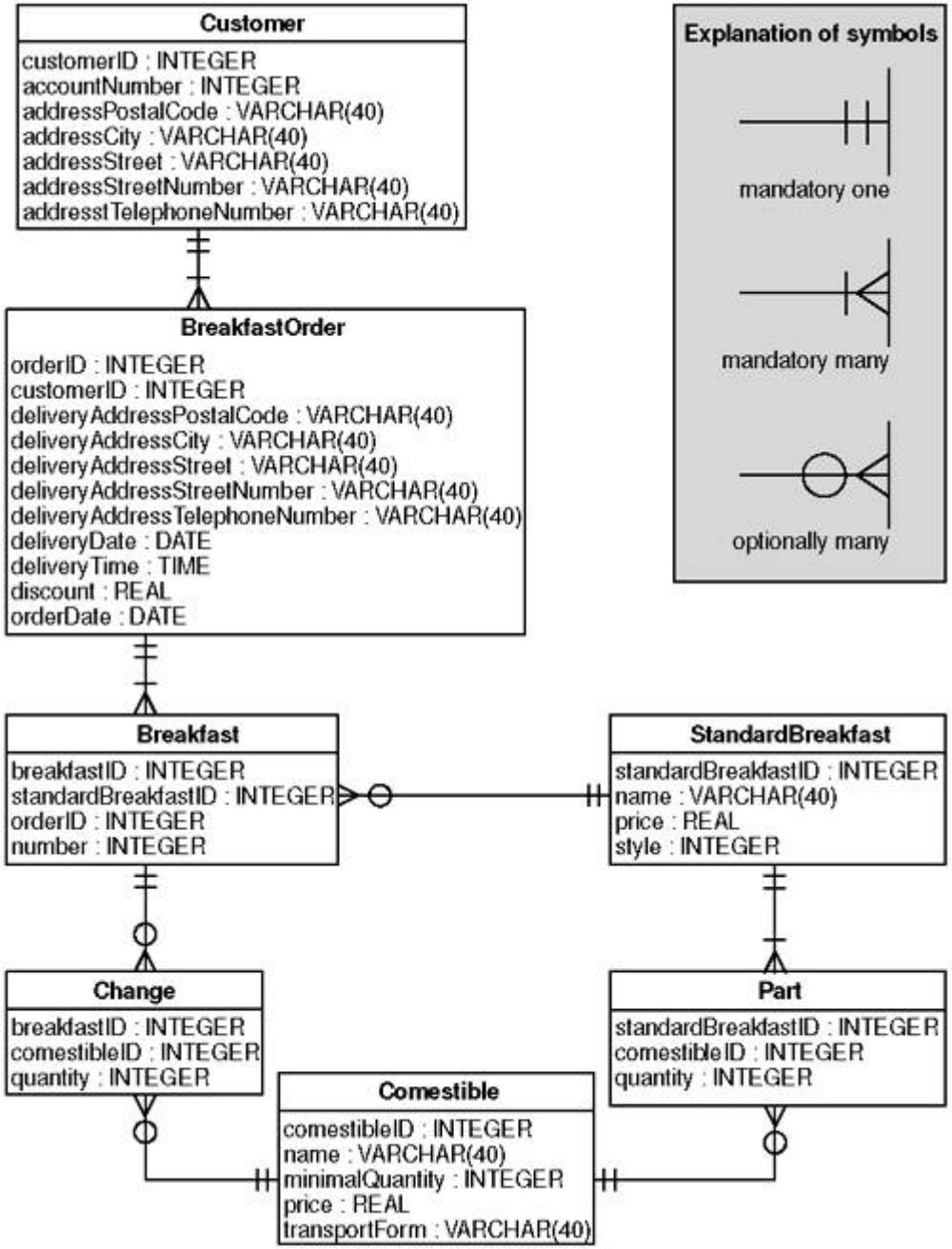


Model Driven Architecture -MDA-

- **Exemple de PIM : Breakfast System**

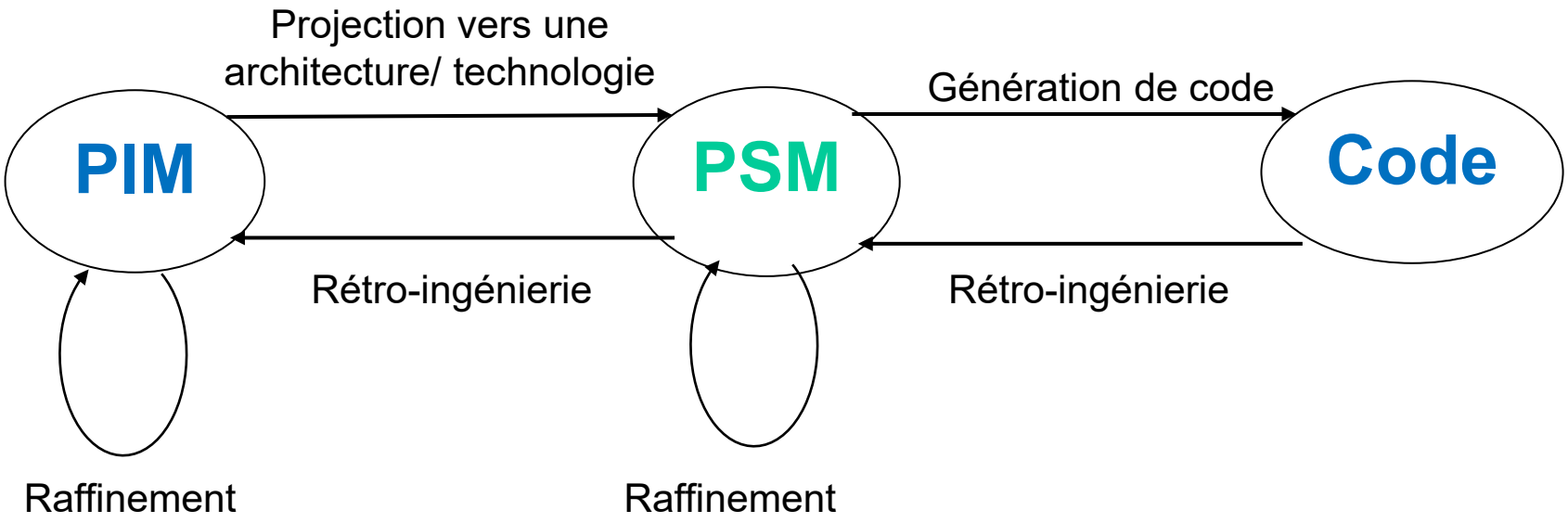
- Ce PIM peut être transformé en un PSM relationnel, un PSM Web et un PSM EJB.
- En prenant en compte les relations entre ces PSM.
- Les trois PSMs seront ensuite transformés en trois modèles de code séparés.

➢ Exemple de PSM : Relational PSM



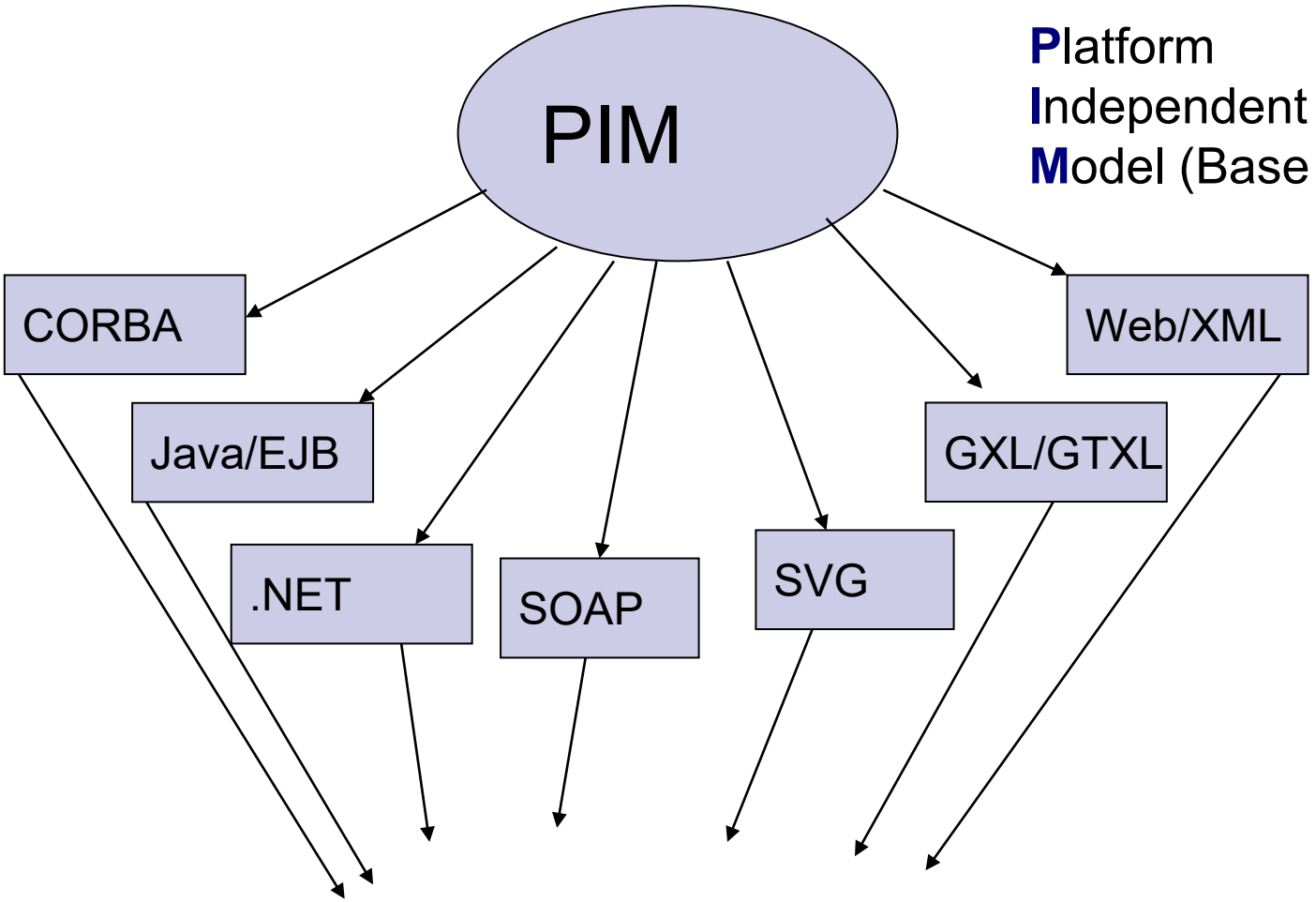
Model Driven Architecture -MDA-

- Relation entre les niveaux de modèles:



Model Driven Architecture -MDA-

MDA: Model Once, Generate Anywhere



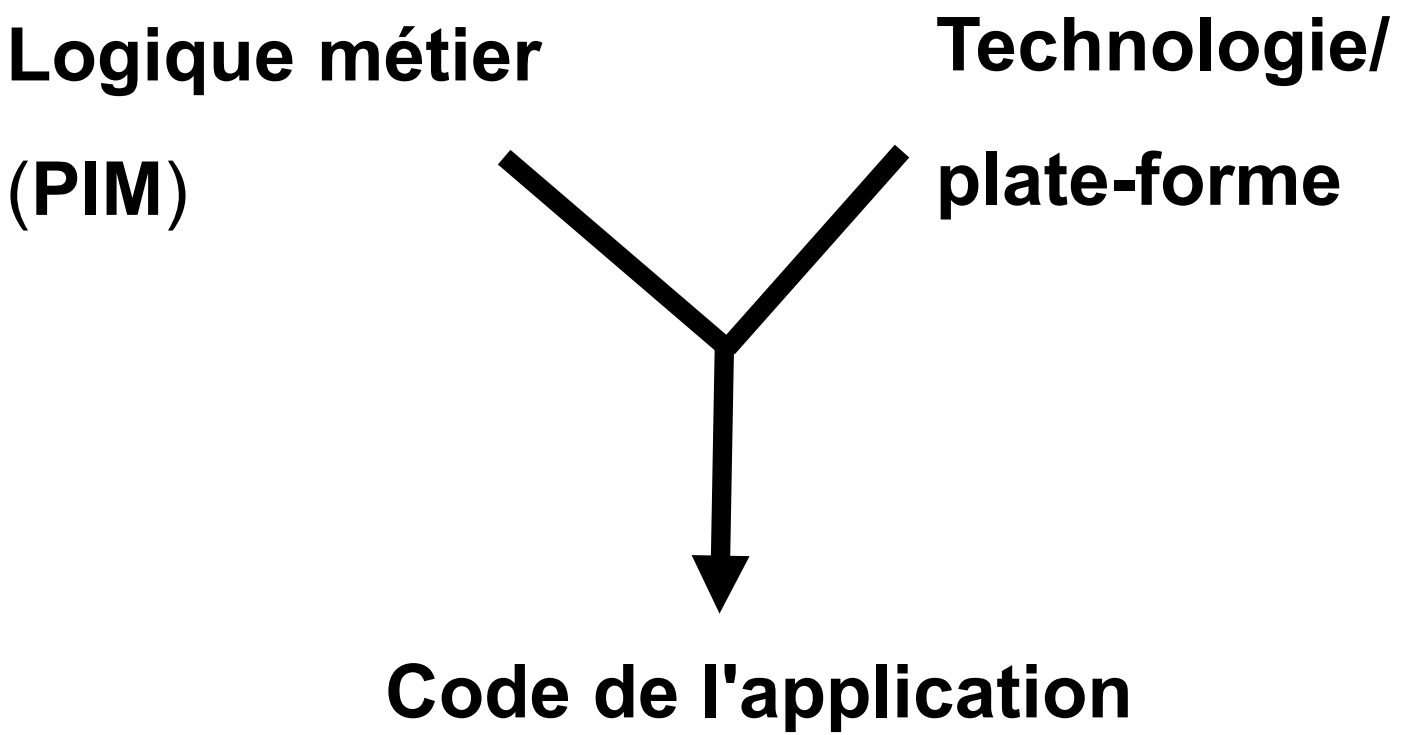
Platform
Independent
Model (Base Model in UML)

Platform
Specific
Models

Platformes MDA supportées

Model Driven Architecture -MDA-

- Cycle de développement d'un logiciel selon le MDA
- Cycle en Y
  - Plus complexe en pratique



Model Driven Architecture -MDA-

- Outils de mise en œuvre du MDA
  - Standards de **l'OMG**
- Spécification des modèles aux différents niveaux
  - Langage de modélisation **UML**
    - **Profils UML**
  - Langage de contraintes **OCL**
  - Langages dédiés à des domaines particuliers (**CWM** ...)
- Spécification des méta-modèles
  - Meta Object Facilities (**MOF**)
- Langage de manipulation et transformation de modèles
  - Query/View/Transformation (**QVT**)
  - ….

Model Driven Engineering

- **Limites du MDA:**
  - Technologies OMG principalement

Ingénierie Dirigée par les Modèles (IDM)

- Approche plus globale et générale que le MDA

- Appliquer les mêmes principes à tout espace technologique et les généraliser
  - **Espace technologique** : ensemble de techniques/principes de modélisation et d'outils associés à un (méta)méta-modèle particulier
  - MOF/UML, EMF/Ecore, XML, grammaires de langages, bases de données relationnelles, ontologies …

- Le **MDA** est un processus de type **MDE**

▪ Principes de MDE

❖ Capitalisation

- ➢ Dans l'approche objets/composants
  - ❑ Capitalisation: c'est la définition et la réutilisation d'éléments logiciels/code.
- ➢ **Dans MDE**
  - ❑ Capitalisation: réutilisation de (parties de) modèles : logique métier, règles de raffinements, de projection ...

❖ Abstraction

- ➢ Modéliser, c'est abstraire ...
- ➢ Par exemple, abstraction des technologies de mise en œuvre permet de:
  - ❑ adapter une logique métier à un contexte
  - ❑ évoluer bien plus facilement vers de nouvelles technologies

Principes du MDE

❖ Modélisation

- La modélisation n'est pas une discipline récente en génie logiciel
- Les processus de développement logiciel non plus
  - ➢ RUP, Merise ...
- **C'est l'usage de ces modèles qui change**
- Le but du MDE est
  - ➢ De passer à une **vision réellement productive**
    - ❑ Pour générer le code final du logiciel pour une technologie de mise en œuvre donnée

Principes du MDE

❖ Séparation des préoccupations

- Deux principales préoccupations
  - Métier : le cœur de l'application, sa logique
  - Plate-forme de mise en œuvre
- Mais plusieurs autres préoccupations possibles
  - Sécurité
  - Interface utilisateur
  - Qualité de service
  - ...
- Chaque préoccupation est modélisée par un modèle

▪ Principes du MDE

❖ Pour passer à une **vision productive**, il faut

- ➢ Que les modèles soient **bien définis**
  - ❑ Notion de **méta-modèle**

- ➢ Pour que l'on puisse les manipuler et les **interpréter via des outils**
  - ❑ Traitement de méta-modèles différents simultanément
    - ○ Pour transformation/passage entre 2 espaces technologiques différents
  - ❑ Outils et langages de transformation, de projection, de génération de code
  - ❑ Langages dédiés à des domaines (DSL : Domain Specific Language)

Model Driven Engineering: Définitions

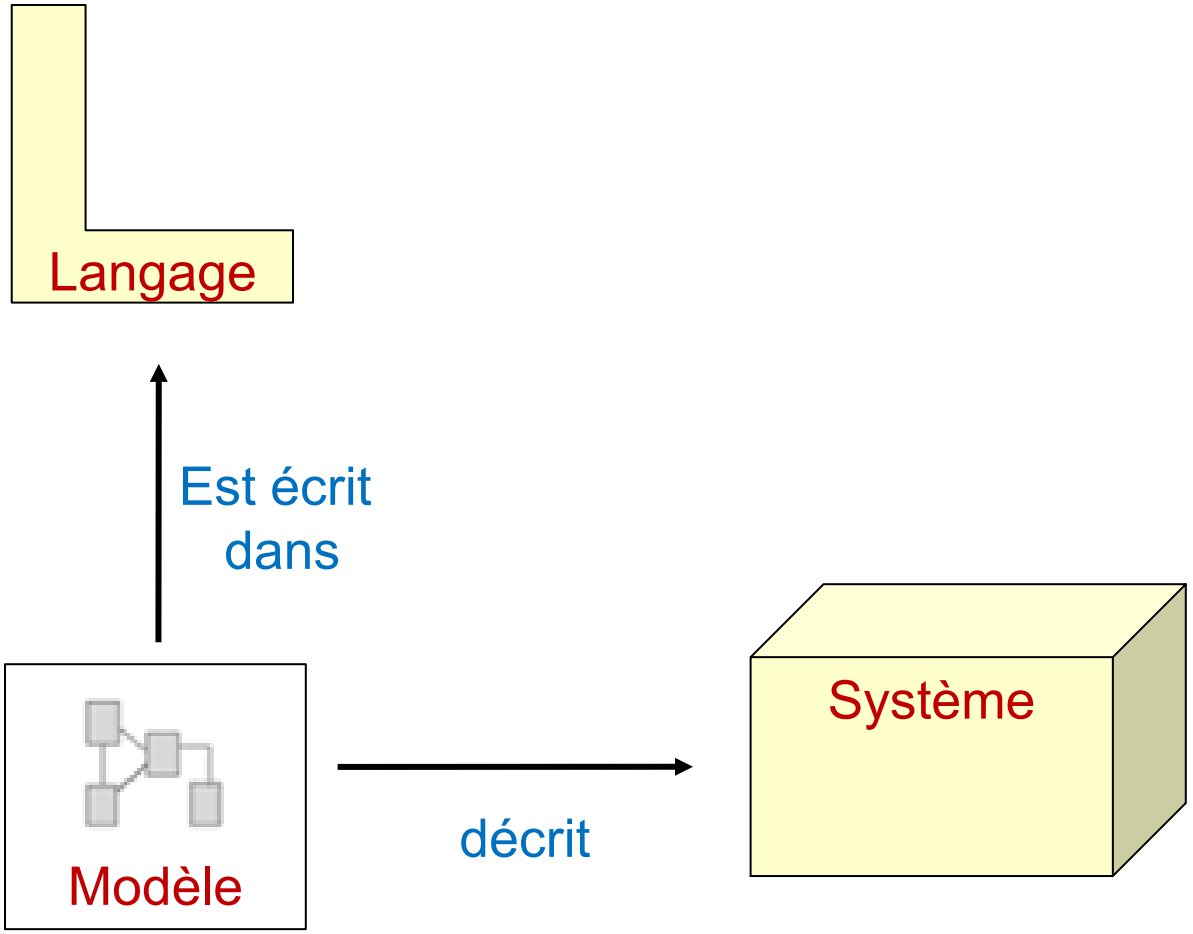
- Notions fondamentales dans le cadre du MDE
  - Modèle
  - Méta-modèle
  - Transformation de modèle
- Nous allons donc les définir précisément

Model Driven Engineering: Définitions

- Un **modèle** est une **description**, une **spécification** partielle d'un système
  - Abstraction de ce qui est intéressant pour un contexte et dans un but donné
- But d'un modèle
  - Faciliter la compréhension d'un système
  - Simuler le fonctionnement d'un système
- Exemples
  - Modèle économique
  - Modèle démographique
  - Modèle météorologique

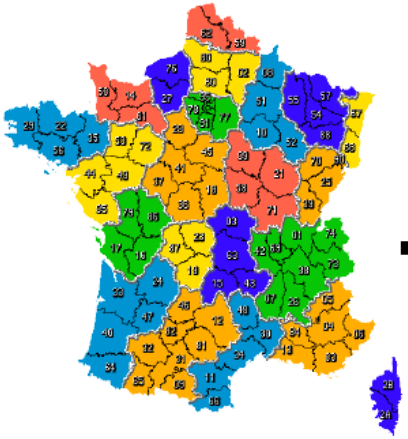
Modèle

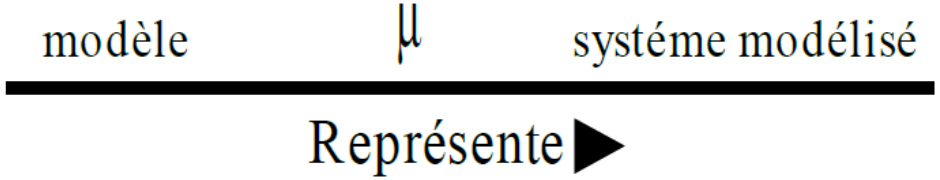
Qu'est ce qu'un modèle dans MDA?



Modèle

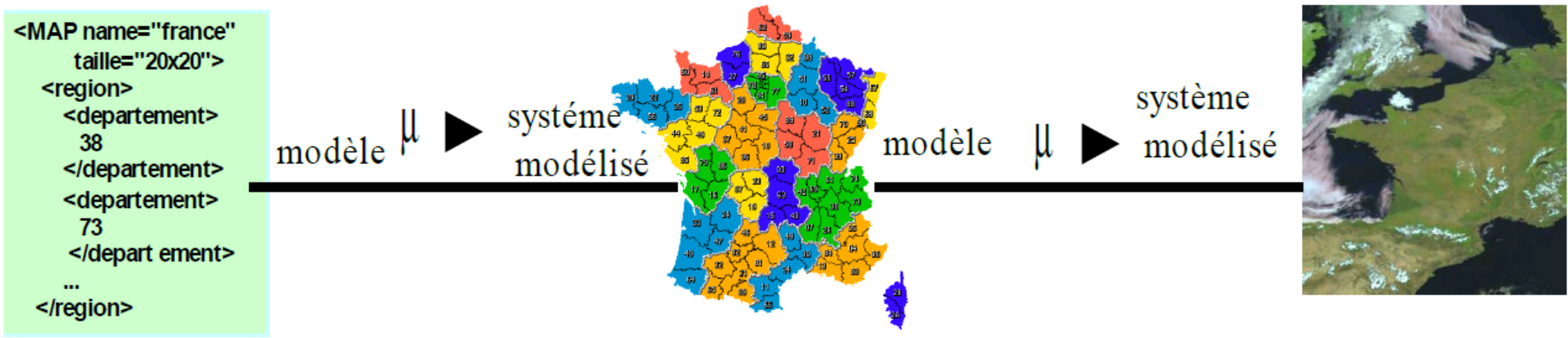
- Différence entre **spécification** et **description ?**
  - **Spécification:** pour un système à construire
  - **Description:** pour un système existant
- Relation entre un système et un modèle
  - **RepresentationDe** (notée **μ** par exemple)





Modèle

- Un modèle représente un système modélisé
- Un modèle peut aussi avoir le rôle de système modélisé dans une autre relation de représentation



- Le **modèle XML** (à gauche) de la carte de la France administrative est un modèle de la France « réelle »

Modèle

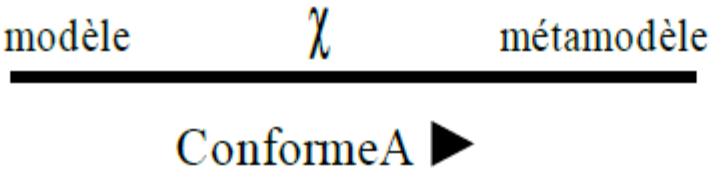
- Un modèle est écrit dans un langage qui peut être
  - **Non ou peu formalisé**, la langue naturelle
    - Le français, un dessin ...
  - **Formel et bien défini, non ambigu**
    - Syntaxe, grammaire, sémantique
    - On parle de **méta-modèle** pour ce type de langage de modèle
- Pour les modèles définis dans un langage bien précis
  - **Relation de conformité**
    - Un modèle est conforme à son **méta-modèle**
    - Relation EstConformeA (notée x)

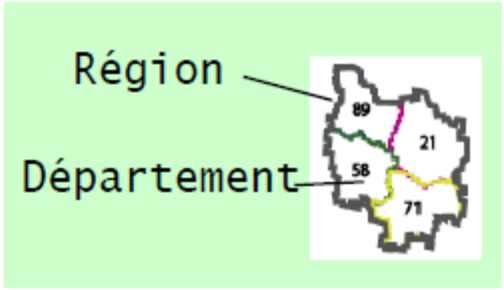
Méta-Modèle

- Un modèle définit quels éléments peuvent exister dans un système.
- Un langage définit les éléments qui peuvent être utilisés dans un modèle.
  - Par exemple, le langage UML définit les concepts de: «**classe**», «**état**», «**attribut** », « **association** », «**package**», et ainsi de suite.
  - Nous pouvons décrire un **langage** par un **modèle** qui contient les éléments qui peuvent être utilisés dans ce langage.
  - Ce modèle qui décrit le langage est nommé **méta-modèle**.

Méta-Modèle

- Un **modèle** est conforme à son **méta-modèle**



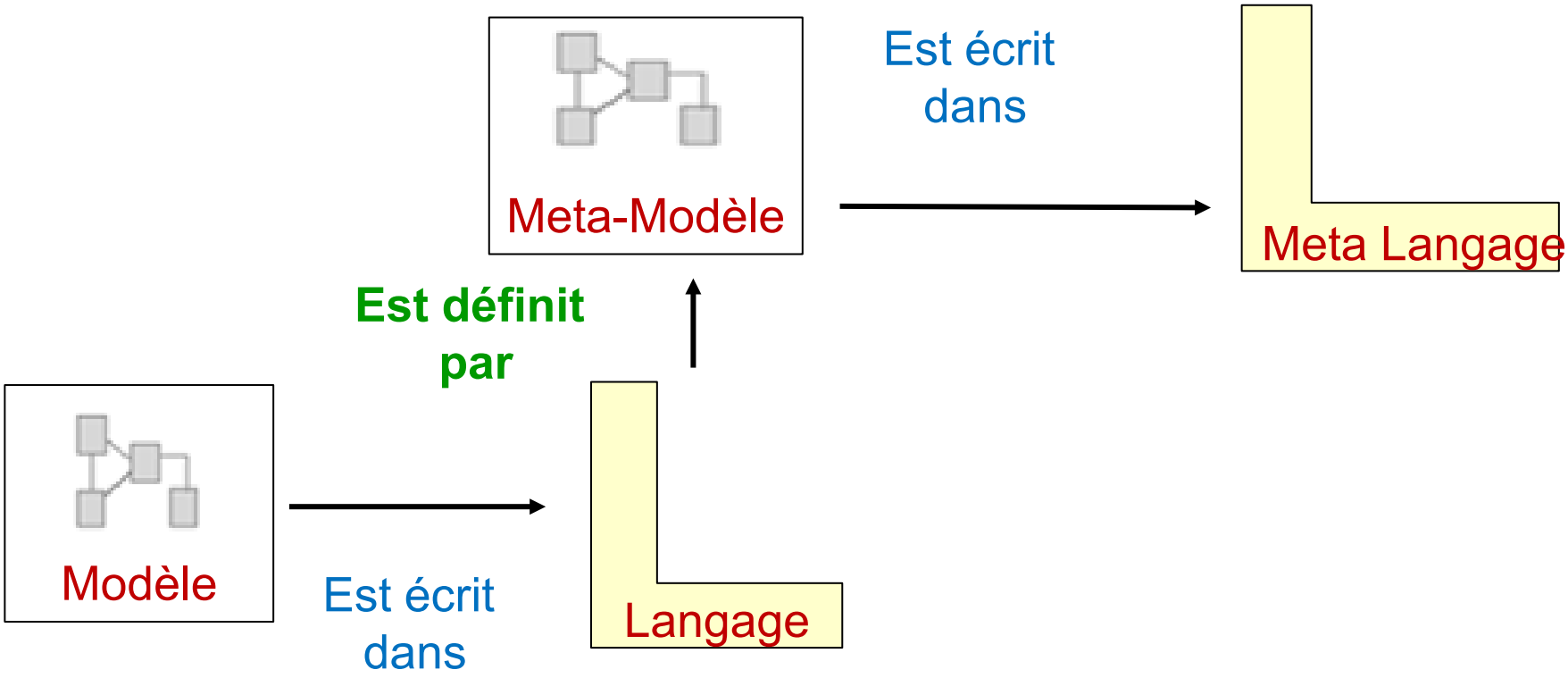


Méta-Modèle

- Cette relation de conformité est essentielle
  - Une **base** du MDE pour développer les outils capables de manipuler des modèles
- Mais pas nouvelle
  - Un texte écrit est conforme à une orthographe et une grammaire
  - Un programme Java est conforme à la syntaxe et la grammaire du langage Java
  - Un fichier XML est conforme à sa DTD
  - Une carte doit être conforme à une légende
  - Un modèle UML est conforme au méta-modèle UML

Modèles, langages, méta-models et méta-langages

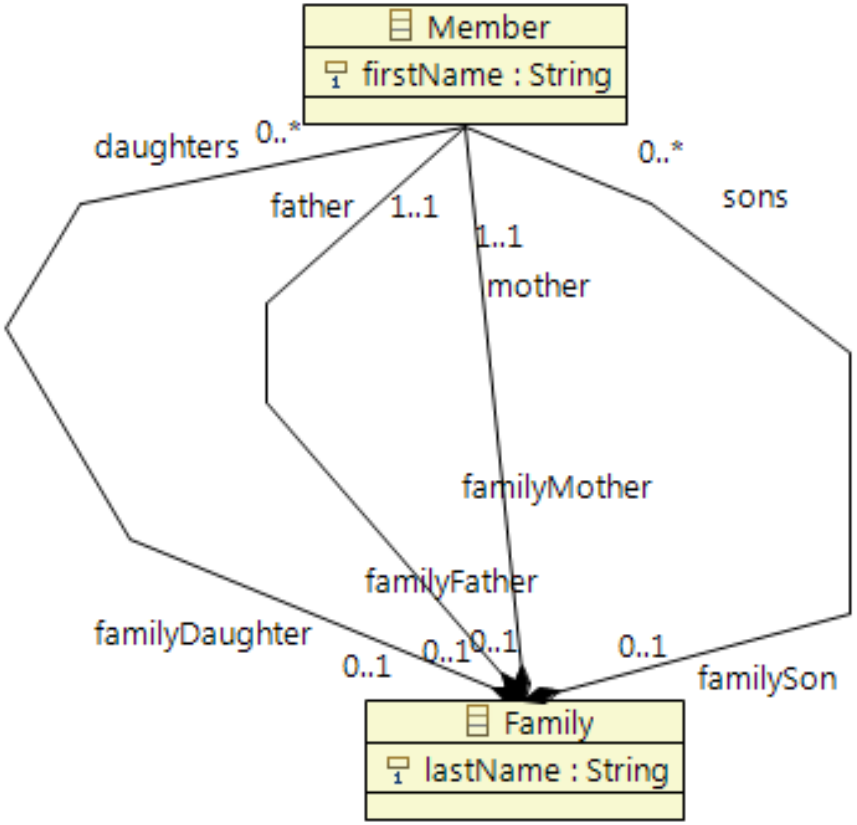
- Un méta-modèle définit le langage pour définir des modèles
- Un méta-modèle n'est donc pas un langage
- Un méta-modèle lui-même doit être écrit dans un langage bien défini. (nommé: Meta Langage).



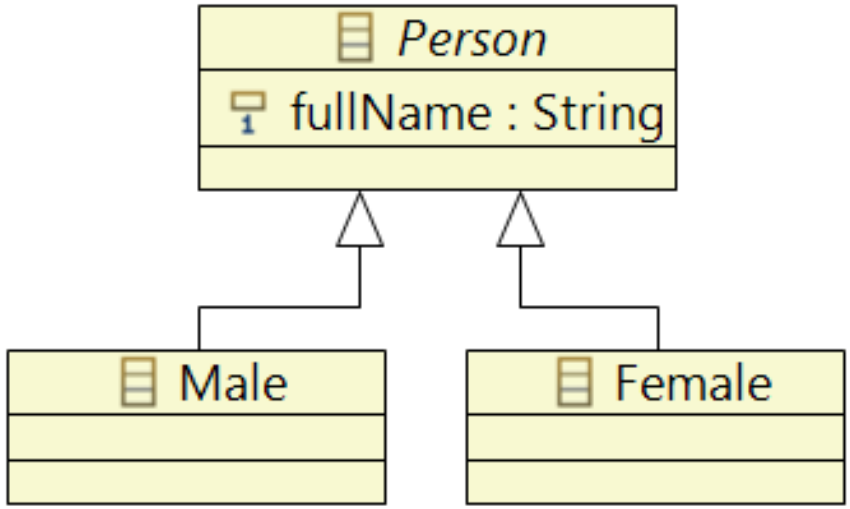
- Le **métalangage** est défini par un **méta-modèle** qui est écrit dans un autre métalangage.

Example de Meta-modèle:

Méta-modèle 1



Méta-modèle 2



Exemple d'instances conforment à ces méta-modèles

Méta-modèle 1

Conforme à

```
<xmi:XMI xmi:version="2.0" xmlns:xmi=« …»>
  <Family lastName="Mansouri">
    <father firstName="Yazid"/>
    <mother firstName="Halima"/>
    <sons firstName="Soufiane"/>
    <daughters firstName="Fatima"/>
  </Family>
  <Family lastName="Yahia">
    <father firstName="Antar"/>
    <mother firstName="Nassima"/>
    <sons firstName="Ahmed"/>
</Family>
</xmi:XMI>
```

Méta-modèle 2

Conforme à

```
<xmi:XMI xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI">
  <Male fullName=" Yazid Mansouri "/>
  <Male fullName="Soufiane Mansouri "/>
  <Male fullName=" Antar Yahia "/>
  <Male fullName=" Ahmed Yahia "/>
 <Female fullName=" Halima Mansouri "/>
  <Female fullName="Fatima Mansouri "/>
  <Female fullName="Nassima Yahia "/>
</xmi:XMI>
```


From Families to Persons ??!!

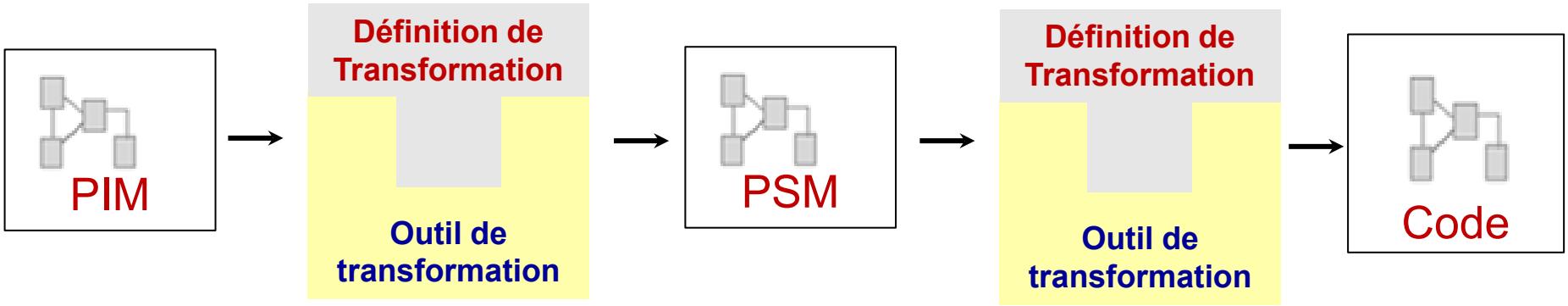
Transformation

Qu'est-ce une transformation?

Une transformation est la génération automatique d'un modèle cible à partir d'un modèle source.

Qu'est-ce qu'un **outil de transformation** en MDA?

- Un logiciel qui permet de transformer :
  - ✓ un **PIM** vers un **PSM**.
  - ✓ Un **PSM** vers un **PSM**
  - ✓ et un **PSM** vers un **code**.



Conclusion

Le MDE est une nouvelle approche pour concevoir des applications

- En plaçant les modèles et surtout les méta-modèles au centre du processus de développement dans un but productif
  - Les **modèles** sont depuis longtemps utilisés **mais** ne couvrait pas l'ensemble du cycle de vie du logiciel
  - Les **méta-modèles** existent aussi depuis longtemps (grammaires, DTD XML, ...) mais peu utilisés en modélisation « à la UML »
  - **Nouvelle vision** autour de notions déjà connues : le méta-modèle devient le point clé de la modélisation
    - Automatisation de la manipulation des modèles
    - Création de DSL : modélisation dédié à un domaine
  - Automatisation du processus de développement
    - Application de séries de transformations, fusions de modèles
  - Les grands éditeurs de logiciel suivent ce mouvement
    - IBM, Microsoft …

Questions?