Ingénierie du Logiciel
Master 1 Informatique – 4I502

Cours 10 :
Modèles
Méta-Modèles
Méta-Méta-Modèles
…?

Yann Thierry-Mieg
Yann.Thierry-Mieg@lip6.fr

Méta-Modélisation

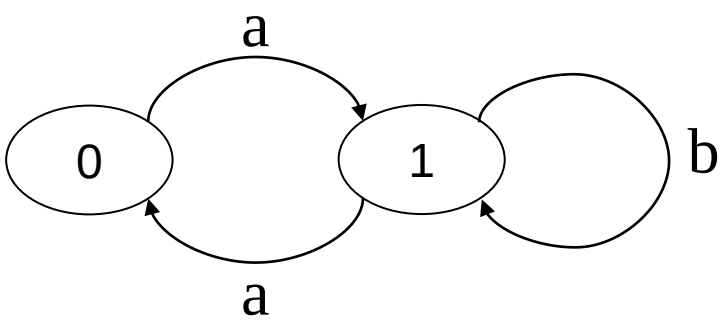
2000+ le millénaire du modèle ?

- Naissance d'UML
  - Niveaux d'abstraction divers, points de vue divers
  - Agregation de langages / concepts
    - Booch -> message sequence charts, diagrammes de sequence,
    - Harel -> state charts, State machine,
    - Rumbaugh -> OMT, diagrammes de classe,
    - Jacobson -> OOSE, use case ….
- Standardisation : 1998 1.0 -> 2005 2.2
  - Assemblage de *notations*, la méthode à instancier est plutôt décrite dans le RUP
  - Consortium industriel important => OMG
  - Comment construire un format de stockage et d'échange pour les modèles ?
    - Suffisamment riche pour couvrir tout UML

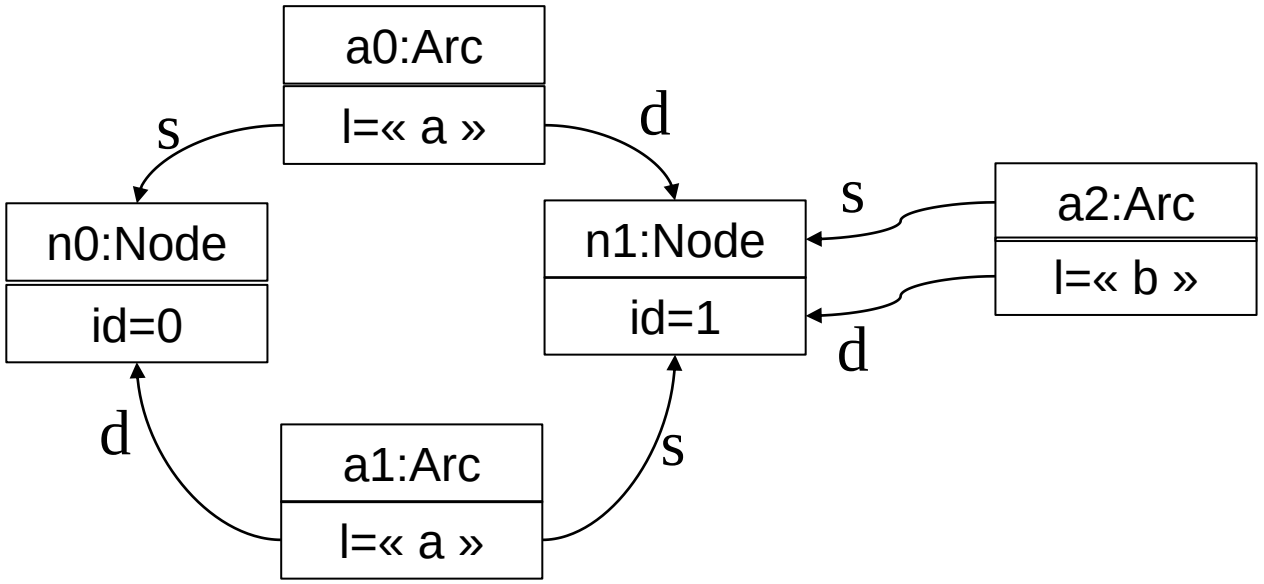
Un format d'échange pour UML ?

- UML c'est :
  - 14 diagrammes,
  - Plusieurs représentations alternatives pour les mêmes concepts
  - Aspect graphique important et configurable
  - Des points de *variation sémantique*
  - Des niveaux de « compliance » des outils UML
- Standardiser le format d'un modèle UML =>
  - Spécifier tout ce qui pourrait être représenté en UML
  - Spécifier les liens entre les concepts des divers diagrammes
  - Spécifier une représentation concrète/syntaxique des modèles UML
- La méta-modélisation
  - Pour faire face à ce défi, définition d'une infrastructure générique pour décrire les modèles et les manipuler
  - ✓ Effet secondaire de la standardisation d'UML, mais indépendant du standard: des outils industriels pour décrire les modèles *mais aussi les langages*.

Exemple Graphe (M0)

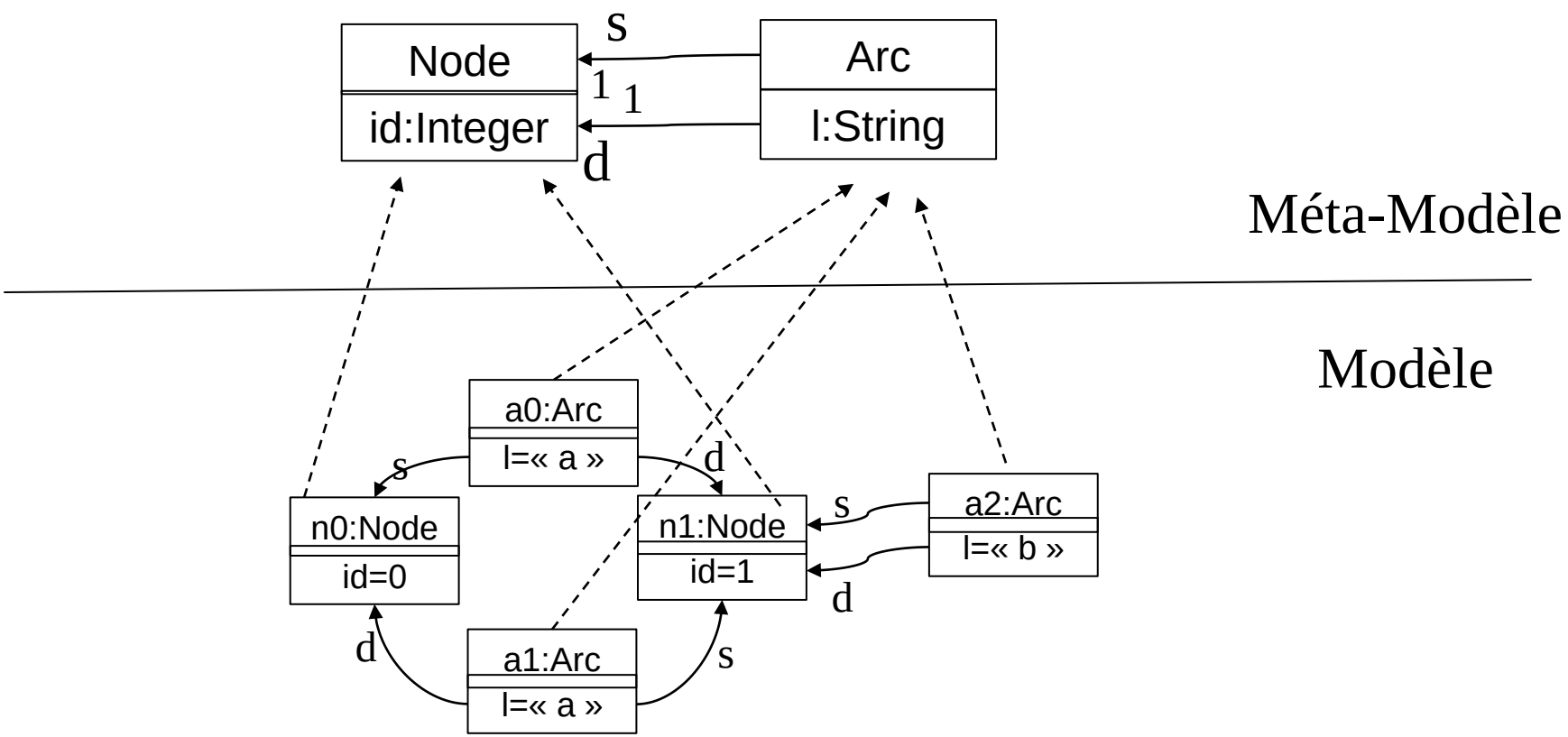


Un graphe = Sommets numérotés + Arcs orientés porteur d'une lettre



La même information en orienté objet : instances de classes

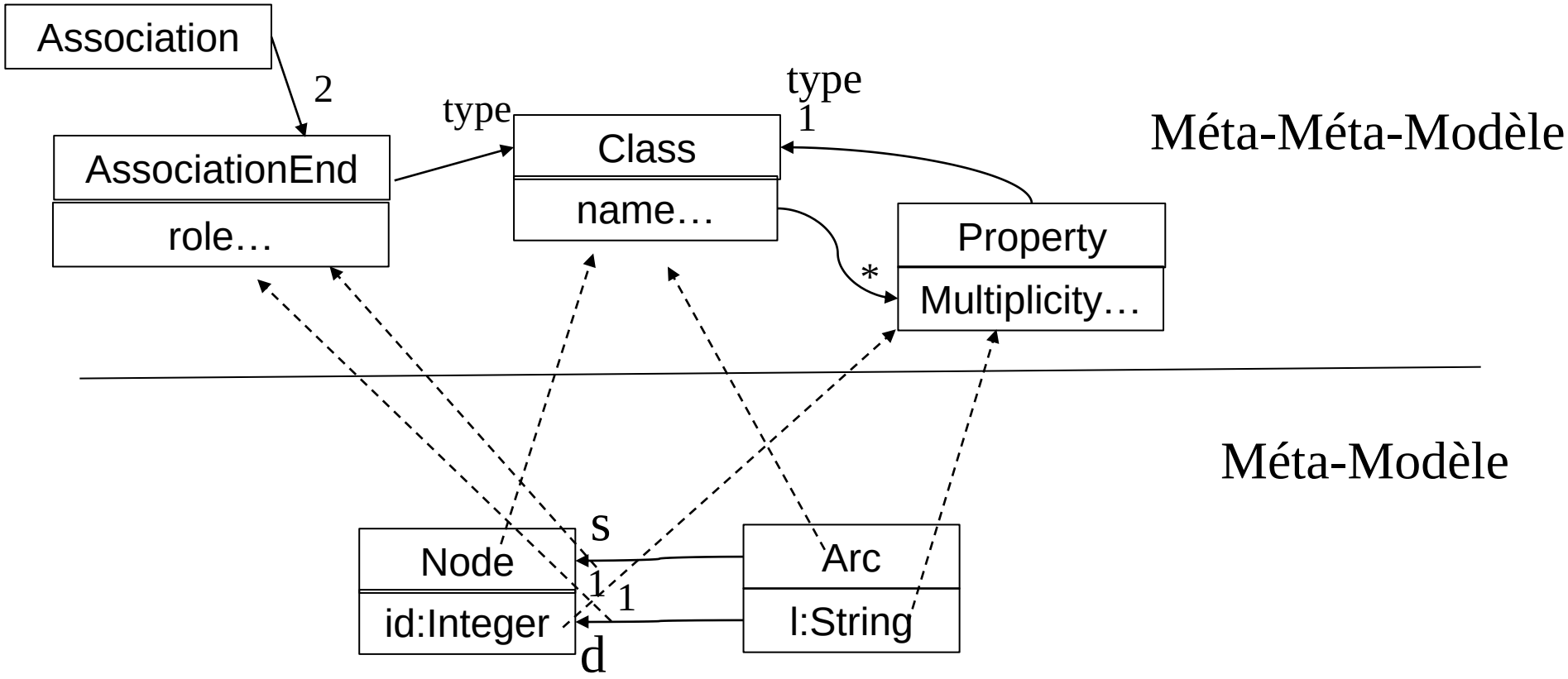
Méta-modèle



Tout graphe peut être vu comme une instanciation de ce *méta-modèle* représenté ici par un diagramme de classe

- ✓ Le métamodèle définit les concepts ou la grammaire abstraite d'un langage.

Méta-Méta-Modèle



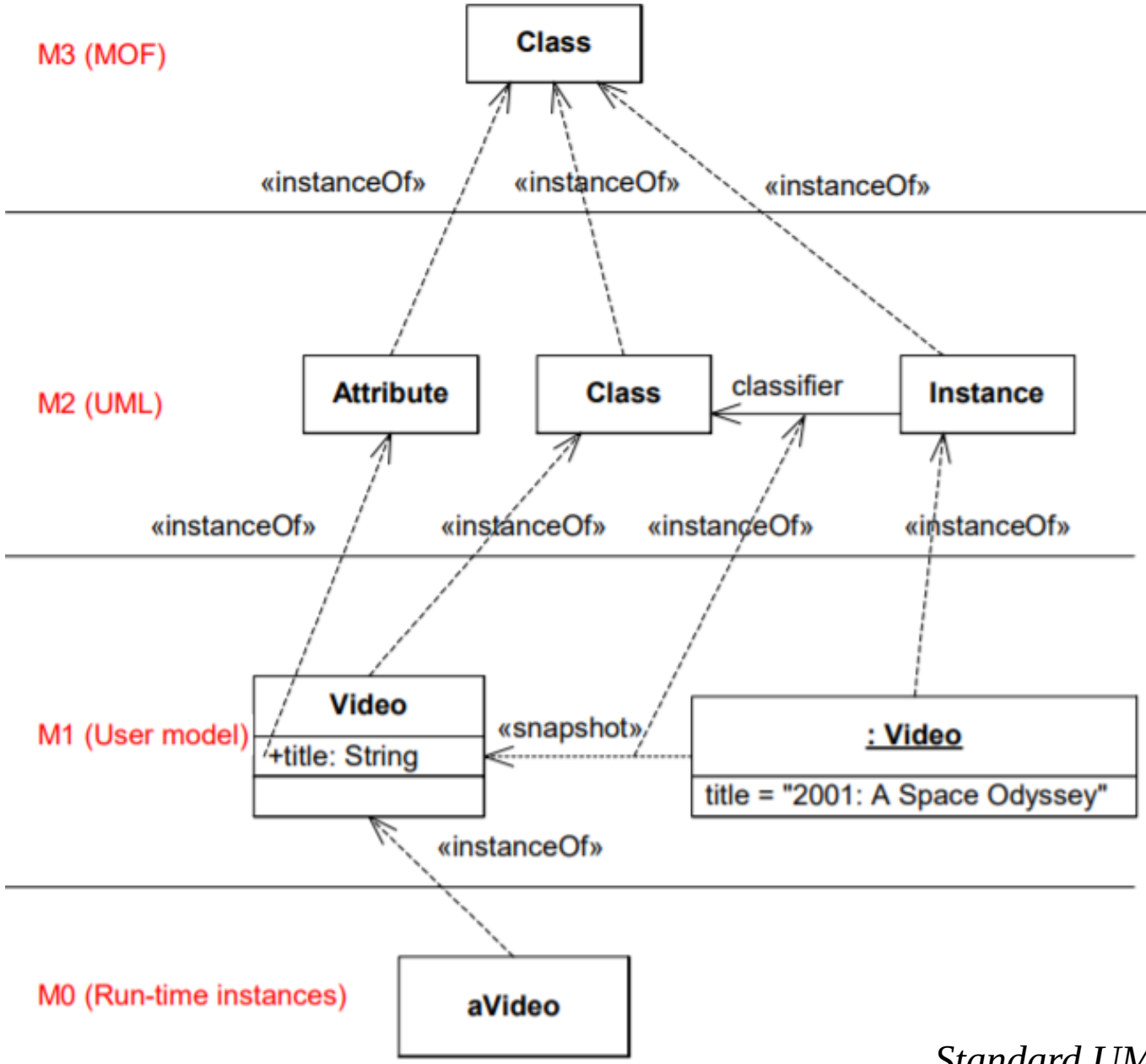
Les méta-modèles sont eux-mêmes des modèles, ils ont leur propre méta-modèle, c'est un méta-méta-modèle

Le niveau méta-méta est *auto-descriptif*.

Les niveaux méta

- Niveaux d'instanciation
  - Je fais le café : M0
  - J'explique comment faire du café : M1
  - J'explique comment expliquer une procédure : M2
  - J'explique comment expliquer : M3
  - Il n'y a pas de niveau au-dessus, si je sais « expliquer comment expliquer » je peux expliquer n'importe quelle idée !

Les niveaux d'abstraction



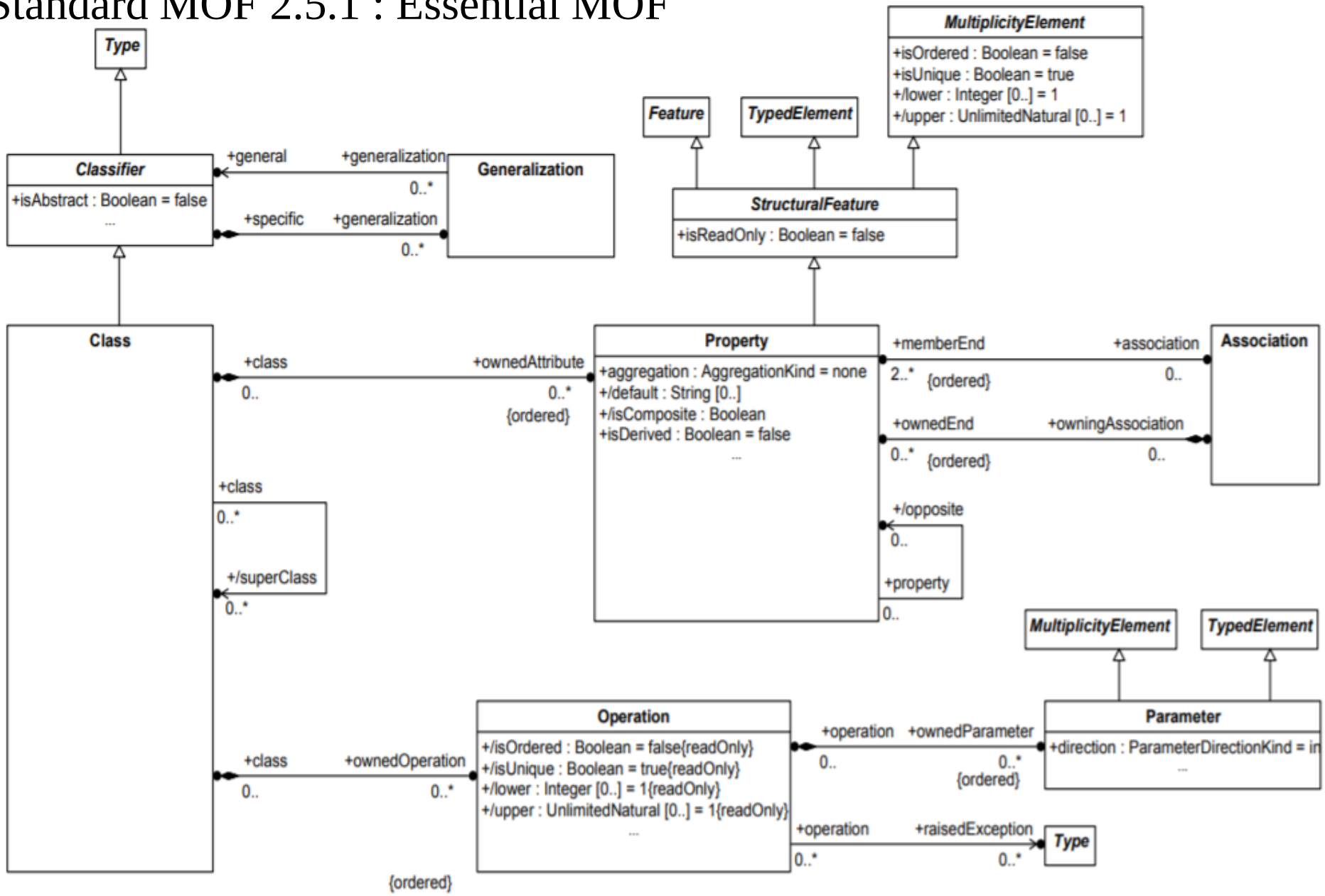
Les 4 couches

On préfère parler d'un seul niveau de méta à la fois dans le standard UML 2.5.

Standard UML 2.2. infrastructure

Le cœur du MOF

Standard MOF 2.5.1 : Essential MOF



LE Méta-Méta-Modèle

- Un seul méta-méta-modèle commun à tous les langages
- Concepts simples :
  - Nœuds typés
  - Propriétés (attributs du noeud) typé + multiplicité
  - Associations : Références/agrégation distinguées des compositions
  - Héritage possible, notion de sous-typage
  - Types de base limités : String, Int, Bool…
- Ces concepts suffisent pour décrire le méta-modèle d'un langage
  - Tout modèle exprimé dans ce langage en est une instance

Meta Object Facility : MOF

- Effets immédiats :
  - Standardisation OMG séparée d'UML : MOF
    - Meta Object Facility
    - Implantation de référence EMF : Eclipse Modeling Framework
  - Cette technologie est au départ développée pour UML
    - Mais portée bien plus importante
    - Un cadre général pour définir des langages arbitraires, pas juste UML

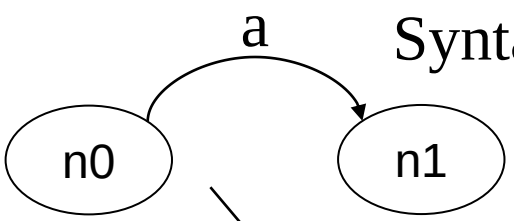
Ingénierie des Modèles

- Modèle
  - Ensemble d'informations structurées selon une certaine grammaire abstraite, son méta-modèle
  - Un modèle est donc vu comme un graphe complet (cycles possibles). Plus riche qu'une vision fichier ou AST. Penser plutôt XML avec des crossref.
- Méta-modèle
  - Le méta-modèle est un modèle particulier, qui décrit un langage.
  - C'est une description *technique* de sa *syntaxe abstraite*. Un MM ne donne pas de définition sémantique.
  - Son méta-modèle est le niveau MOF/EMF
  - Un méta-modèle se représente parfois à l'aide de diagrammes similaires à des diagrammes de classe MAIS ce n'est pas une technologie liée à UML en pratique
- Modèle et Méta-modèle sont homogènes
  - Chacun à leur niveau instancie un méta-modèle

EMF/MOF en pratique

```
graph G (
  n1;
  n2;
  n1->n2 [label="a"];
)
```

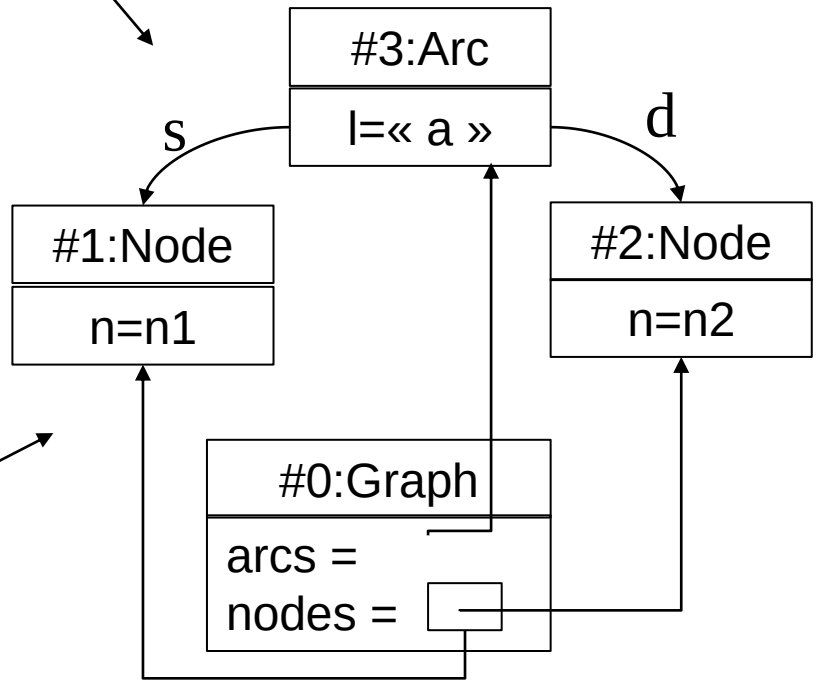
Syntaxe textuelle



Syntaxe graphique

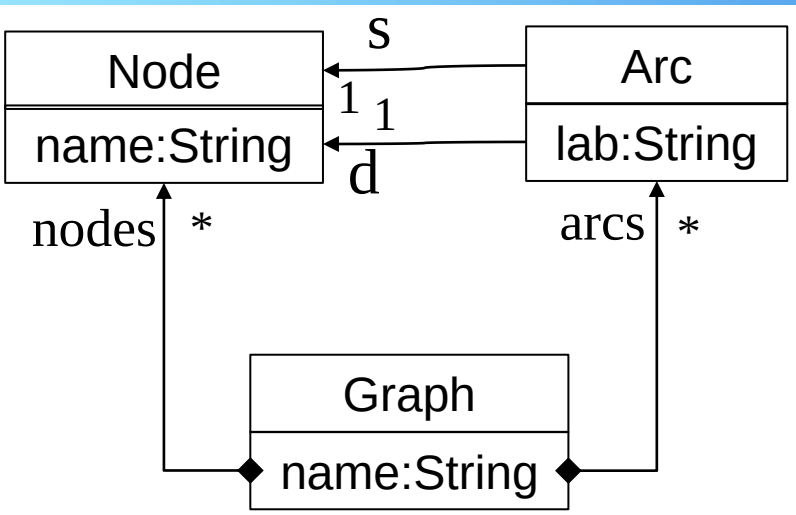
```
<graph name="G" #0>
  <nodes>
        <node name="n1" #1 />
        <node name="n2" #2 />
  </nodes>
  <arcs>
        <arc src=#1 dst=#2 label="a" #3 />
  </arcs>
</graph>
```

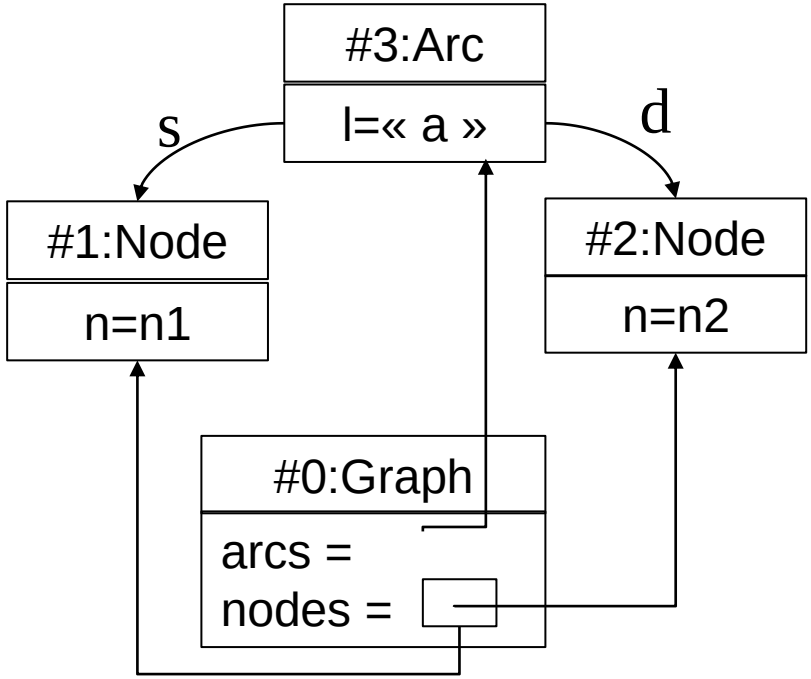
Syntaxe XML



modèle

Méta-Modèle





modèle

Le MM influe sur la représentation

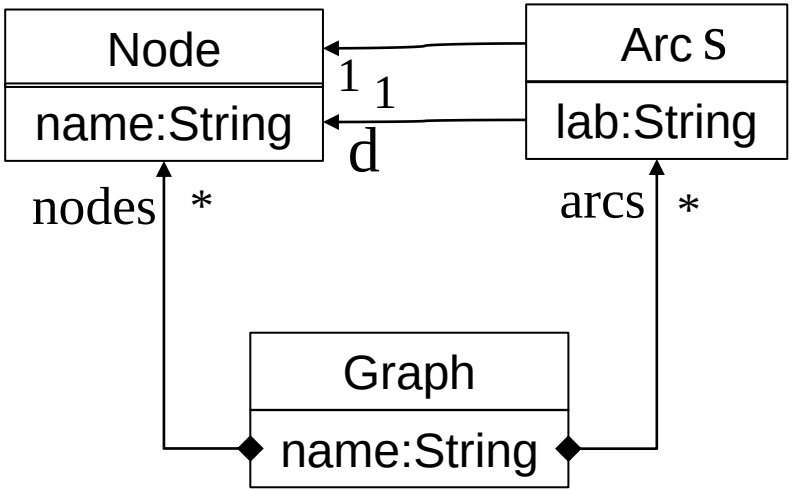
- Accent sur la différence entre composition et aggregation

Aggregation/Composition

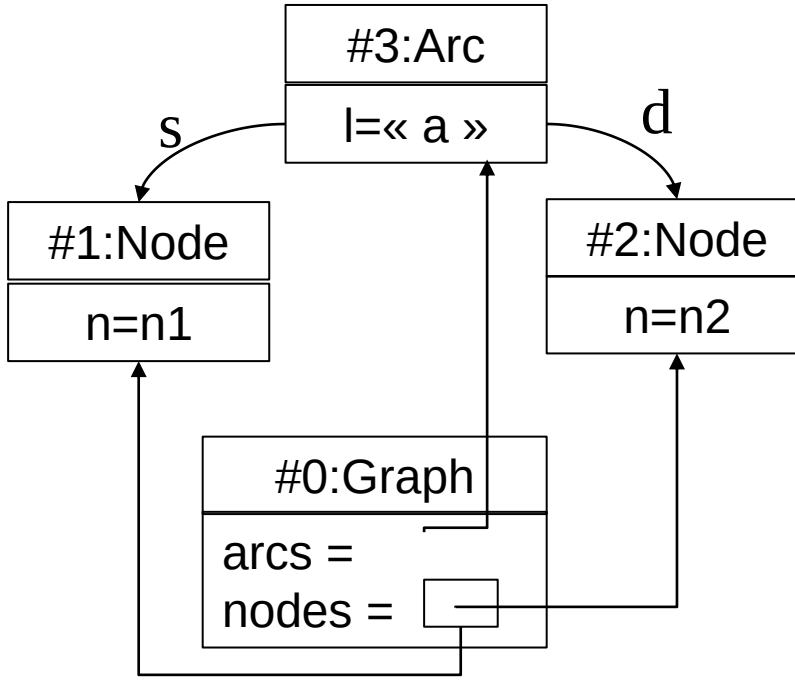
- Un catalogue **contient** des produits, composition
  - Il en détient la responsabilité
- Une recherche **référence** des produits, aggregation
  - Si elle est détruite, les références existent encore dans le catalogue

- Un méta-modèle est muni d'une racine
  - Tous les noeuds du graphe doivent être accessibles depuis la racine par un unique chemin de compositions

Méta-Modèle




```
<graph name="G" #0>
 <nodes>
        <node name="n1" #1 />
        <node name="n2" #2 />
 </nodes>
 <arcs>
        <arc src=#1 dst=#2 label="a" #3 />
 </arcs>
</graph>
```



modèle

Le MM influe sur la représentation

Et si c'était plutôt Node qui contenait les arcs ?

Méta-modèle conclusion

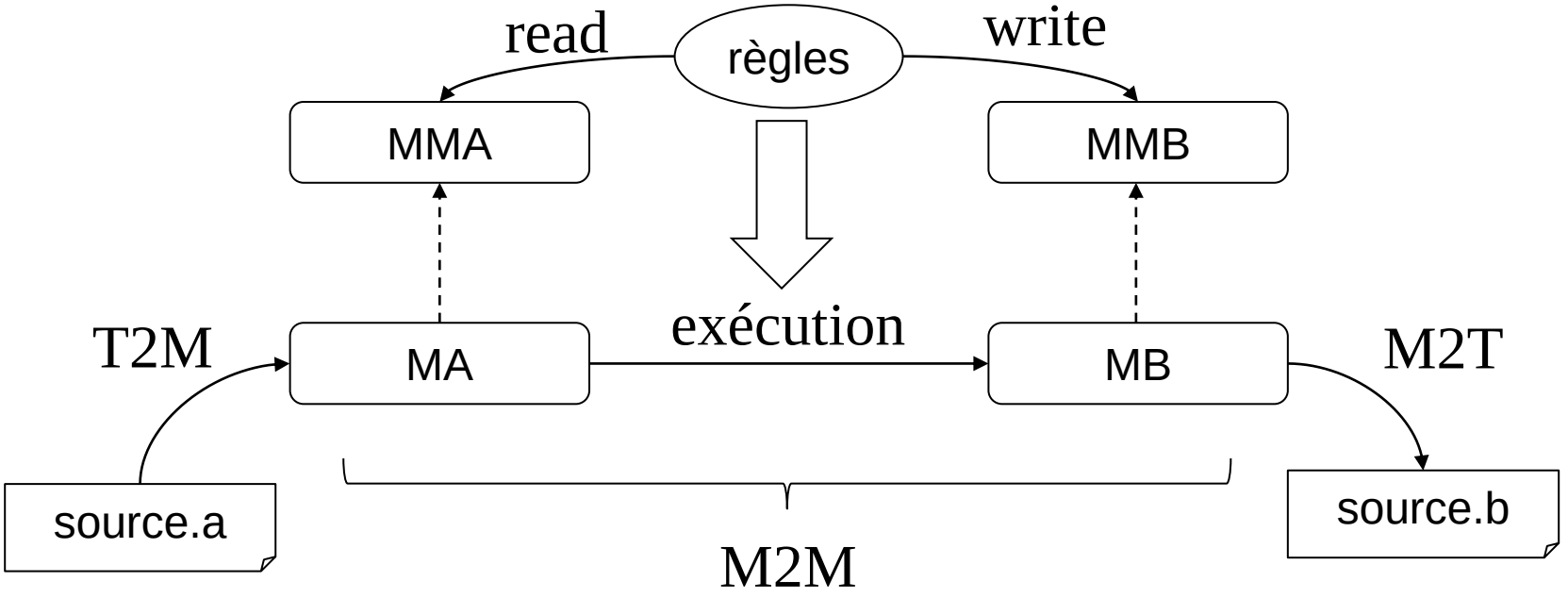
- Un méta-modèle définit une syntaxe abstraite d'un langage
  - Concepts simples similaires à ceux de nos diagrammes de classe métier
  - On peut engendrer du code à partir de la description du méta-modèle
    - Une classe pour chaque concept
    - API en lecture (getter) et écriture (setter)
    - Factory pour engendrer des modèles à partir de programmes
    - Permet de manipuler les modèles instances du langage
- Le méta-modèle induit un arbre couvrant du graphe des modèles instance
  - On peut en déduire un format de stockage XML immédiat et universel

Des modèles « productifs »

Utiliser les modèles ?

- Le modèle est une information structurée
  - Il se conforme à son MM
  - Les concepts qu'on peut rencontrer et les liaisons possibles entre ces concepts sont donc connues
  - Possibilité de raisonner sur ces concepts algorithmiquement !
- Actions sur les modèles :
  - Générer une documentation
  - Générer du code ou globalement exécuter le modèle
  - Faire une vérification ou validation du modèle
  - Proposer un refactoring
  - Transformer le modèle vers une autre notation ou formalisme
  - Extraire des informations du modèle pour les présenter différemment
  - …

Transformations de modèle



- T2M : Text-to-Model, parse, désérialisation
  - action de charger un modèle en mémoire comme une instance d'un MM
- M2T : Model-to-Text, sérialisation, export
  - Produire une représentation sous la forme de fichier(s)
- M2M : Model-to-Model

Transformations de modèle

- T2M
  - Parcourir le fichier d'entrée et construire (écriture) en mémoire un modèle
  - Format standard de stockage des modèles (et des méta-modèles) : XMI
    - Basé sur XML
  - ✓ Pour les autres formats on utilisera un parser, typiquement appuyé par un outil e.g. ANTLR, Xtext, …
- M2T
  - Parcourir le modèle (en lecture) et produire une sortie
  - Problème de réversibilité : **M2T(T2M(M)) = M** ?
- M2M
  - Exprimer des règles de transformation, e.g. pour chaque « classe UML » engendrer une « table SQL ».
    - « exogène » : MMA!=MMB, transformation d'un langage à un autre
    - « endogène » : MMA=MMB, transformation d'un modèle sans changer de langage, e.g. refactoring
  - A l'exécution, on parcourt le modèle source (lecture) pour construire la cible (écriture)

Modèle-Evangélisme (Bézivin)

- Toute information est un modèle
  - Un « modèle » est suffisamment général pour capturer n'importe quel type d'information
- Tout algorithme ou raisonnement prend en entrée un modèle et produit en sortie un modèle
  - Modèle d'entrée = les données du problème
  - Modèle de sortie = les conclusions de l'algorithme
- Tout algorithme peut donc être exprimé ou perçu comme une transformation de modèle !

Transition vers les DSL

- UML est trop gros !
  - 260 méta-classes, 14 diagrammes,
  - des stéréotypes, des profils, des {tagged values} pour étendre le langage
  - Points de variation sémantique, pas de syntaxe concrète pour l'algorithmique
  - Sémantique opérationnelle vague => génération de code ? Simulation ?
- Emergence des Domain-Specific Language DSL
  - Cibler un domaine métier particulier au lieu d'être trop général
  - Se limiter aux concepts pertinents, faire des petits langages
  - Automatiser les taches répétitives du métier en offrant des abstractions adaptées
  - Faire des langages "executables"
  - Utiliser les technologies issues du MOF pour faciliter le développement du DSL

Evolution des DSL

- Les DSL existent depuis longtemps
  - Makefile, SQL, dot/Graphviz…
  - Historiquement développement assez lourd et dédié
    - Outils d'édition, parser, compilateur, affichage des erreurs…
- On trouve aussi des DSL graphiques ou semi-graphiques e.g. UML
  - Edition des modèles plus difficiles, problème de passage à l'échelle
  - La tendance actuelle est plutôt aux DSL textuels, accompagnés éventuellement de visualisations non éditables.
- Le développement d'un DSL = nouveau langage
  - Il suffit de définir le méta-modèle du langage dans le format EMF standard pour avoir accès à une galaxie d'outils dédiés au services autour du modèle
  - Des centaines d'homme-an de développement FOSS disponible + des outils propriétaires

Les services à partir du méta-modèle

- Génération du code d'une API de manipulation
  - Lecture, Ecriture, Construction d'instances (factory), réflexion
- Génération de formats de stockage fichier
  - M2T/T2M réversible standard vers format XMI
  - Support pour d'autres formats, eg. Grammaire Xtext
- Edition des modèles
  - Editeur arborescent par défaut,
  - Editeurs pour saisie de données, EMF Forms
  - Editeurs graphiques ou semi-graphiques, GMF, Graphitti, …
  - Editeurs textuels correction au cours de la frappe, coloration, complétion…
- Comparaisons, versioning
  - EMF compare, integration adapteur git
- Validation des modèles
  - Définir des règles en OCL, epsilon, … les voir affichées problem view

Les services à partir du méta-modèle (2)

- Stockage de grands modèles, interrogation
  - CDO, Net4J, Teneo…
- Transformations de modèles
  - ATL, Epsilon, Acceleo, Xtext...

…

https://www.eclipse.org/modeling/

Un exemple de Méta-modèle

https://lip6.github.io/ITSTools-web/galmm.html

Issu d'une grammaire Xtext :
https://github.com/lip6/ITSTools/blob/master/fr.lip6.move.gal/src/fr/lip6/move/Gal.xtext

The End

Bonne chance pour l’examen

Soutenances finales LDVH première semaine de Janvier

+ Consultations des copies