Architecture dirigée par les modèles MDA

Model Driven Architecture - MDA

Introduction

- **Problème lié à la MAJ des logiciels**
  - Les chefs d'entreprise hésitent à faire des MAJ sans la garantie d'un retour sur investissement.
  - Cette MAJ consiste à redévelopper et porter les logiciels vers une nouvelle plate-forme.
  - La logique métier de l'entreprise n'a pas changé.
  - Evolution incessante et inévitable des technologies

Model Driven Architecture - MDA

❑ Introduction

- **CORBA** (Common Object Request Broker Architecture) standard du Middleware (OMG). Interopérable avec RMI API java, permet d'appeler des méthodes distantes;
- **EJB** (Entreprise Java Bean) est une architecture de composants logiciels côté serveur. Cette architecture propose un cadre pour créer des composants distribués ;
- **.Net** est un Framework pouvant être utilisé par un système d'exploitation Microsoft Windows. Il a pour but de faciliter la tâche des développeurs en proposant une approche unifiée à la conception d'applications Windows ou Web.

Model Driven Architecture - MDA

Introduction

❑ Solution proposée par OMG

❖ Pour permettre l'interopérabilité entre les différents systèmes, réduire les coûts de développement et augmenter l'évolutivité, l'OMG (Object Management Group) propose la démarche MDA (Model Driven Architecture), où la notion de modèle devient essentielle et centrale.

❖ La logique métier étant stable. Ce standard a pour but d'apporter une nouvelle façon de concevoir des applications en séparant la logique métier de l'entreprise, de toute plate-forme technique.

Model Driven Architecture - MDA

❑ Introduction

- ❖ La démarche MDA propose de définir un modèle métier indépendant de toute plate-forme technique et de générer automatiquement du code vers la plate-forme choisie.

- ❖ L'accent est mis non plus sur les approches objet mais sur les approches modèles. Le but est de favoriser l'élaboration de modèles de plus haut niveau.

Model Driven Architecture - MDA

❑ Définition

- ❖ L'**architecture dirigée par les modèles** ou **MDA** (pour l'Anglais **Model Driven Architecture**) est une démarche de réalisation de logiciel, proposée et soutenue par l'OMG.
- ❖ C'est une variante particulière de l'ingénierie dirigée par les modèles (IDM, ou MDE pour l'Anglais *Model Driven Engineering*).
- ❖ L'**Ingénierie dirigée par les modèles** (IDM) est le domaine de l'informatique mettant à disposition des outils, concepts et langages pour créer et transformer des modèles.
  - ✓ Langages : UML, MOF, QVT :
  - ✓ Mots-clés : Modèle, Métamodèle, Méta métamodèle, Diagramme de classe…

Model Driven Architecture - MDA

❑ Meta-Object Facility

Le **Meta-Object Facility** (MOF) est un standard de l'OMG s'intéressant à la représentation des méta-modèles et leur manipulation. Le langage MOF s'auto-définit.

Le standard MOF est situé au sommet d'une architecture de modélisation en 4 couches:
M3: le métamétamodèle MOF (couche auto descriptive)
M2: les métamodèles
M1: les modèles
M0: Le monde réel

Model Driven Architecture - MDA

- **Meta-Object Facility :** Le langage UML est décrit par un métamodèle conforme au MOF. Ainsi un modèle UML peut être **sérialisé en XMI**. Mais il y a également de nombreux autres métamodèles situés au même niveau que UML.
- **Sérialisation :** c'est un processus visant à encoder l'état d'une information qui est en mémoire sous la forme d'une suite d'informations plus petites, en octets ou en bits, pour la sauvegarde (persistance) ou le transport sur le réseau (proxy, RPC…).
  - L'activité symétrique, visant à décoder cette suite pour créer une copie conforme de l'information d'origine, s'appelle la désérialisation (ou unmarshalling).

Model Driven Architecture - MDA

- **Interopérabilité des méta-modèles :** Le MOF spécifie la structure et la syntaxe de tous ces méta-modèles. Il spécifie aussi des mécanismes d'interopérabilité entre ces méta-modèles. Il est donc possible de les comparer et de les relier. Grâce à ces mécanismes d'échanges, le MOF peut faire cohabiter plusieurs méta-modèles différents.
- **Systems Modeling Language :** SysML, est un langage de modélisation spécifique au domaine de l'ingénierie système. Il permet la spécification, l'analyse, la conception, la vérification et la validation de nombreux systèmes et *systèmes-de-systèmes*.

  - SysML se définit comme une extension d'un sous-ensemble d'UML via l'utilisation du mécanisme de profil défini par UML

Model Driven Architecture - MDA

- **Systems Modeling Language :** SysML offre aux ingénieurs systèmes plusieurs améliorations notables par rapport à UML, qui a tendance à être centré sur le logiciel.

- La sémantique de SysML est plus riche et flexible: SysML impose moins de restrictions liées à la vision d'UML centrée sur le logiciel.

Model Driven Architecture - MDA

Systems Modeling Language

SysML ajoute deux nouveaux types de diagrammes.

•Le premier peut être utilisé pour la gestion des besoins (requirements);

•le deuxième peut être utilisé pour l'analyse des performances et l'analyse quantitative.

Grâce à ces améliorations, SysML est capable de modéliser une large gamme de systèmes, incluant tant du matériel, que du logiciel, de l'information, des processus, du personnel, ou des équipements (au sens large).

Model Driven Architecture - MDA

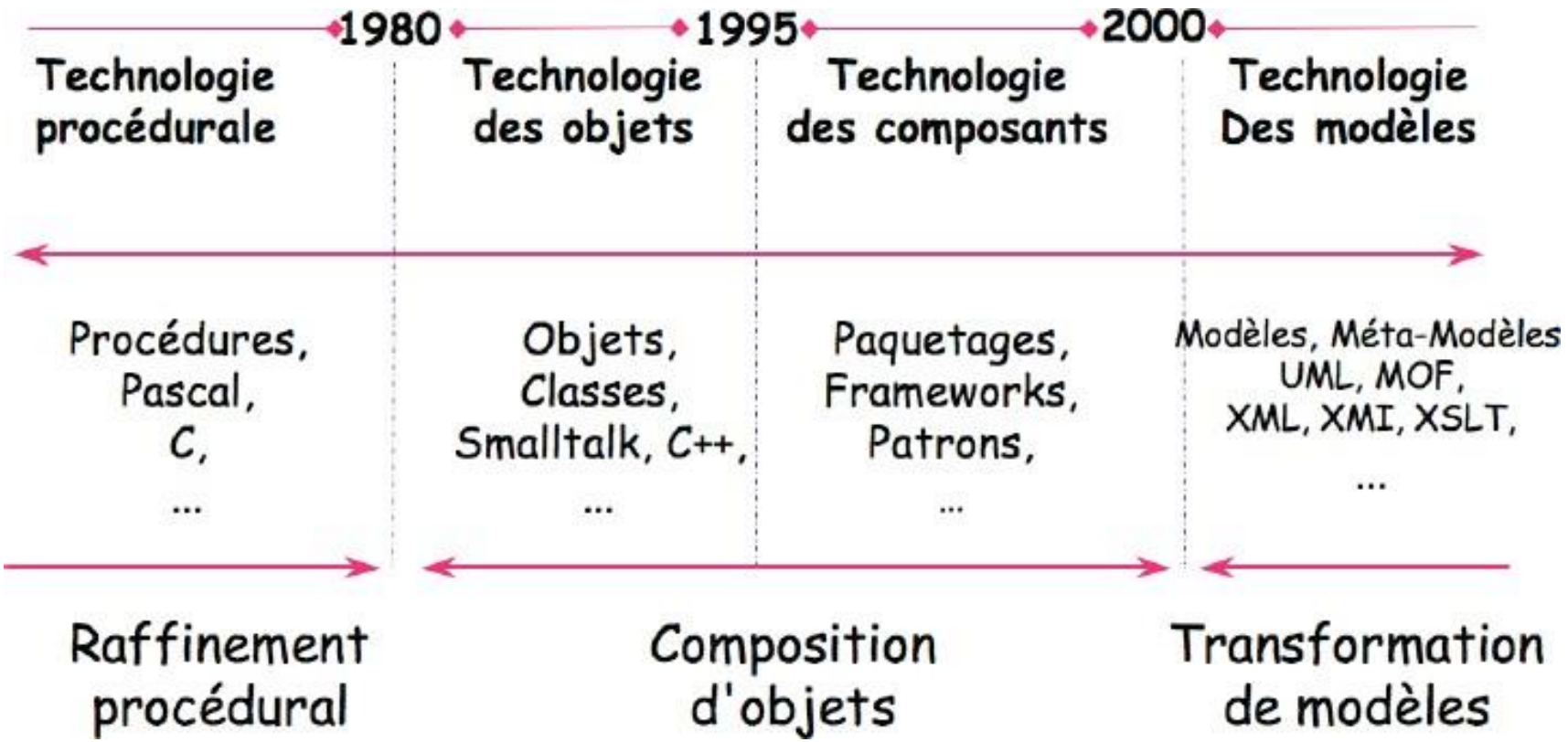
- **Systems Modeling Language : SysML** réutilise sept des treize diagrammes d'UML 2; Il ajoute deux diagrammes spécifiques (Diagrammes de Requirements et diagrammes Paramétriques),

- ***Les modèles :*** On utilise des modèles pour mieux comprendre un système. Pour un observateur A, M est un modèle de l'objet O, si M aide A répondre aux questions qu'il se pose sur O.

→Un modèle est une simplification, une abstraction du système.

Model Driven Architecture - MDA

Les modèles



Model Driven Architecture - MDA

Les métamodèles

•Selon MOF, un métamodèle définit la structure que doit avoir tout modèle conforme à ce métamodèle.

•Tout modèle doit respecter la structure définie par son métamodèle

•MOF les représente sous forme de *diagrammes de classes.*

•MDA ne donne aucune préconisation pour définir la sémantique
•d'un modèle

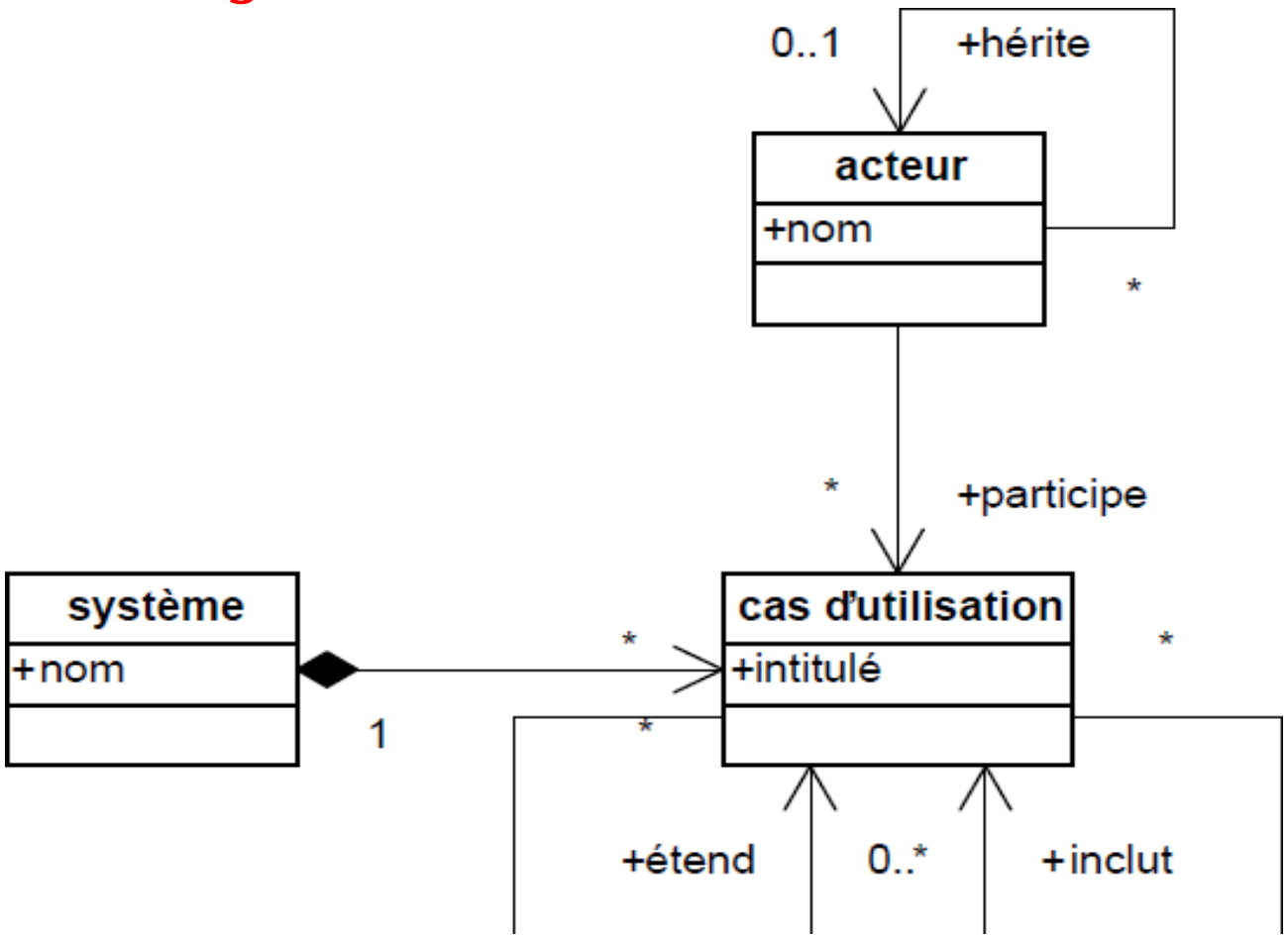
Model Driven Architecture - MDA

Exemples de métamodèles

« Un diagramme de cas d'utilisation contient des **acteurs,** un **système** et des **cas d'utilisation.** Un acteur a un nom et est *relié aux cas d'utilisation. Un acteur peut hériter d'un autre acteur. Un cas d'utilisation a un intitulé et peut étendre ou inclure un* autre cas d'utilisation. Le système a lui aussi un nom, et il inclut tous les cas d'utilisation. »

Model Driven Architecture - MDA

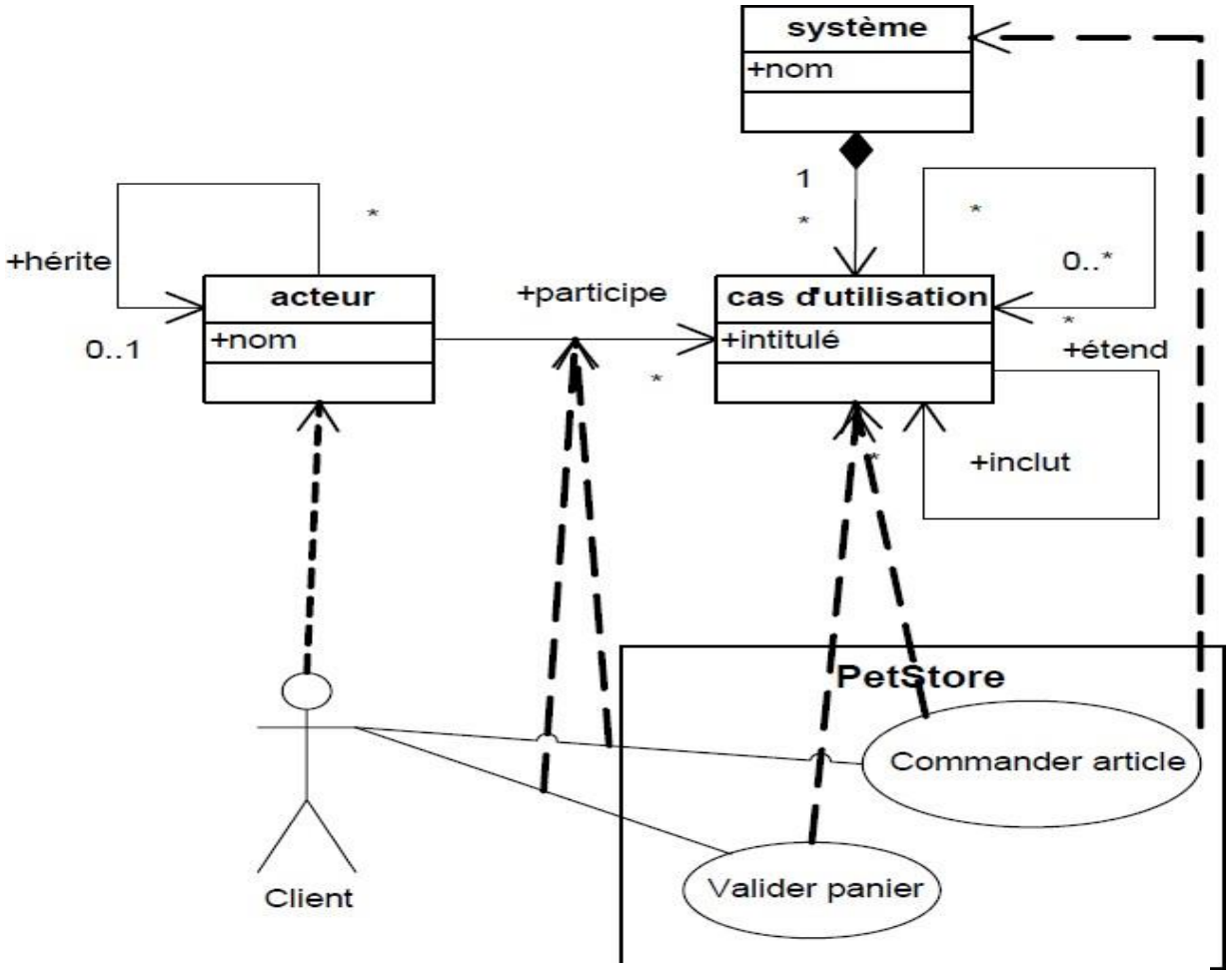
Métamodèle des diagrammes de cas d'utilisation



Model Driven Architecture - MDA

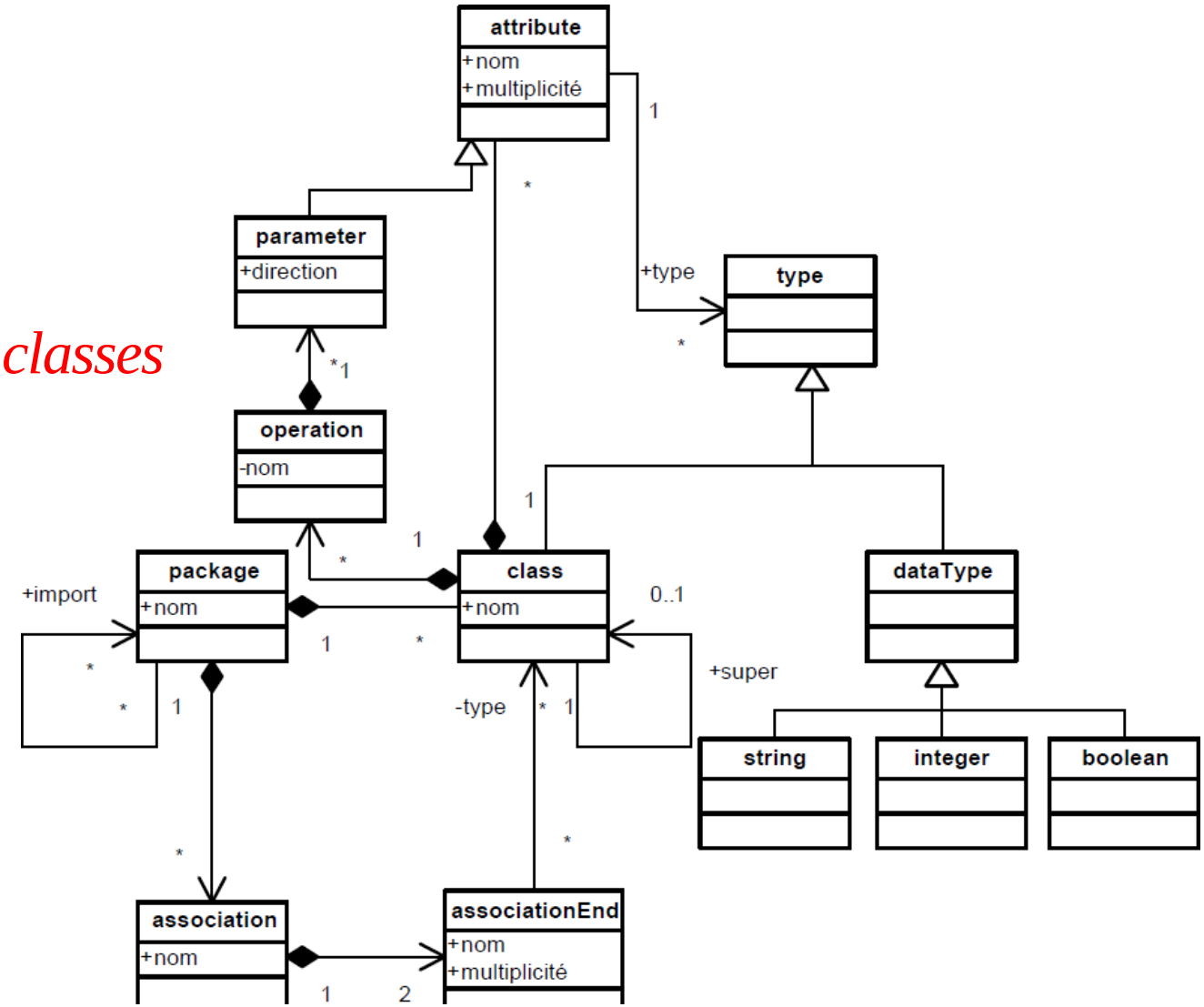
Métamodèle des diagrammes de cas d'utilisation

Un modèle est l'instance de son métamodèle



Model Driven Architecture - MDA

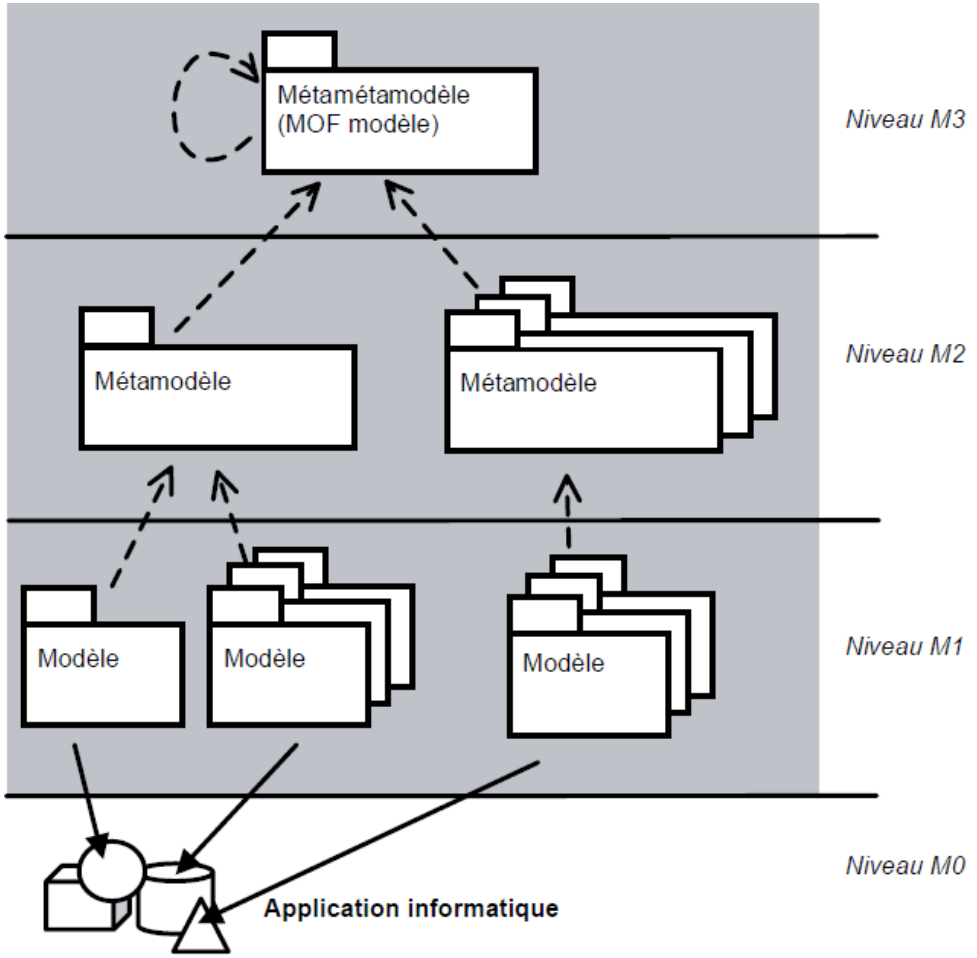
Représentation
de MOF1.4
sous forme
de diagramme de classes



Model Driven Architecture - MDA

Métamétamétamodèle

le métamétamodèle
s'autodéfinit et est
à lui-même
son propre
métamodèle



Model Driven Architecture - MDA

MOF2.0

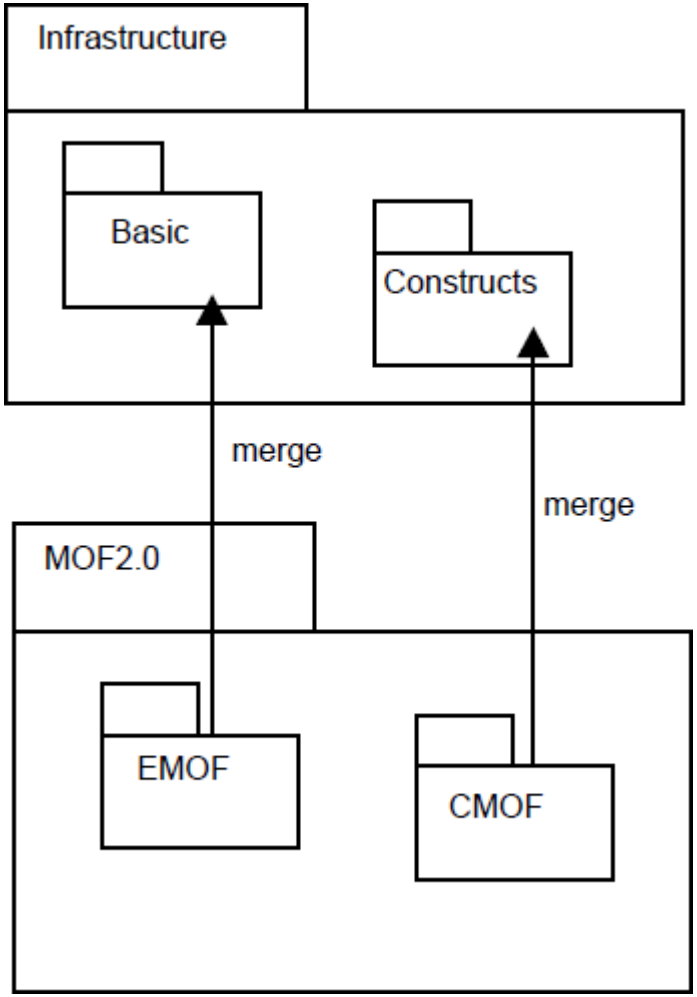
MOF2.0 a été décomposé en deux parties : EMOF (Essential MOF), pour l'élaboration de métamodèles sans association, et CMOF (Complete MOF), pour celle de métamodèles avec association

EMOF a pour source principale le framework de métamodélisation EMF (Eclipse Modeling Framework) proposé par Eclipse. *Ce framework* permet de générer automatiquement des interfaces Java à partir de métamodèles Ecore. La particularité des métamodèles Ecore est qu'ils ne contiennent pas de méta-associations entre leurs métaclasses.

Pour exprimer une relation entre deux métaclasses, il faut utiliser des méta-attributs et les typer par des métaclasses.

Model Driven Architecture - MDA

MOF2.0



Model Driven Architecture - MDA

Modèles CIM

Il est important de noter qu'un modèle d'exigences ne contient pas d'information sur la réalisation de l'application ni sur les traitements. C'est pourquoi, dans le vocabulaire MDA, les modèles d'exigences sont appelés des CIM (Computation Independent Model), littéralement « modèle indépendant de la programmation ».

Une fois le modèle d'exigences réalisé, le travail d'analyse et de conception peut commencer. Dans l'approche MDA, cette phase utilise elle aussi un modèle.

Model Driven Architecture - MDA

Principe de base du MDA

L'élaboration de modèles indépendants de plate-formes (*Platform Independent Model*, PIM) (PLATEFORME : EJB, CORBA, .NET,…)et la transformation de ceux-ci en modèles dépendants de plates-formes(*Platform Specific Model*, PSM) pour l'implémentation concrète du système. Les techniques employées sont donc principalement des techniques de modélisation et des techniques de transformation de modèles.

Model Driven Architecture - MDA

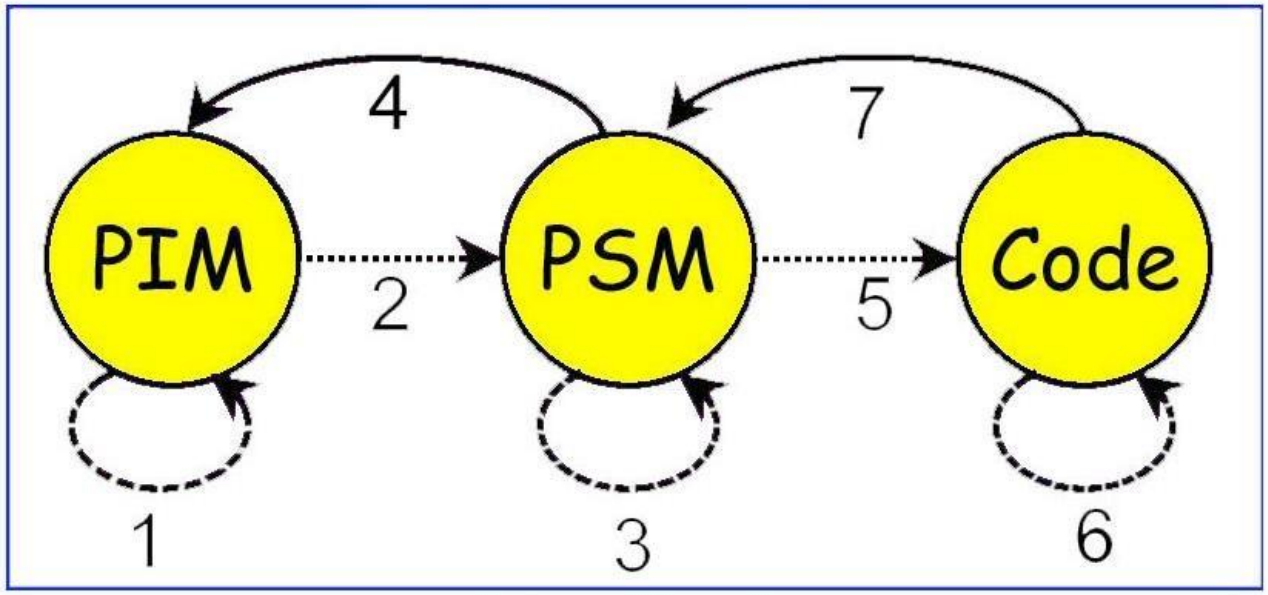
Le formalisme utilisé pour exprimer un PIM est un diagramme de classes en UML qui peut-être couplé avec un langage de contrainte comme OCL* (Object Constraint Language).

Il n'est bien sûr pas requis que TOUT le code soit généré automatiquement, mais l'architecture globale du système (ex: squelettes de code).

La traduction entre le PIM et les PSM est normalement effectuée à l'aide d'outils automatisés, par exemple des transformations de modèles réalisées avec des outils plus ou moins compatibles avec le standard de l'OMG nommé **QVT**.

Model Driven Architecture - MDA

La transformation des modèles du MDA



Model Driven Architecture - MDA

1: Un exemple de PIM à PIM : est le passage du modèle d'analyse à celui de conception.
Refactoring : réorganiser le code :utilisation d'un patron de conception

2: Les principales plates-formes visées sont J2EE, .NET ou CORBA, …

Model Driven Architecture - MDA

Une transformation PIM vers PSM n'est pas toujours suffisante pour permettre la génération de code d'où la nécessité de passer de PSM à PSM en utilisant des formalismes intermédiaires. Par exemple, pour générer un code C++, à partir d'un formalisme en UML, un passage d'UML vers SDL puis de SDL vers C++ pourrait être utilisé

C'est une opération de rétroingénierie (reverse engineering) qui est assez complexe à réaliser et difficilement automatisable. Ces transformations sont néanmoins nécessaires pour permettre l'intégration d'applications existantes dans le processus MDA.

Model Driven Architecture - MDA

QVT (Query/View/Transformation)

est un standard défini par l'OMG. Il s'agit d'un langage standardisé pour exprimer ces transformations de modèles.

Il définit un ensemble de langages :

- QVT-Relation est un langage déclaratif ;
- QVT-Operational est un langage hybride qui propose une structure déclarative à base de règles et permet l'utilisation d'expressions impératives ;
- QVT-Core définit la sémantique des concepts déclaratifs.

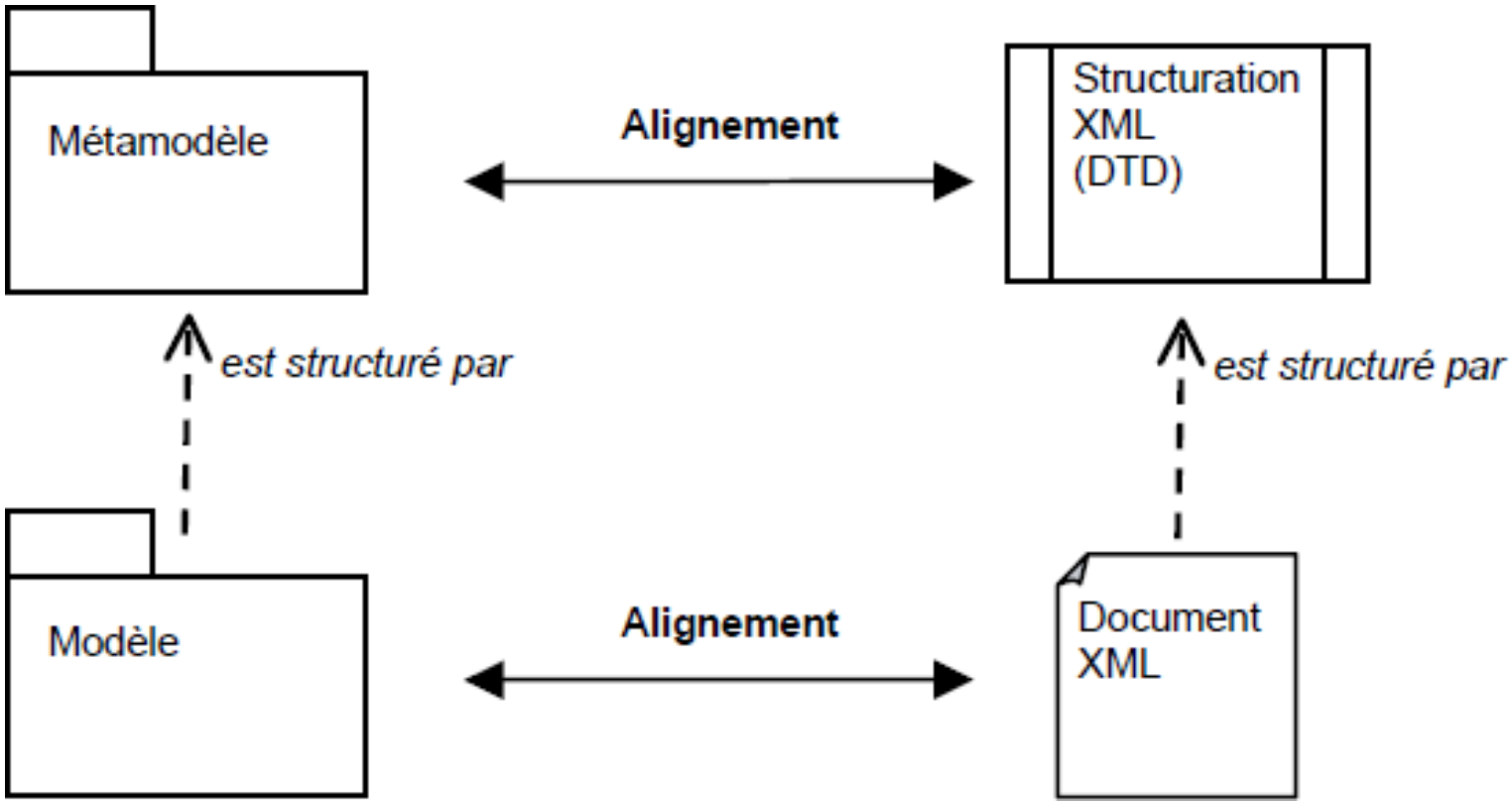
Model Driven Architecture - MDA

XMI (XML Metadata Interchange)

- XMI est un standard créé par l'OMG
- XMI est un procédé de sérialisation d'objets MOF (un autre standard de l'OMG), permettant de décrire des objets sous forme XML
- Les modèles étant des entités abstraites, ils ne disposent pas intrinsèquement de représentation informatique.
- L'objectif premier de XMI est de définir un moyen permettant de représenter un modèle sous forme dedocument XML.
- utilise les mécanismes de définition de structure de balises XML DTD et XML Schema

Model Driven Architecture - MDA

Alignement entre métamodèle/modèle et DTD/document XML



Model Driven Architecture - MDA

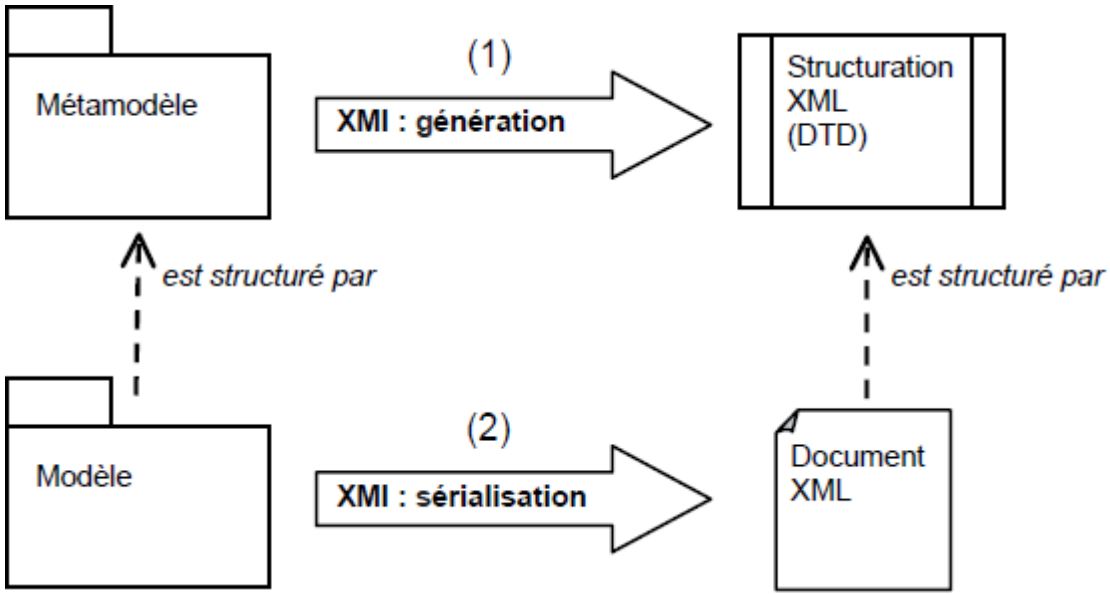
Alignement entre métamodèle/modèle et DTD/document XML

La structuration des balises XML permettant de représenter des modèles sous forme de document XML se fonde sur les métamodèles des modèles. Par exemple, la structuration des balises XML permettant de représenter des modèles UML sous forme de document XML s'appuie sur le métamodèle UML.

Model Driven Architecture - MDA

Règles de génération des balises XML

L'alignement des métamodèles et des structurations de balises XML permet à XMI de définir un ensemble de règles de génération automatique de structuration de balises XML à partir d'un métamodèle



Model Driven Architecture - MDA

Règles XMI de génération de DTD à partir d'un métamodèle MOF1.4

1. Toute métaclasse fournit la définition d'une balise ayant comme nom le nom de la métaclasse. Cette balise doit posséder un attribut XML nommé id permettant d'identifier l'instance de la métaclasse afin de la référencer au besoin.
2. Tout méta-attribut d'une métaclasse fournit la définition d'une balise ayant comme nom le nom du méta-attribut. Le contenu de la balise représentera la valeur du métaattribut. Cette balise doit être contenue dans la balise correspondant à la métaclasse contenant l'attribut.

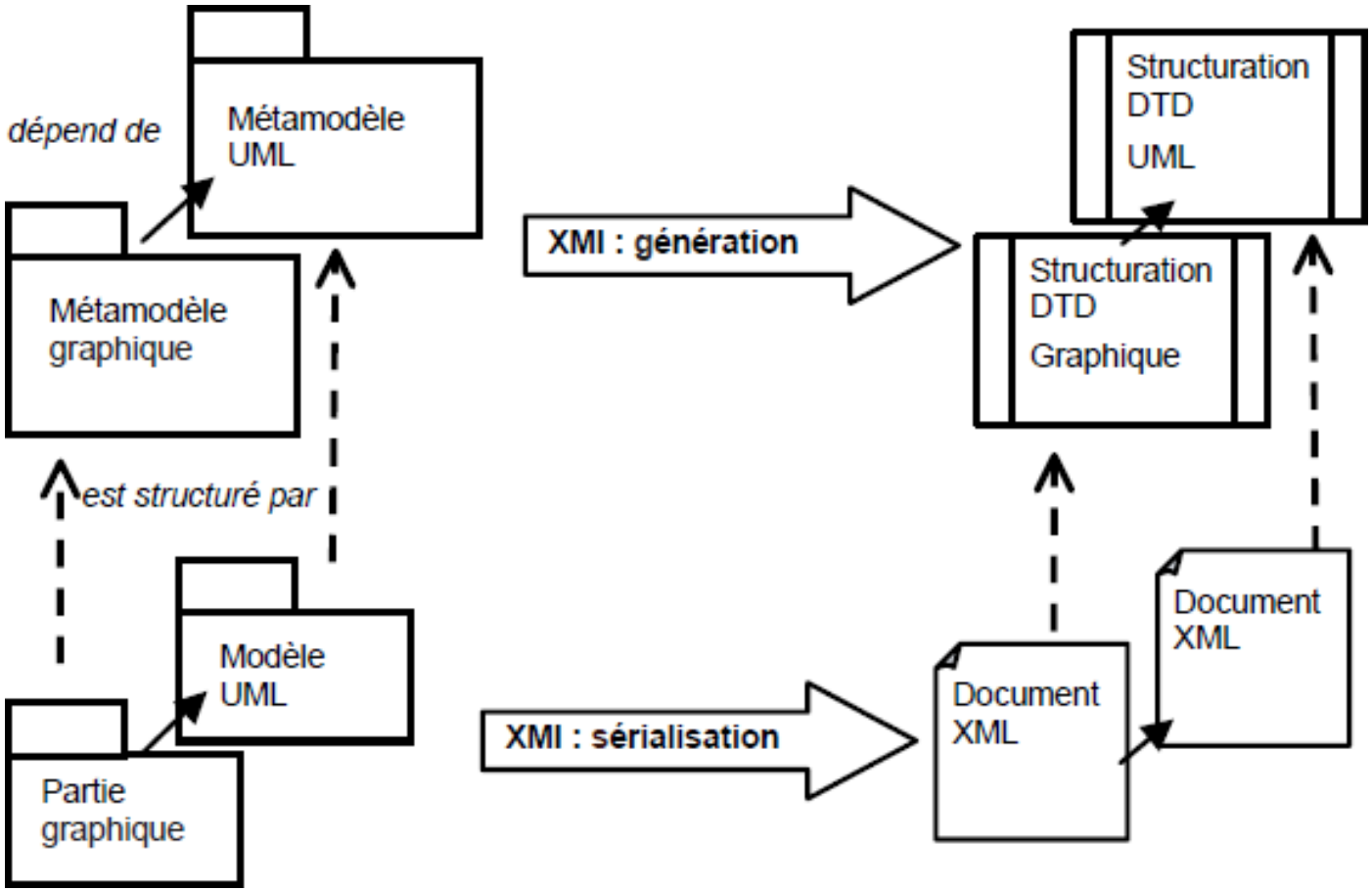
Model Driven Architecture - MDA

DI (Diagram Interchange)

- Il n'est pas tout à fait exact de dire que le standard XMI permet de représenter les modèles sous forme de document XML.
- la partie graphique des modèles, c'est-à-dire les diagrammes, n'est pas définie dans le métamodèle UML
- L'approche définie par DI permet la représentation au format XML des parties graphiques des modèles UML
- L'idée est de définir un nouveau métamodèle, lié au métamodèle UML, représentant sous forme de métaclasses les idiomes graphiques nécessaires et suffisants à la représentation de toutes les parties graphiques d'UML.

Model Driven Architecture - MDA

DI (Diagram Interchange)



Model Driven Architecture - MDA

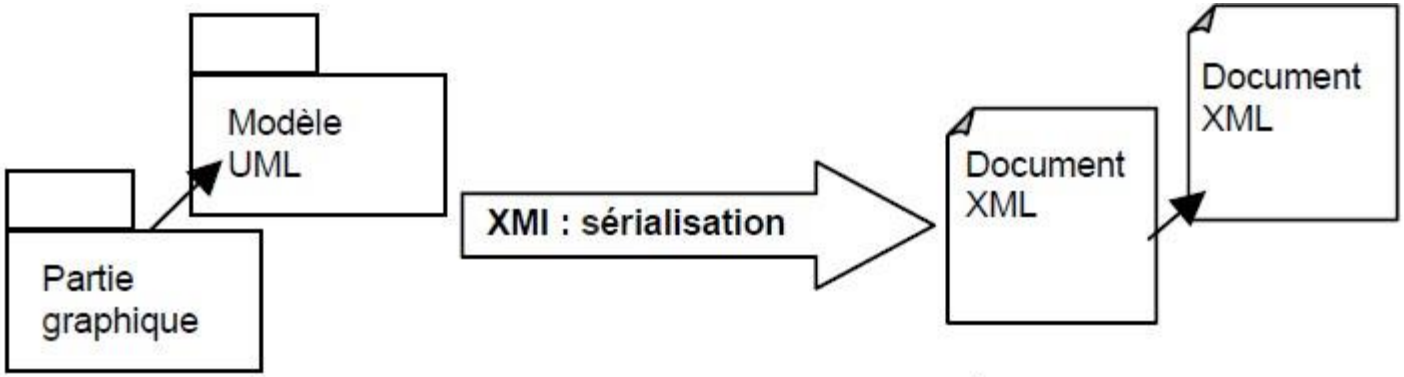
DI (Diagram Interchange)

Le standard DI permet de représenter les parties graphiques des modèles UML sous forme de documents XML

DI définit une transformation XML permettant de générer automatiquement des documents SVG à partir des documents XML représentant les modèles UML et leurs parties graphiques.

Cette génération permet de rendre visible les modèles UML dans n'importe quel outil supportant le format SVG. Cette génération étant standard, il est de plus possible d'échanger complètement des modèles UML, parties graphiques comprises.

Model Driven Architecture - MDA



Transformation vers SVG

```
<svg
xmlns="http://www.w3.org/2000/svg">
<style type="text/css">
...
</style>
```