

1<sup>ère</sup> année Master GLSD

Département d'Informatique

Université de Biskra

Année 2020/2021

# Ingénierie des Modèles

## Méta-modélisation

Dr. Mohamed Lamine Kerdoudi

Email: [l.kerdoudi@univ-biskra.dz](mailto:l.kerdoudi@univ-biskra.dz)

# Principales normes de OMG

- **MOF** : Meta-Object Facilities
  - Langage de définition de méta-modèles
- **UML** : Unified Modelling Language
  - Langage de modélisation
- **CWM** : Common Warehouse Metamodel
  - Modélisation ressources, données, gestion d'une Entreprise
- **OCL** : Object Constraint Language
  - Langage de contraintes sur modèles
- **XMI** : XML Metadata Interchange
  - Standard pour échanges de modèles et méta-modèles entre outils

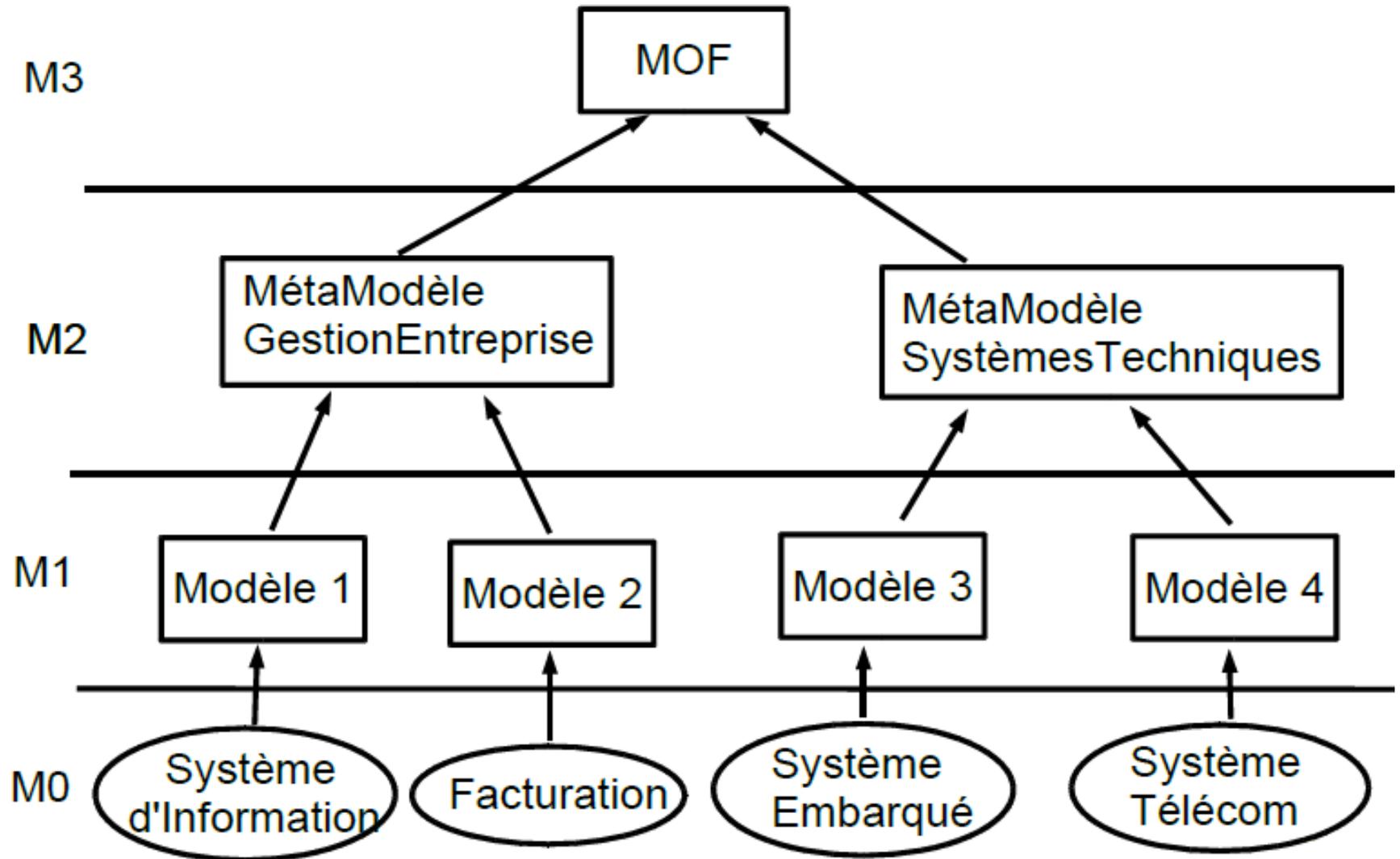
# Normes OMG de modélisation

- Plusieurs de ces normes concernent la définition et l'utilisation de méta-modèles
  - **MOF** : but de la norme
  - **UML** et **CWM** : peuvent être utilisés pour en définir
  - **XMI** : pour échange de (méta-)modèles entre outils
- **MOF**
  - C'est un **méta-méta-modèle**
    - Utilisé pour définir des méta-modèles
  - Définit les concepts de base d'un méta-modèle
    - Entité/classe, relation/association, type de données, référence, package ...
  - Le MOF peut définir le MOF

# Hiérarchie de modélisation à 4 niveaux

- L'OMG définit 4 niveaux de modélisation
  - M0 : système réel, système modélisé
  - M1 : modèle du système réel défini dans un certain langage
  - M2 : méta-modèle définissant ce langage
  - M3 : méta-méta-modèle définissant le méta-modèle
    - Le niveau M3 est le **MOF**
    - Dernier niveau, il est méta-circulaire : il peut se définir lui même
- Le MOF est – pour l'OMG – le méta-méta-modèle unique servant de base à la définition de tous les méta-modèles

# Hiérarchie de modélisation à 4 niveaux



# Hiérarchie de modélisation à 4 niveaux

- Hiérarchie à **4 niveaux** existe en dehors du MOF et d'UML, dans d'autres espaces technologiques que celui de l'OMG
- Exemple: Langage de programmation
  - M0 : l'exécution d'un programme
  - M1 : le programme
  - M2 : la grammaire du langage dans lequel est écrit le programme
  - M3 : le concept de grammaire EBNF:

**Extended Backus-Naur Form** (BNF: est une notation permettant de décrire les règles syntaxiques des langages de programmation.)

# Méta-modélisation UML

- Avec UML, on retrouve également les **4 niveaux**
- **Mais**, le **niveau M3** est défini en UML directement à la place du MOF
- Exemple de **ystème réel** à modéliser (**niveau M0**)

## Spécification

- Une pièce possède 4 murs, 2 fenêtres et une porte
- Un mur possède une porte ou une fenêtre mais pas les 2 à la fois
- Deux **actions** sont associées à une porte ou une fenêtre :  
**ouvrir** et **fermer**
- Si on ouvre une porte ou une fenêtre fermée, elle devient ouverte
- Si on ferme une porte ou une fenêtre ouverte, elle devient fermée

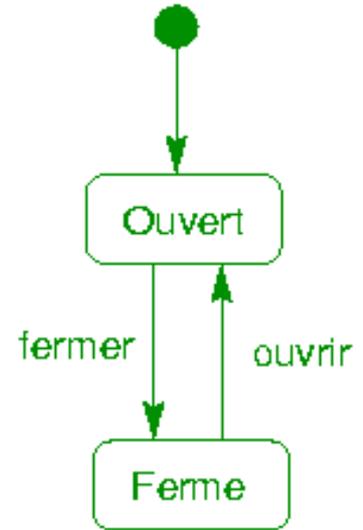
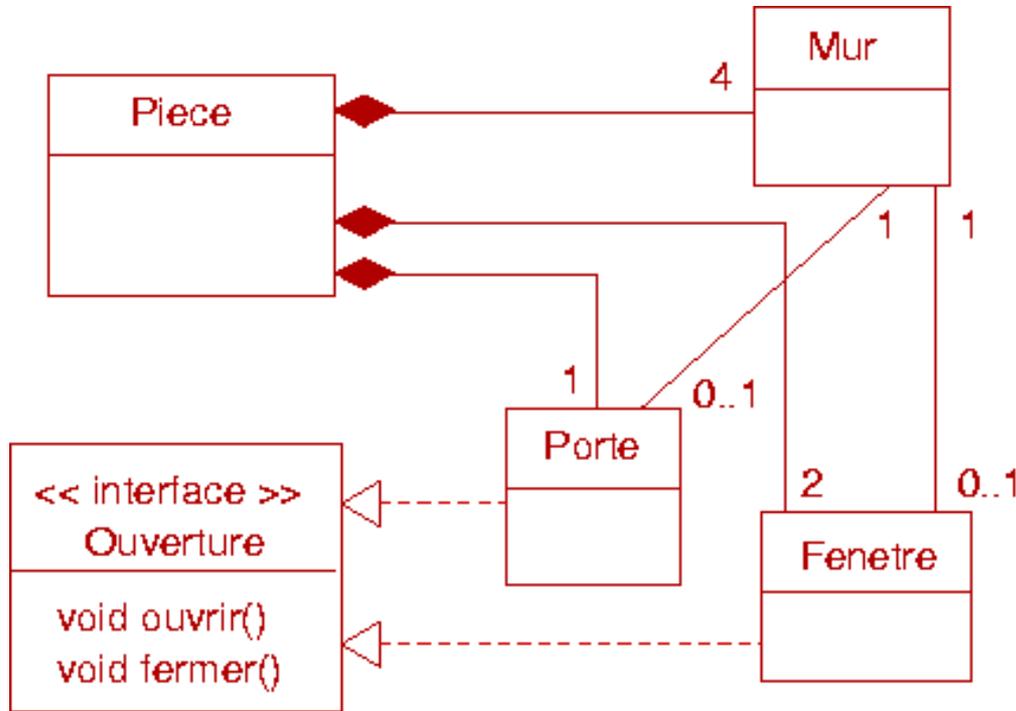
# Méta-modélisation UML

- Pour modéliser ce système, on a besoin de définir 2 diagrammes UML :

## Niveau M1

- Un **diagramme de classe** pour représenter les relations entre les éléments (portes, murs, pièce)
- Un **diagramme d'état** pour spécifier le comportement d'une porte ou d'une fenêtre (ouverte, fermée)
- On peut **abstraire** le comportement des portes et des fenêtres en spécifiant les opérations d'ouverture/fermeture dans une Interface
  - Le diagramme d'état est associé à cette interface
- Il faut également ajouter des contraintes OCL pour préciser les contraintes entre les éléments d'une pièce

# Méta-modélisation UML



*diagramme d'etat associe  
a l'interface Ouverture*

**context** Piece inv:

**mur.fenetre -> size() = 2** -- 2 murs de la pièce ont une fenêtre

**mur.porte -> size() = 1** -- 1 mur de la pièce a une porte

**context** Mur inv:

**fenetre -> union(porte) -> size() <= 1**

-- un mur a soit une fenêtre soit une porte (soit rien)

# Métab-modélisation UML

- Les 2 diagrammes de ce modèle de **niveau M1** sont des diagrammes UML valides
- Les contraintes sur les **éléments des diagrammes** UML et **leurs relations** sont définies dans le méta-modèle UML : **niveau M2**

➤ Un diagramme UML (de classes, d'états ...) doit être conforme au **méta-modèle UML**

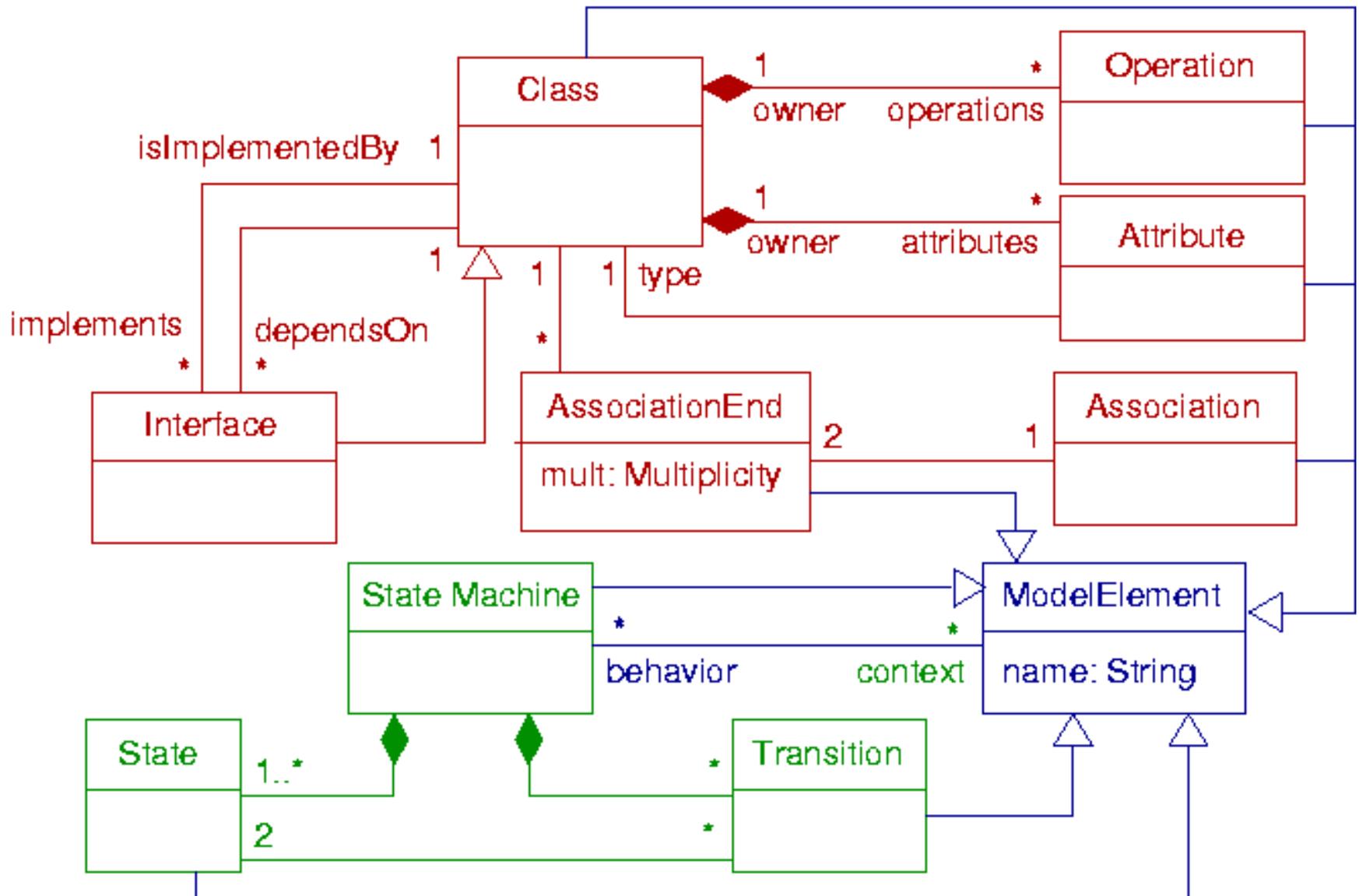
- **Méta-modèle UML**

- Est un diagramme de classe spécifiant la structure de tous types de diagrammes UML

- Diagramme de classe car c'est le diagramme UML permettant de définir **des éléments/concepts** (via des classes) et **leurs relations** (via des associations)

- Avec des **contraintes OCL** pour une spécification précise

# M2 : Méta-modèle UML (très simplifié)





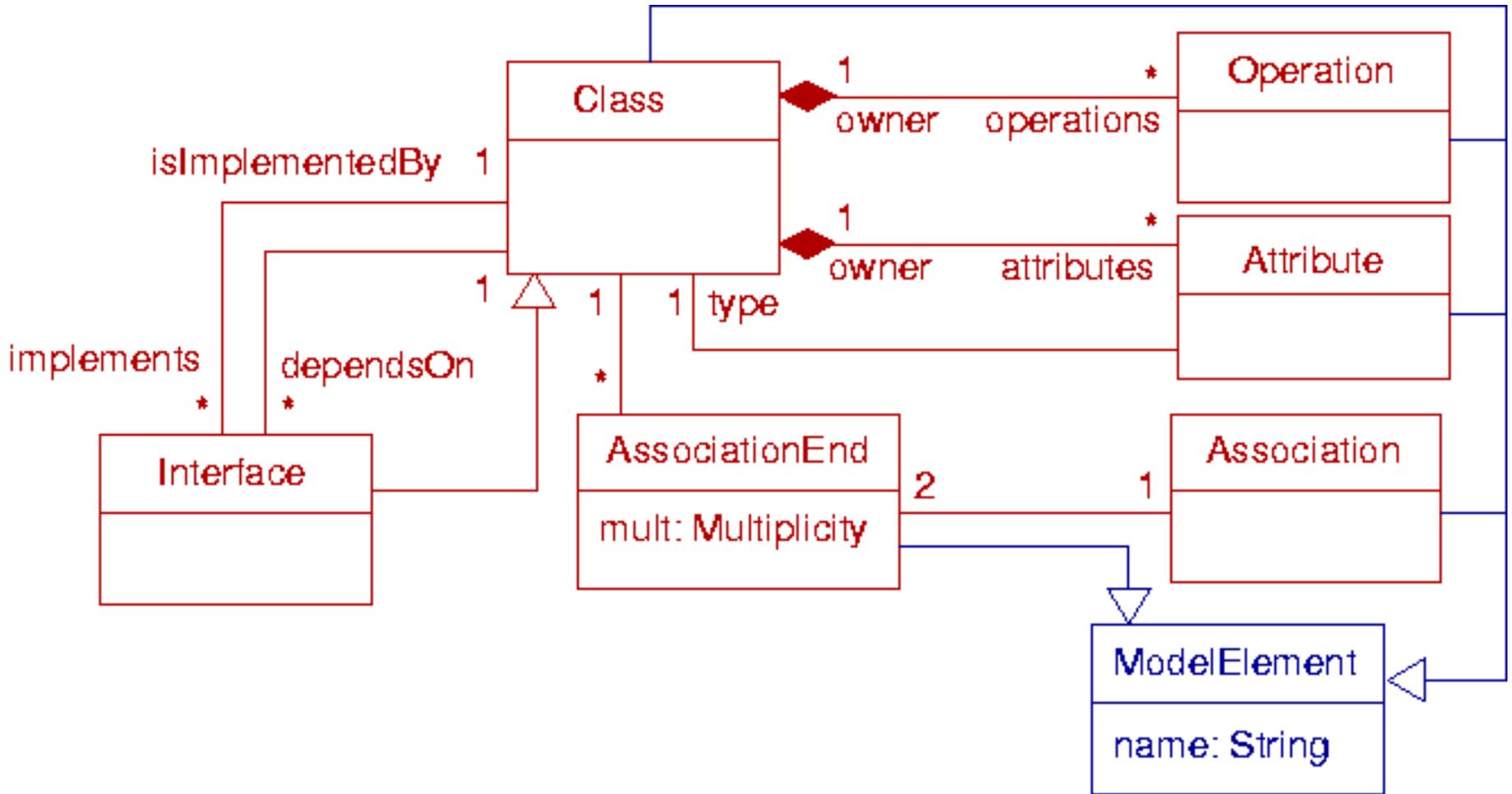


# Méta-modélisation UML

- Le **méta-modèle** UML doit aussi être précisément défini
    - Il doit être conforme à un **méta-modèle**
    - C'est le méta-méta-modèle UML
- Niveau M3

→ **Méta-méta-modèle UML** = copie partielle du « **méta-modèle de UML** »

# M3 : Méta-méta-modèle UML (simplifié)



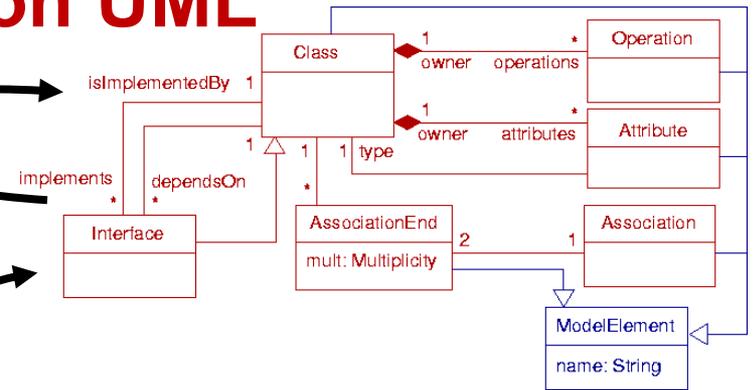
# Méta-modélisation UML

- **Méta-méta-modèle** UML doit aussi être clairement défini
  - Il doit être conforme à un **méta-modèle**
- Le méta-méta-modèle UML peut donc se définir lui-même
  - Méta-circulaire
  - Pas besoin de niveau méta supplémentaire

# Méta-modélisation UML

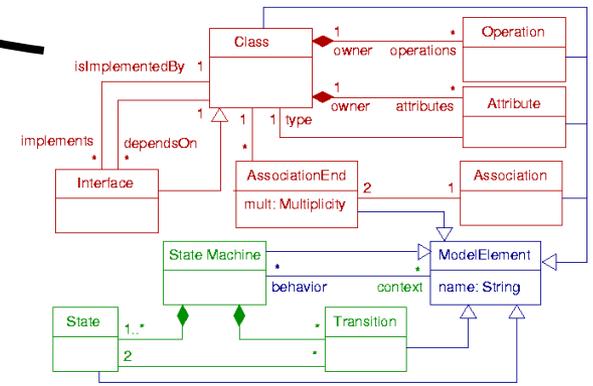
Niveau M3

conforme à



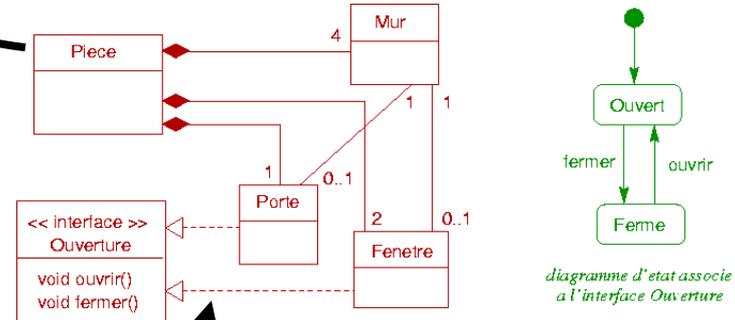
Niveau M2

conforme à



Niveau M1

conforme à



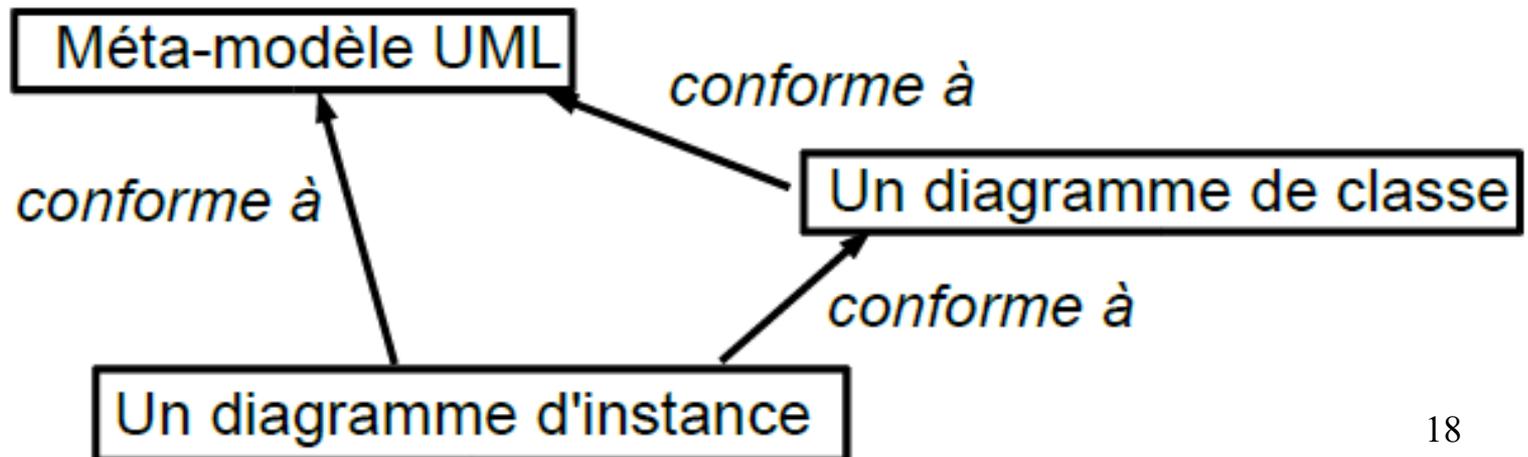
Niveau M0

conforme à



# Diagrammes d'instance UML

- Un **diagramme d'instance** est particulier car
  - Doit être conforme au méta-modèle UML
    - Qui définit la structure générale des diagrammes d'instances
  - Doit aussi être conforme à un diagramme de classe
    - Ce diagramme de classe est un méta-modèle pour le diagramme d'instance
    - Diagramme de classe est conforme également au méta-modèle UML



# Hiérarchie de modélisation

- **Architecture à 4 niveaux**
  - Conceptuellement **pertinente**
- **En pratique**
  - ***Certains niveaux sont difficiles*** à placer les uns par rapport aux autres
    - ***Cas du diagramme d'instance***
      - Diagramme d'instance de l'application : **niveau M1**
      - Méta-modèle UML : **niveau M2**
      - Diagramme de classe de l'application : **niveau M1 ou M2 ?**
      - **M1** normalement car, il modélise l'application et il est conforme au méta-modèle de niveau **M2**
      - Mais devrait être **M2** rapport au diagramme d'instance qui est de niveau **M1**
    - ***Conclusion***
      - Pas toujours facile ni forcément pertinent de chercher à placer absolument les modèles à tel ou tel niveau
      - **L'important** est de **savoir quel rôle (modèle / méta-modèle)** joue un modèle dans une **relation de conformité**

# Syntaxe

- Un langage est défini par un **méta-modèle**
- Un langage possède une **syntaxe**
  - Définit comment représenter chaque type **d'élément** d'un modèle
  - Élément **d'un modèle** = instance d'un **méta-élément**
- **Syntaxe textuelle**
  - Ensemble de mots-clé et de mots respectant des contraintes défini selon des règles précises
    - Notions de syntaxe et de grammaire dans les langages
- Exemple : pour le **langage Java** :

```
public class MaClasse implements MonInterface { ... }
```

→ Grammaire Java pour la déclaration d'une classe :

```
class_declaration ::= { modifier } "class" identifier  
[ "extends" class_name ] [ "implements" interface_name  
{ "," interface_name } ] "{" { field_declaration } "
```

# Syntaxe

## ■ Syntaxe graphique

- Notation graphique, chaque type d'élément a une forme graphique particulière

- **Exemple** : relations entre classes/interfaces dans les diagrammes de classe UML

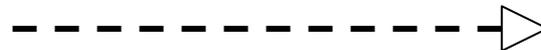
- Trait normal : **association**



- Flèche, trait pointillé : **dépendance**



- Flèche en forme de triangle, trait en pointillé : **implémentation**



- Flèche en forme de triangle, trait plein : **spécialisation**



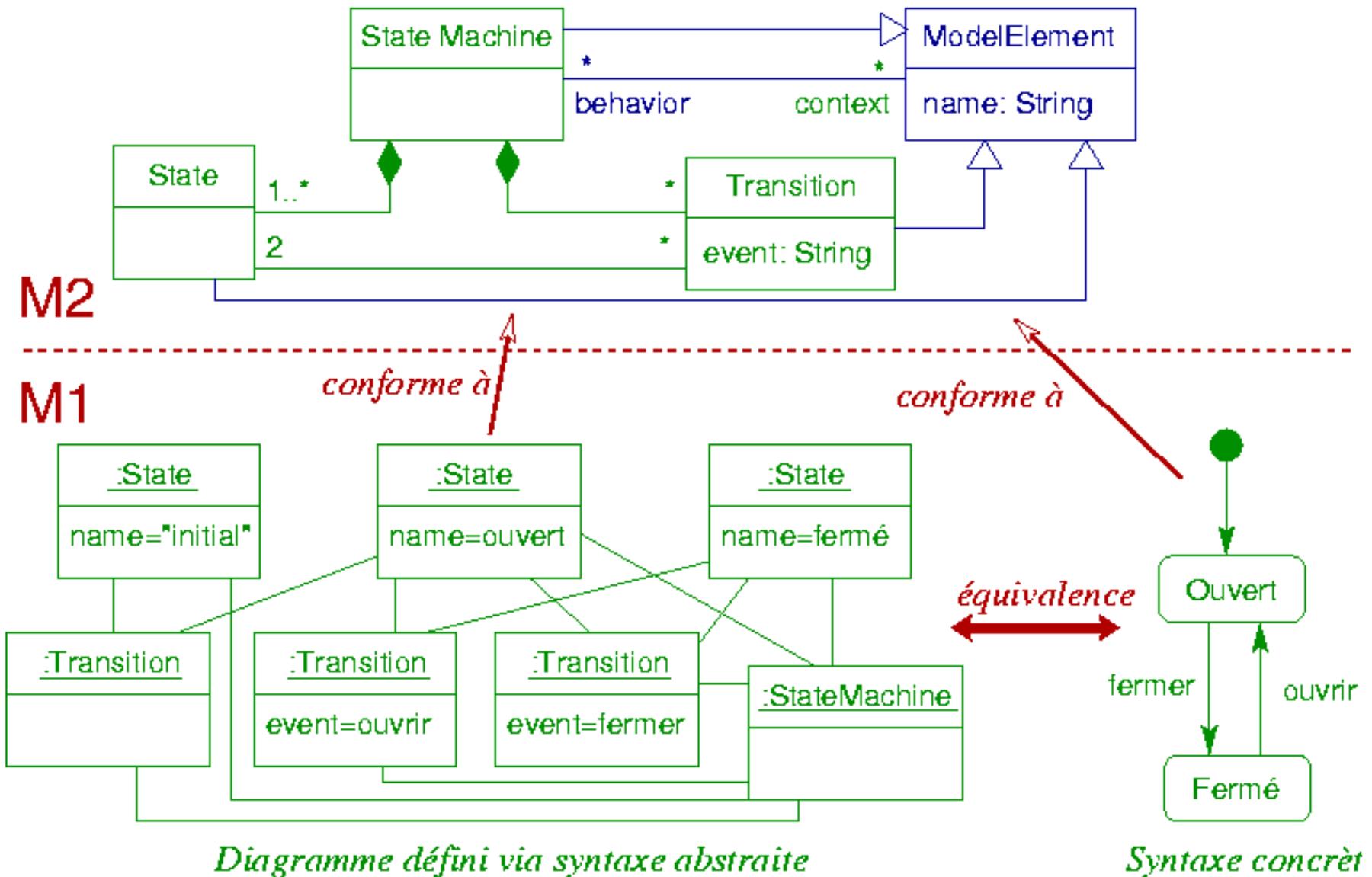
# Syntaxe

- **Syntaxe abstraite/concrète**
  - **Abstraite**
    - Les éléments et leurs relations **sans** une notation spécialisée
    - Correspond à ce qui est défini au niveau du méta-modèle, à des instances de méta-éléments
  - **Concrète**
    - **Syntaxe graphique** ou **textuelle** définie pour un type de modèle
    - Plusieurs **syntaxes concrètes** possibles pour une même syntaxe abstraite
- Un modèle peut être défini via n'importe quelle syntaxe
  1. L'abstraite
  2. Une des concrètes

# Syntaxe

- **Exemple** de la modélisation de la **pièce**
  - **Syntaxe concrète**
    - 2 diagrammes UML (**classes et états**) avec syntaxes graphiques spécifiques à ces types de diagrammes
  - **Via la syntaxe abstraite**
    - Diagramme d'instance (conforme au méta-modèle) précisant les instances particulières de classes, d'associations, d'états...
  - **Pour la partie diagramme d'états**
    - Diagramme défini via syntaxe **concrète** : diagramme d'états de l'exemple
    - Diagramme défini via syntaxe **abstraite** : diagramme d'instances conforme au méta-modèle UML

# Syntaxe : Exemple diagramme état



**Questions ?**