Transformation de modèles

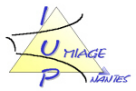
dans le cadre de l'OMG

Pascal André

MIAGE - LINA
Université de Nantes

January 18, 2011

Master Miage M2 - Option ISI



Plan

Généralités

1. Modélisation
2. Méta-modélisation
3. Transformation de modèles
4. Cadre général de la transformation de modèles

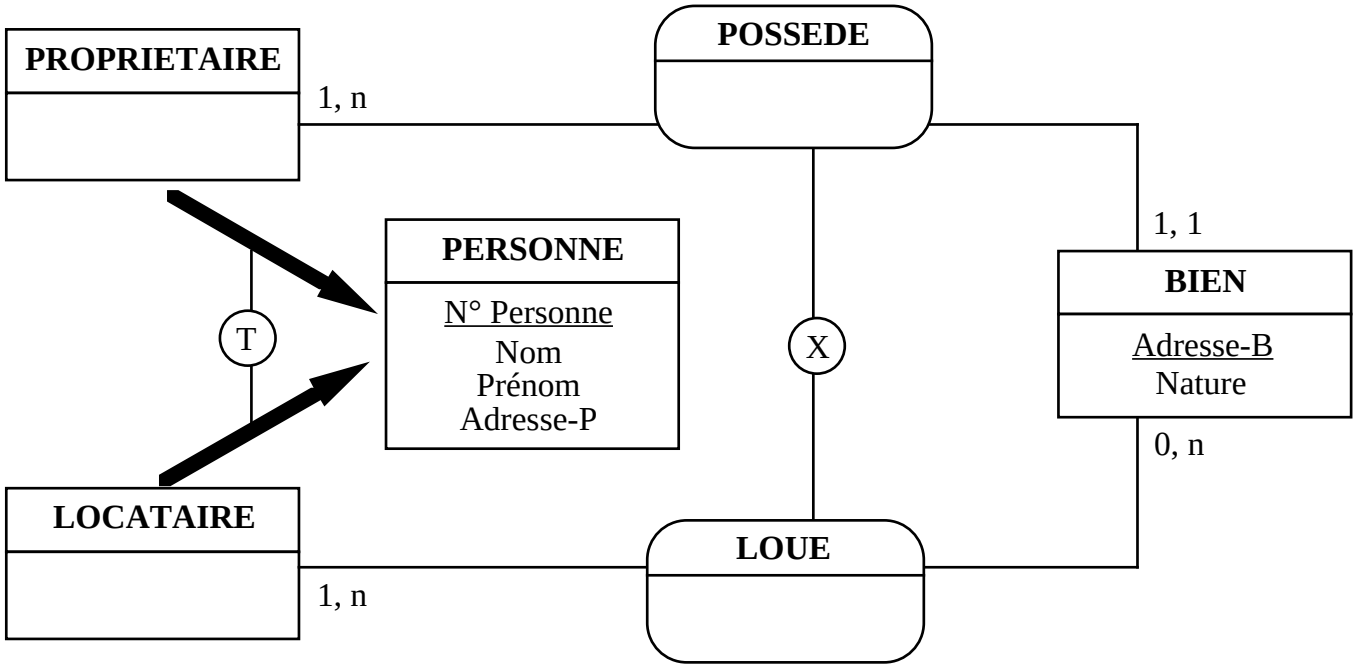
Modélisation

- Pour raisonner sur des systèmes, on en fait une abstraction : un modèle.
  Un modèle est donc une abstraction d'un système, c'est-à-dire une image dans laquelle on ne retient que les caractéristiques utiles pour raisonner sur le système (répondre à des questions qu'on se pose sur le système).
- Le modèle est exprimé dans un langage.
  Le langage associé au modèle décrit les éléments des systèmes retenus pour le modèle.
- Exemples : E-A-P, Automates, RDP, Z, [AV01a]

Modélisation : les manipulations

- Vérifications (compilation, preuves...)
  - vérifier les propriétés du modèle (pas d'erreur)
  - vérifier les propriétés du système (non-redondance, non blocage, sûreté, ...)
- Validations : comparer le modèle avec son système la réalité (lectures, tests, simulations...)
- Transformations
  - Raffiner/Abstraire : changer de niveau d'abstraction
  - Implanter : version exécutable du modèle
  - Extraction/Fusion : points de vue d'un modèle, structuration de modèles,
  - Transformation vers d'autres modèles : changement de représentation, communication...

Modélisation : Merise



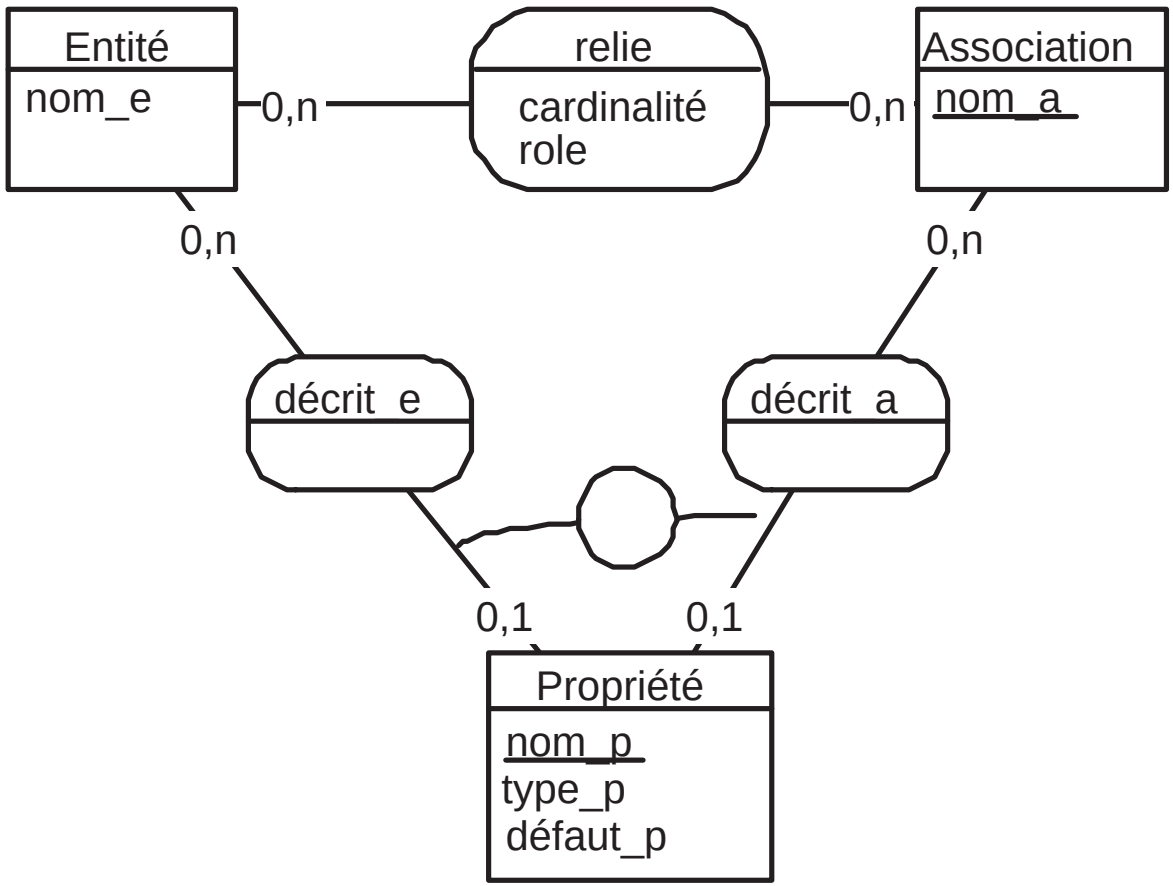
Méta-modélisation

- Pour raisonner sur des modèles, on en fait une abstraction : un métamodèle. Un métamodèle est donc une abstraction d'un modèle, c'est-à-dire une image dans laquelle on ne retient que les caractéristiques utiles pour raisonner sur les modèles (comparer des modèles, transformer des modèles).
- Le métamodèle est exprimé dans un langage.
  Le langage associé au métamodèle décrit les éléments des modèles et donc des langages de modélisation.
- Par la notation elle-même $\Rightarrow$ réflexif
- Exemple : Merise - E-A-P

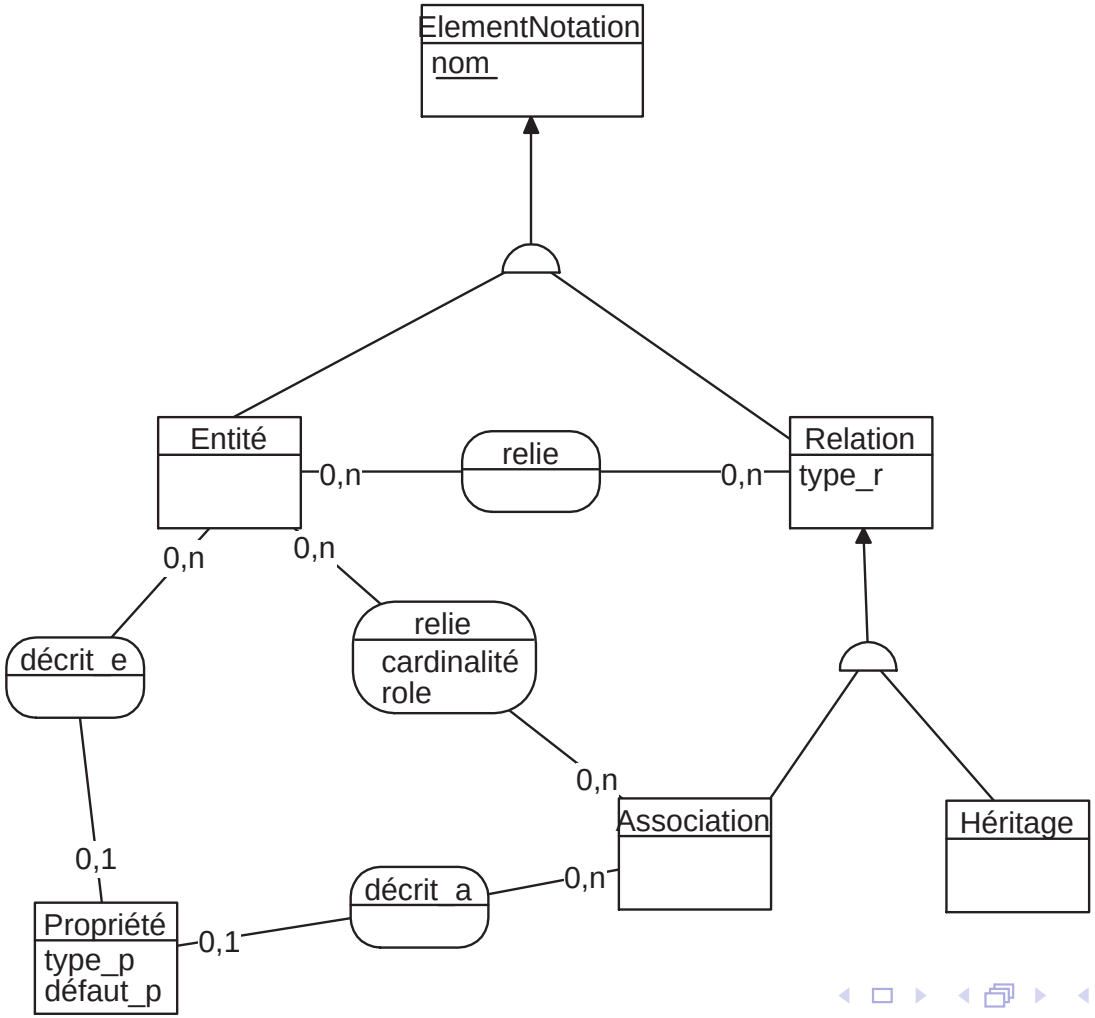
Méta-modélisation : les manipulations

- Vérifications
  - vérifier les propriétés du métamodèle (cohérent, complet)
  - vérifier les propriétés du langage (couverture, bien fondé, ...)
- Comparaison : comparer les métamodèles (classification...)
- Transformations
  - Celles de modèles
  - Gérer l'évolution (ex: UML 1.3 $\rightarrow$ UML 1.4) ou la compatibilité
  - Conversion entre langages, documentation
  - Portabilité, Interopérabilité, Normes d'échange

Méta-modèle : Merise 1/2



Méta-modèle : Merise 2/2



Exercice

Etablir les correspondances entre
le modèle de la figure 1
et
le métamodèle de la figure 3.

Transformation de modèles : principes 1/4

1. Modèle
   - Langage (syntaxe concrète, syntaxe abstraite, sémantique)
2. Transformation
   - Type = relation/fonction
   - Expression (calcul, assertions...)

Transformation de modèles : principes 2/4

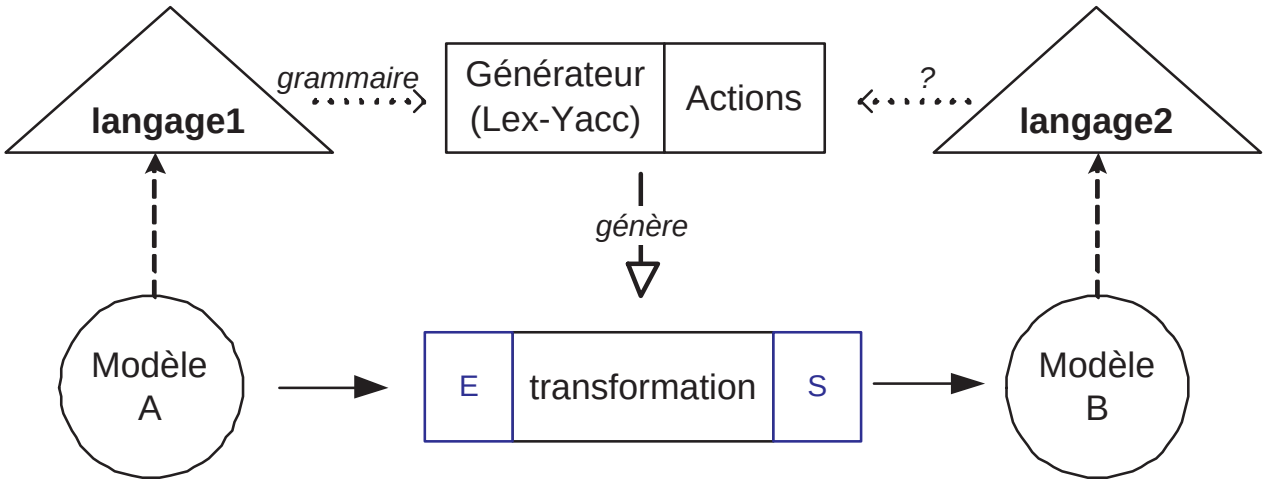
Entrée = Acquisition, analyse
Sortie = Génération "de code"

- lourde de tâche de codage
- quelle généralité pour l'entrée ? la sortie ?
- spécifique / générique
- normes ?

Pour cela, on doit disposer de modèles des entrées et/ou des sorties.

Transformation de modèles : principes 3/4

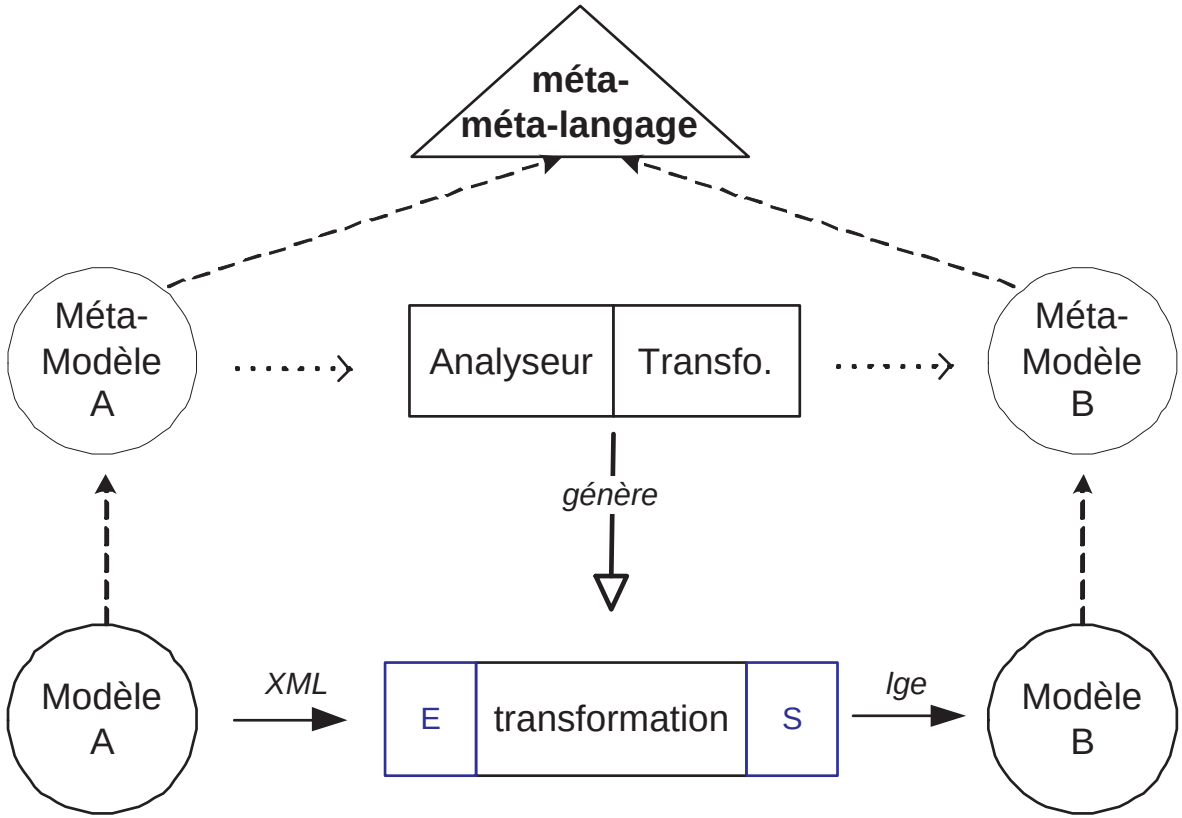
Premier niveau d'abstraction



Premier niveau d'abstraction : à partir de grammaires, on spécifie les entrées, la génération (de code) est orientée par la syntaxe (abstraite ou concrète).

Transformation de modèles : principes 4/4

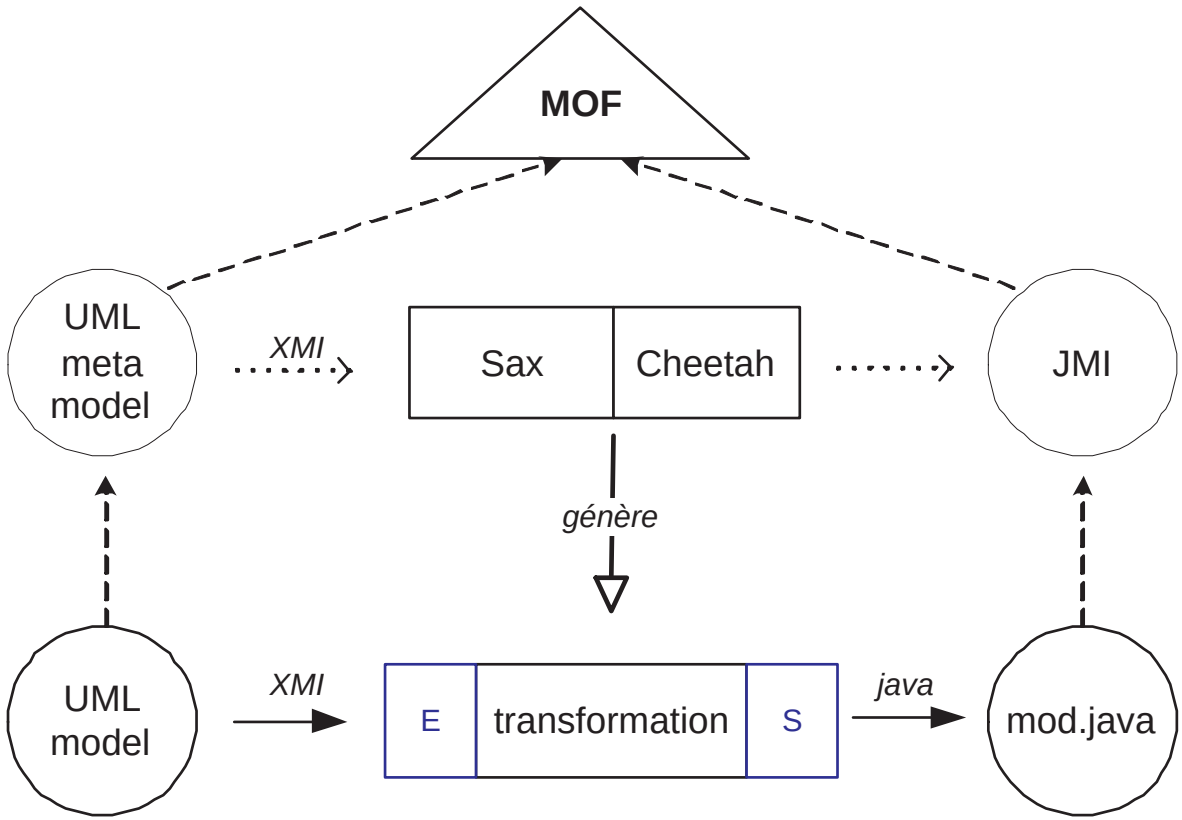
Second niveau d'abstraction



on ne manipule que des modèles

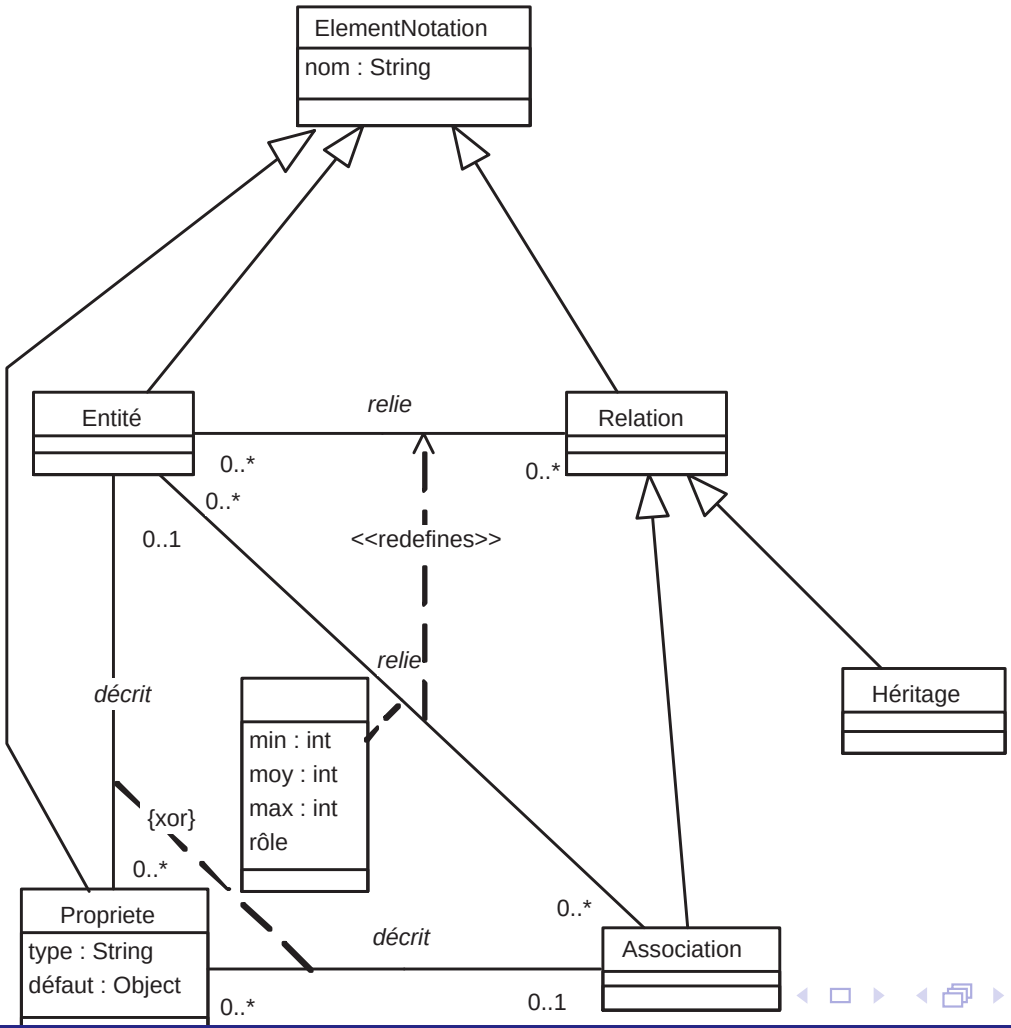
Transformation de modèles : exemple

Modèles objet



Normes d'échange : XML, XMI, EDOC...

Méta-modèle : Merise - UML



Méta-modèle : Merise - Java 1/2

Listing 1: Code Java de la classe `Association`

```
// Source file: Association.java

public class Association extends Relation {
    public Entité relie [];
    public Propriete décrit [];

    Association () {
    }
}
```

Méta-modèle : Merise - Java 2/2

Listing 2: Code Java de la classe `Propriete`

```
// Source file: Propriete.java

public class Propriete {
    private String type;
    private Object défaut;
    public Entité décrit;
    public Association décrit;

    Propriete() {
    }
}
```

Transformation de modèles

Implantation

1. langage normalisé pour exprimer des modèles : XML
2. arbre abstrait
3. enrichir la description
4. opérations de traitement de modèles : visiteurs

Architecture OMG - Cadre global MDA

- Langage de modélisation ⇒ UML, OCL, (AS)
- Langage de métamodélisation ⇒ MOF
- Normes d'échange de modèles ⇒ XML, XMI
- Langage pour la transformation ⇒ QVT

Exercices

TD numéro 2

Plan

OMG : généralités

- Consortium à but non lucratif créé en 1989 afin de normaliser les systèmes à objets
- Regroupe des acteurs de l'industrie informatique (fabricants de matériels, fournisseurs et éditeurs de logiciels, utilisateurs), des institutions, des universités...
- http://www.omg.org
- But :
  Produire et maintenir des standards (des spécifications) indépendants pour l'interopérabilité des applications informatiques et des systèmes informatiques hétérogènes.

OMG : les membres

Fondé à l'initiative de 11 grandes sociétés américaines [BGV97] :
American Airlines, Canon, Data General, Gold Hill
Hewlett-Packard, Philips, Prime, Sun, Soft-switch Unisys, 3Com.

Matrix
http://www.omg.org/memberservices/feestructure.htm

OMG : les membres

Fondé à l'initiative de 11 grandes sociétés américaines [BGV97] :
American Airlines, Canon, Data General, Gold Hill
Hewlett-Packard, Philips, Prime, Sun, Soft-switch Unisys, 3Com.

- 1989 : 11 membres puis 40 membres
- 1993 : 290 membres
- 1996 : plus de 500 adhérents
- actuellement : des "centaines d'organisations" (381 membres référencés sur le site de l'OMG en 2011)

Matrix
http://www.omg.org/memberservices/feestructure.htm

OMG : les membres

Fondé à l'initiative de 11 grandes sociétés américaines [BGV97] :
American Airlines, Canon, Data General, Gold Hill
Hewlett-Packard, Philips, Prime, Sun, Soft-switch Unisys, 3Com.

- 1989 : 11 membres puis 40 membres
- 1993 : 290 membres
- 1996 : plus de 500 adhérents
- actuellement : des "centaines d'organisations" (381 membres référencés sur le site de l'OMG en 2011)

Organisation indépendante et ouverte à tous
cotisation de 500$ à 75000$ annuels selon l'influence (!) Matrix
http://www.omg.org/memberservices/feestructure.htm

OMG : les objectifs

Objectif fondamental : interopérabilité d'applications à objets (intégration)

Objectifs initiaux

- Interopérabilité des applications à objets hétérogènes
- Mettre fin à la cacophonie des langages à objets (programmation, modélisation)
- Normaliser les systèmes, les langages à objets

Objectifs actuels

- Interopérabilité des développements à objets
- Normaliser les processus de développement
- Normaliser les modèles et leurs échanges
- Expertise, validation

OMG : les activités

Deux grandes générations à l'OMG

- Avant 2000
  le modèle OMA : Object Management Architecture
  interopérabilité entre applications à objets développées sur des réseaux hétérogènes
  - CORBA 1.1 → CORBA 3.0
  - IDL

OMG : les activités

Deux grandes générations à l'OMG

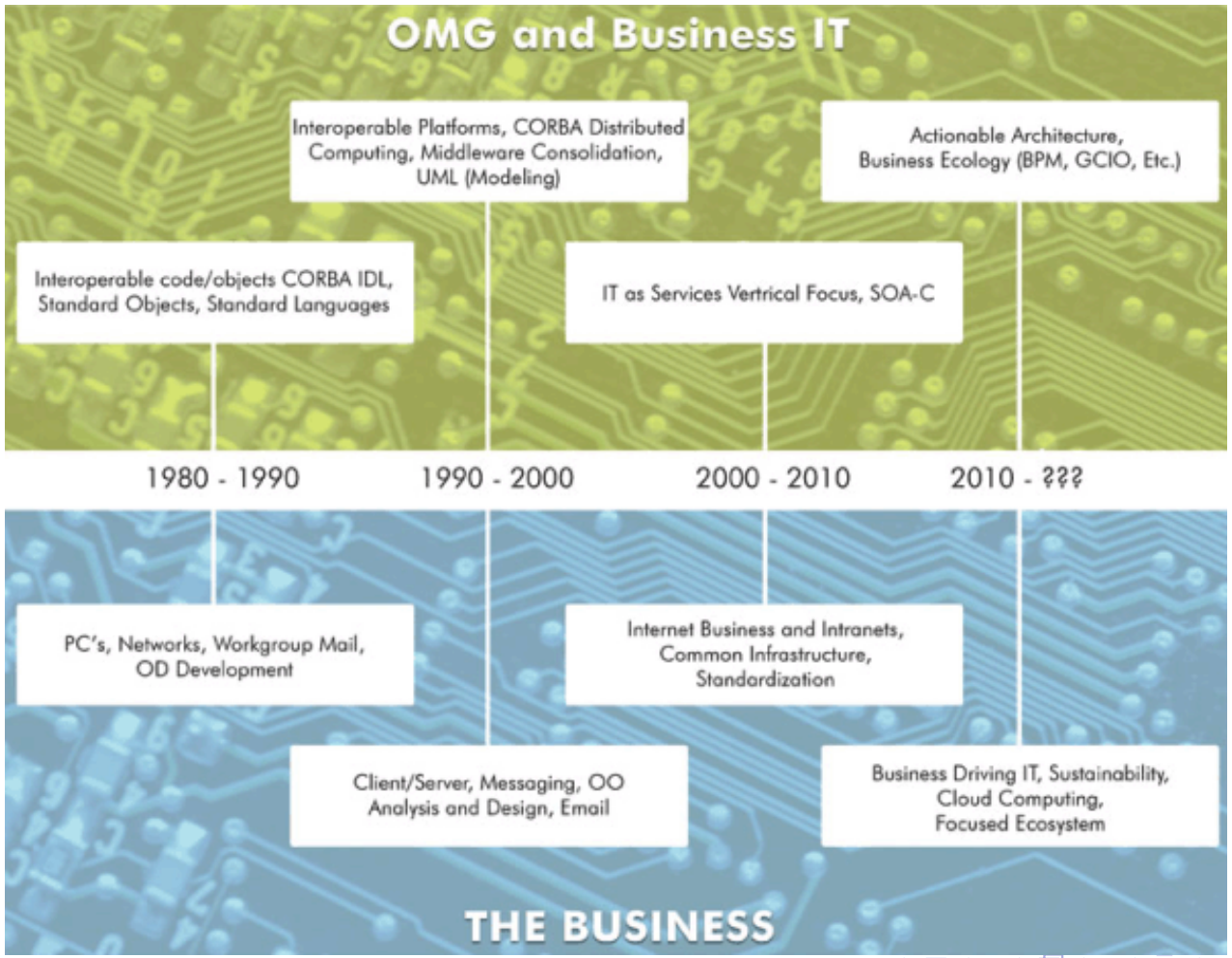
- Avant 2000
  le modèle OMA : Object Management Architecture
  interopérabilité entre applications à objets développées sur des réseaux hétérogènes
  - CORBA 1.1 → CORBA 3.0
  - IDL
- Progressivement
  - normalisation des langages : UML, OCL, XMI
  - réflexion sur les langages : MOF
  - adaptation et personnalisation : CWM
  - réflexion sur les processus : SPEM
  - multiplication des middleware (CORBA, EJB, SOAP, COM+, .NET...)

OMG : les activités

Deux grandes générations à l'OMG

- Avant 2000
  le modèle OMA : Object Management Architecture
  interopérabilité entre applications à objets développées sur des réseaux hétérogènes
  - CORBA 1.1 $\rightarrow$ CORBA 3.0
  - IDL
- Après 2000 : Le modèle MDA : Model Driven Approach
  fédère l'ensemble des travaux
  interopérabilité entre modèles hétérogènes
  - MDA, MOF, UML, CWM, CORBA, XMI...

OMG : les objectifs - évolution

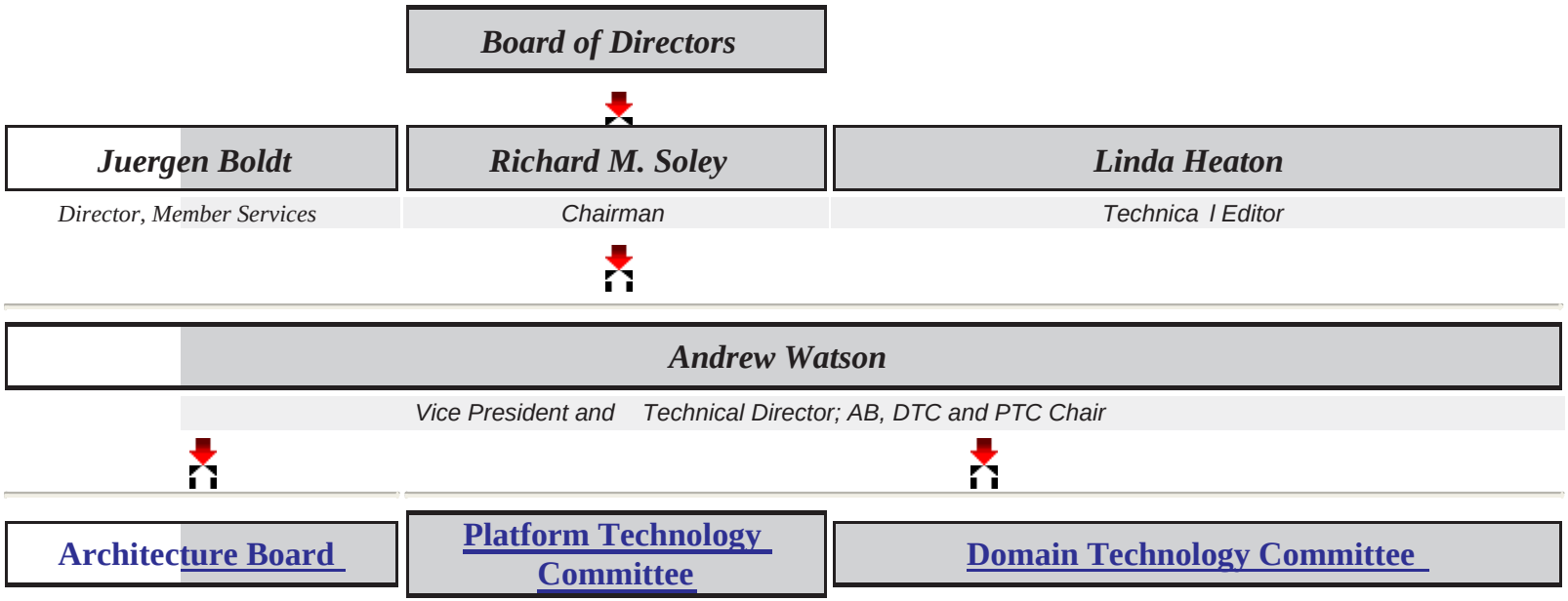


OMG : la structure (organisation)

L'organisation est régie par le document :
Policies and Procedures of the OMG Technical Process (version 2.9)
http://www.omg.org/cgi-bin/doc?pp/2009-12-01

1. Comité de direction : *board of directors* (28 sociétés en 2011)
2. Direction : un président (R.M. Soley), un administrateur et un éditeur technique
3. Des sous-directions techniques principales
4. Différents groupes de travail : Sous-comités, Task Force, Special Interest Group...

OMG : la structure (organigramme)



OMG : la structure (comité directeur)

Chairman and CEO: **Richard Soley, Ph.D.**

- Adaptive
- Atego
- CA Technologies
- CSC
- DoCoMo Communication Laboratories Europe GmbH
- FireStar Software
- Fujitsu
- HSBC
- Hewlett Packard
- Hitachi
- International Business Machines
- KDM Analytics
- Lockheed Martin
- MITRE
- Mentor Graphics Corporation
- Microsoft
- Model Driven Solutions
- No Magic, Inc.
- Northrop Grumman
- Objective Interface Systems
- Oracle
- PrismTech
- Progress Software
- Real-Time Innovations
- THALES
- TethersEnd Consulting
- Unisys
- Visumpoint

OMG Counsel: *Morse, Barnes -Brown & Pendleton, P.C.*

OMG : la structure (architecture)

- *Architecture Board* (11 sièges)
  http://www.omg.org/news/about/ab.htm
  The AB operates under a Constitution (approved by the OMG Board of Directors) that establishes its domain of operations. The AB can make changes to the published architectural documents of the OMG and it approves RFP issuances and technology adoptions.
  To perform these duties the AB has a set of less formal procedures that facilitate the flow of actions between and during OMG meetings. It also occasionally issues documents about what it expects in RFPs and submissions.

OMG : la structure (architecture)

- *Architecture Board* (11 sièges)
  http://www.omg.org/news/about/ab.htm

  - Intellectual Property Policy Subcommittee (ipr@omg.org)
  - Liaison AB Subcommittee (liaison@omg.org) - ABSC
  - Object and Reference Model AB Subcommittee (ormsc@omg.org)
  - Specification Management AB Subcommittee (smsc@omg.org)

  - Architecture Ecosystem ABSIG (architecture-ecosystem@omg.org)
  - Business Architecture AB SIG (basig@omg.org)
  - MDA Users ABSIG (mda-users@omg.org)
  - Service Oriented Architecture ABSIG (soa@omg.org)
  - Sustainability AB S IG

OMG : la structure (architecture)

- *Architecture Board* (11 sièges)
  http://www.omg.org/news/about/ab.htm
- Direction des standards
  - *Platform Technology Committee*

    The PTC concentrates on infrastructure and modeling specifications (e.g., the ORB, object services, protocols, IDL and its language mappings, UML, XMI, MOF and CWM.)

  - *Domain Technology Committee*

    The DTC concentrates on vertical market specifications (e.g., finance, healthcare, manufacturing and telecommunications.) The AB ensures that specifications developed by the other two plenaries are consistent with the OMG's architecture and existing suite of specifications.

OMG : la structure (Platform Technology Committee)

The purpose of the OMG's Platform Technology Committee (PTC) is to solicit, propose, review, recommend modifications to, recommend adoption of and maintain specifications of technology in pursuit of the goals stated in the OMG by-laws.
The principal foci of PTC activity is the specification of OMG's Model Driven Architecture (MDA); the Common Object Request Broker Architecture (CORBA) applied at the enterprise and embedded systems levels; and the Unified Modeling Language (UML), Meta Object Facility (MOF) and Common Warehouse Meatmodel (CWM) modeling technologies.

OMG : la structure (Platform Technology Committee)

- Analysis and Design PTF
- Architecture Driven Modernization PTF
- Middleware and Related Services PTF
- System Assurance PTF
- Agent PSIG
- Data Distribution Services PSIG
- Japan PSIG
- Korea PSIG
- Ontology PSIG
- Telecommunications PSIG
- Abstract Syntax Tree Metamodel (ASTM) FTF
- 2nd Lightweight Load Balancing FTF
- 2nd Model Level Testing and Debug FTF
- MOF 2 Facility and Object Lifecycle FTF
- Semantics of a Foundational …(FUML) FTF
- Software Metrics Metamodel (SMM) FTF
- UML Profile for DDS FTF
- 2nd UML Profile for MARTE FTF
- …
- MOF 2 Core RTF
- MOF M odel to Text 1.1 RTF
- MOF QVT 1.1 RTF
- OCL 2.1 RTF
- UML 2.3 RTF
- UML Profile for Systems Engineering (SysML) 1.2 RTF
- UML Profile for Voice1.1 RTF
- XMI for MOF 2 (XMI 2. 1) RTF
- …

OMG : la structure (Domain Technology Committee)

- Business Modeling and Integration DTF
- Consultation, Command, Control, Communications & Intelligence (C4I) DTF
- Emergency, Crisis and Major Event Management Domain Special Interest Group (ECMEM DSIG)
- Finance DTF
- Government DTF
- Healthcare DTF
- Life Sciences Research DTF
- Manufacturing Technology and Industrial Systems DTF
- Robotics DTF
- Software-Based Communications DTF
- Space DTF
- Mathematical Formalism SIG
- Regulatory Compli ance DSIG
- Super Distributed Objects DSIG
- Systems Engineering DSIG

Revision Task Force (RTF)

Finalization Task Force (FTF)

OMG : les méthodes de travail

Les méthodes de travail sont basées sur une répartition des rôles pour les groupes identifiés dans la structuration de l'organisation et un processus d'adoption des normes.

La définition des rôles et les procédures d'adoption des spécifications sont régies par le document :
Policies and Procedures of the OMG Technical Process (version 2.9)
http://www.omg.org/cgi-bin/doc?pp/2009-12-01

Une version technique du processus d'adoption est décrite dans le document :
The OMG Hitchhiker's Guide A Handbook for the OMG Technology Adoption Process (version 7.8)
http://www.omg.org/cgi-bin/doc?omg/08-09-02

OMG : le processus de normalisation 1/2

- Identification du problème.
  Le Comité Technologique TC (*Technology Committee*) charge un groupe de travail TF (*task force*) de faire des recommandations dans un domaine technologique particulier.
  Seuls les membres influents peuvent les proposer.
- Creation de la RFP (*request for proposal*)
- Approbation de la RFP
- Soumissions de la RFP
- Finalisation de la RFP
- Post-adoption de la RFP

OMG : le processus de normalisation 1/2

- Identification du problème.
- Creation de la RFP (*request for proposal*)
  Le groupe de travail élabore une RFP avec éventuellement et au préalable une étude RFI (*request for information*). La RFI viser à collecter des informations dans l'industrie. La RFP aboutit ensuite à l'élaboration d'une proposition de norme soumise à l'OMG.
- Approbation de la RFP
- Soumissions de la RFP
- Finalisation de la RFP
- Post-adoption de la RFP

OMG : le processus de normalisation 1/2

- Identification du problème.
- Creation de la RFP (*request for proposal*)
- Approbation de la RFP
  La RFP est soumise à l'AB (*Architecture Board*), aux TFs et aux TC, qui après étude et modifications votent la recommandation de la spécification.
- Soumissions de la RFP
- Finalisation de la RFP
- Post-adoption de la RFP

OMG : le processus de normalisation 1/2

- Identification du problème.
- Creation de la RFP (*request for proposal*)
- Approbation de la RFP
- Soumissions de la RFP
  Les membres peuvent répondre à la RFP par une lettre d'intention LOI (*letter of intent*) puis une soumission initiale. Ces soumissions sont ensuites revues jusqu'à obtenir une soumission finale votée.
- Finalisation de la RFP
- Post-adoption de la RFP

OMG : le processus de normalisation 1/2

- Identification du problème.
- Creation de la RFP (*request for proposal*)
- Approbation de la RFP
- Soumissions de la RFP
- Finalisation de la RFP
  La finalisation est assurée par la FTF (*finalization task force*) qui rend disponible la norme.
- Post-adoption de la RFP

OMG : le processus de normalisation 1/2

- Identification du problème.
- Creation de la RFP (*request for proposal*)
- Approbation de la RFP
- Soumissions de la RFP
- Finalisation de la RFP
- Post-adoption de la RFP
  La révision est assurée par la RTF (*revision task force*) qui efectue des modifications mineures.

OMG : le processus de normalisation 1/2

- Identification du problème.
- Creation de la RFP (*request for proposal*)
- Approbation de la RFP
- Soumissions de la RFP
- Finalisation de la RFP
- Post-adoption de la RFP

Ceci est une version simplifiée du processus.

OMG : le processus de normalisation 2/2

- La demande d'information RFI (*request for information*) suit un processus similaire à la RFP mais allégé (demande, approbation, réponse, évaluation).

OMG : le processus de normalisation 2/2

- La demande d'information RFI (*request for information*) suit un processus similaire à la RFP mais allégé (demande, approbation, réponse, évaluation).
- La demande de commentaire RFC (*request for comment*) est une procédure de reconnaissance d'une technologie existante, par l'OMG. Elle émane non pas d'une demande de l'OMG mais d'un industriel. La RFC suit un processus similaire à la la fin du processus RFP.

OMG : le processus de normalisation 2/2

- La demande d'information RFI (*request for information*) suit un processus similaire à la RFP mais allégé (demande, approbation, réponse, évaluation).
- La demande de commentaire RFC (*request for comment*) est une procédure de reconnaissance d'une technologie existante, par l'OMG. Elle émane non pas d'une demande de l'OMG mais d'un industriel. La RFC suit un processus similaire à la la fin du processus RFP.
- Il existe aussi une procédure pour retirer une norme.

Normes : la première génération

- The Object Management Architecture (OMA)

  is a set of standard interfaces for standard objects that support CORBA applications. It includes the base-level CORBAservices, the CORBAfacilities, and a large and growing set of Domain Specifications.

Normes : la première génération

- The Object Management Architecture (OMA)
- CORBA - the Common Object Request Broker Architecture

  is OMG's showcase specification for application interoperability independent of platform, operating system, programming language - even of network and protocol. CORBA includes a number of specifications that you may have heard about separately: OMG Interface Definition Language (OMG IDL), the network protocols GIOP and IIOP, an infrastructure for server-side scalability termed the POA (for Portable Object Adapter), and the CORBA Component Model (CCM). The CCM integrates Enterprise Java Beans, and a mapping to XML provides the most robust support in the industry for XML document usage and interoperability.

Normes : la première génération

- The Object Management Architecture (OMA)
- CORBA - the Common Object Request Broker Architecture
- UML - the Unified Modeling Language

  standardizes representation of object oriented analysis and design. A graphical language, its dozen diagram types include Use Case and Activity diagrams for requirements gathering, Class and Object diagrams for design, and Package and Subsystem diagrams for deployment. UML lets architects and analysts visualize, specify, construct, and document applications in a standard way.

Normes : la seconde génération

- MOF - The MetaObject Facility

  standardizes a metamodel for object oriented analysis and design, and a repository. (The CWM standardizes a metamodel for data modeling; look two paragraphs down.) Because they are based on the MOF metamodel, UML models can be freely passed from tool to tool using XMI - without the commonality of definition provided by the MOF, this would not be practical.

Normes : la seconde génération

- MOF - The MetaObject Facility
- CWM - The Common Warehouse Metamodel

  standardizes a basis for data modeling commonality within an enterprise, across databases and data stores. Building on a foundation metamodel, it adds metamodels for relational, record, and multidimensional data; transformations, OLAP, and data mining; and warehouse functions including process and operation. CWM maps to existing schemas, supporting automated schema generation and database loading. This makes it the basis for data mining and OLAP across the enterprise.

Normes : la seconde génération

- MOF - The MetaObject Facility
- CWM - The Common Warehouse Metamodel
- XMI - XML Metadata Interchange

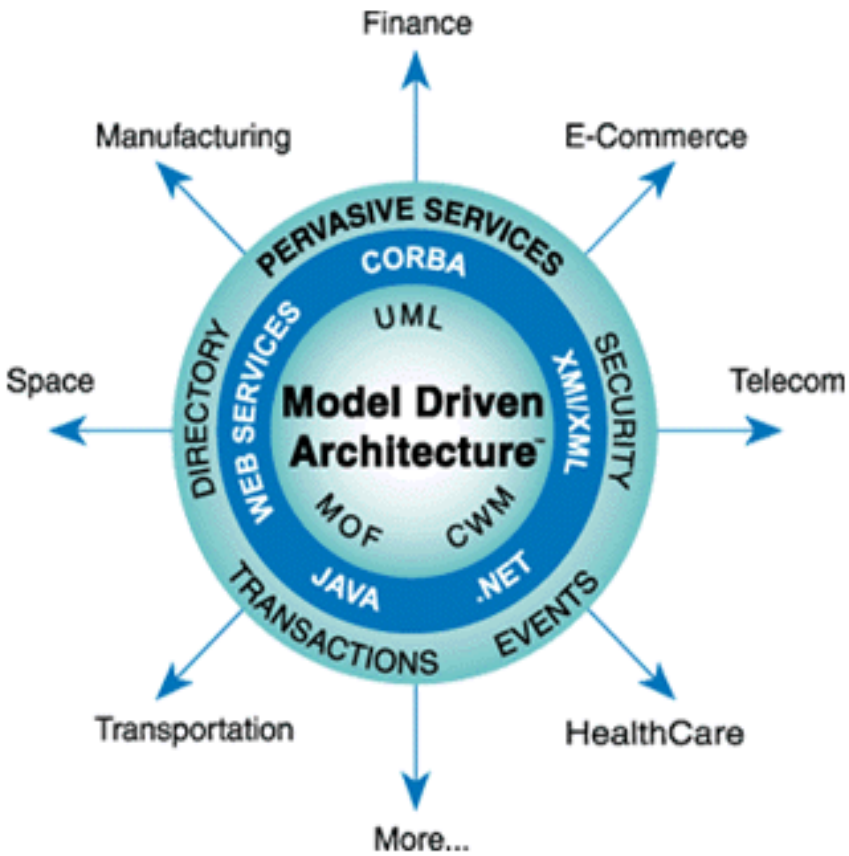
  allows MOF-compliant metamodels (and therefore models, since a model is just a special case of a metamodel) to be exchanged as XML datasets. Both application models (in UML) and data models (in CWM; see below) may be exchanged using XMI. In addition to allowing model exchange, XMI serves as a mapping from UML and CWM to XML.

Normes : la seconde génération

- MOF - The MetaObject Facility
- CWM - The Common Warehouse Metamodel
- XMI - XML Metadata Interchange
- MDA - The Model Driven Architecture.

  Unifying the Modeling and Middleware spaces, OMG's MDA supports applications over their entire lifecycle from Analysis and Design, through implementation and deployment, to maintenance and evolution. Based on UML models which remain stable as the technological landscape changes around them, MDA-based development maximizes software ROI as it integrates applications across the enterprise, and one enterprise with another. Adopted by members as the basis for OMG specifications in September, 2001, the MDA is truly a unique advance in distributed computing.

Normes : unification par MDA



OMG : liste des spécifications de références

Modeling and Metadata Specifications

SPECIFICATION	acronym	Version	Document #
Action Language for Foundational UML	ALF	1.0 - Beta 1	ptc/2010 -10-05
Common Warehouse Metamodel	CWM	1.1	formal/2003 -03-02
CWM Metadata Interchange Patterns	MIPS	1.0	formal/2004 -03-25
Diagram Definition	DD	1.0 - Beta1	ptc/2010 -12-18
Meta Object Facility Core	MOF	2.0	formal/2006 -01-01
Model Driven Message Interoperability	MDMI	1.0	formal/2010 -03-01
Model -level Testing and Debugging	MLTD	1.0 Beta 1	ptc/2007 -05-14
MOF Model to Text Transformation Language	MOFM2T	1.0	formal/2008 -01-16
MOF Query / Views / Transformations	QVT	1.1 Beta 2	ptc/2009 -12-05
MOF Support for Semantic Structures	SMOF	1.0 - Beta 1	ptc/2010 -11-39
MOF 2 Facility and Object Lifecycle	MOFFOL	1.0	formal/2010 -03-04
MOF 2 Versioning and Development Lifecycle	MOFVD	2.0	formal/2007 -05-01
Object Constraint Language	OCL	2.2	formal/2010 -02-01
OMG Systems Modeling Language	SysML		
Ontology Def inition Metamodel	ODM	1.0	formal/2009 -05-01
Reusable Asset Specification	RAS	2.2	formal/2005 -11-02
Semantics of a Foundational for Executable UML Models	FUML	1.0 Beta 1	ptc/2008 -11-03
Service oriented architecture Modeling Language	SoaML	1.0 Beta 2	ptc/2009 -12-09
Software Process Engineering Metamodel	SPEM	2.0	formal/2008 -04-01
Unified Modeling Language			
- Infrastructure	UML	2.3	formal/2010 -05-03
- Superstructure	UML	2.3	formal/2010 -05-05
UML Diagram Interchange	UMLDI	1.0	formal/2006 -04-04
UML Human -Usable Textual Notati on	HUTN	1.0	formal/2004 -08-01
XML Metadata Interchange	XMI	2.1.1	formal/2007 -12-01

OMG : liste des spécifications de références

Modeling and Metadata Specifications - profils

SPECIFICATION	acronym	Version	Document #
OMG Systems Modeling Language	SysML		
PIM and PSM for Smart Antenna	smartant		
UML Profile for CCM	CCMP	1.0	formal/2005 -07-06
UML Profile for CORBA and CCM	CCCMP	1.0	formal/2008 -04-07
UML Profile for Data Distribution	UML4DDS	1.0 Beta 1	ptc/2008 -07-02
UML Profile for Enterprise Application Integration	EAI	1.0	formal/2004 -03-26
UML Profile for Enterprise Distributed Object Computing	EDOC	1.0	formal/2004 -02-01
UML Profile for Modeling and Analysis of Real-time and Embedded Syste ms	MARTE	1.0 Beta 2	ptc/2008 -06-09
UML Profile for QoS and Fault Tolerance	QFTP	1.1	formal/2008 -04-05
UML Profile for Schedulability, Performance and Time	SPTP	1.1	formal/2005 -01-02
UML Profile for System on a Chip	SoCP	1.0.1	formal/2006 -08-01
UML Profile for Software Radio	SDRP	1.0	formal/2007 -03-01
UML Profile for Voice	VOICP	1.0	formal/2008 -04-06
UML Testing Profile	UTP	1.0	formal/2005 -07-07

http://www.omg.org/technology/documents/spec_summary.htm

Normes : celles qui nous concernent

- UML - the Unified Modeling Language et OCL - the Object Constraint Language
  les langages de modélisation
- XMI - XML Metadata Interchange
  le format d'échange de modèles
- MOF - The MetaObject Facility
  les règles de modélisation
- MDA - The Model Driven Architecture.
  le cadre global

Noter qu'une partie des normes est reprise -liaison- par d'autres organismes (ISO, W3C...).

Plan

OMG : normes de référence

1. UML
2. OCL
3. AS — *supposé connus* $\Rightarrow$ *synthèse rapide*
4. MOF
5. XMI
6. QVT

http://www.omg.org/technology/documents/modeling_spec_catalog.htm

UML : état actuel

Specification Name:	**Unified Modeling Language™ (UML®)**
NOTE: Version 2.0 does not have XSD or XML associated files due to structural problems with the UML metamodel.	
Description:	A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.
Keywords:	abstraction, action sequence, action state, activity graph, architecture, association, class diagram, collaboration diagram, component diagram, control flow, data flow, deployment diagram, execution, implementation, pins, procedure.
Latest / past specifications:	Current version: 2.3 Past versions
Related OMG Specifications:	Diagram Interchange, OCL, MOF, XMI
Related Industry Standards:	UML 1.4.2 is available as ISO/I EC 19501; 19505 assigned to UML 2.1.2; ITU-T Recommendations Z.100 (SDL) and Z.109 (SDL UML profile).
Most recent IPR and Implementation questionnaire responses:	BCQ, Borland, IBM, Kabira, Oracle, Softeam, Sparx Systems, THALES

http://www.omg.org/spec/UML/2.3/

UML : langage de spécification

- Notation complète et extensible
- Structuration des concepts
- Diagrammes
- Modèles et vues
- Multi-formalisme
- Classification

UML : une notation complète et extensible

- Complète
  UML inclut un grand nombre de concepts autour de

UML : une notation complète et extensible

- Complète
  UML inclut un grand nombre de concepts autour de
  - l'objet : objets, classes, opérations, attributs, relations, envois de message, etc.

UML : une notation complète et extensible

- Complète
  UML inclut un grand nombre de concepts autour de
  - l'objet : objets, classes, opérations, attributs, relations, envois de message, etc.
  - l'analyse des besoins : acteurs, cas d'utilisation,

UML : une notation complète et extensible

- Complète
  UML inclut un grand nombre de concepts autour de
  - l'objet : objets, classes, opérations, attributs, relations, envois de message, etc.
  - l'analyse des besoins : acteurs, cas d'utilisation,
  - la conception du logiciel : composants, modules, processus,

UML : une notation complète et extensible

- Complète
  UML inclut un grand nombre de concepts autour de
  - l'objet : objets, classes, opérations, attributs, relations, envois de message, etc.
  - l'analyse des besoins : acteurs, cas d'utilisation,
  - la conception du logiciel : composants, modules, processus,
  - l'implantation : nœuds, liaisons, déploiement.

UML : une notation complète et extensible

- Complète
  UML inclut un grand nombre de concepts autour de
  - l'objet : objets, classes, opérations, attributs, relations, envois de message, etc.
  - l'analyse des besoins : acteurs, cas d'utilisation,
  - la conception du logiciel : composants, modules, processus,
  - l'implantation : nœuds, liaisons, déploiement.

  $\Rightarrow$ décrire des modèles couvrant l'ensemble du cycle de développement.

UML : une notation complète et extensible

- Complète
  UML inclut un grand nombre de concepts autour de
  - l'objet : objets, classes, opérations, attributs, relations, envois de message, etc.
  - l'analyse des besoins : acteurs, cas d'utilisation,
  - la conception du logiciel : composants, modules, processus,
  - l'implantation : nœuds, liaisons, déploiement.

  $\Rightarrow$ décrire des modèles couvrant l'ensemble du cycle de développement.
- Extensible
  UML autorise l'enrichissement ou la personnalisation de la notation au moyen des stéréotypes.

UML : structuration des concepts

- Paquetages : point de vue utilisateur
- Diagrammes : point de vue langage $\Rightarrow$ la seule normalisée
- Modèles : point de vue méthode
- Vues : point de vue méthode

La notation UML : diagrammes 1/3

UML 1.x propose neuf types de combinaisons cohérentes et complémentaires : les **diagrammes**.

- Les diagrammes de cas d'utilisation (UC - *Use Case*) décrivent les acteurs et l'utilisation du système.
- Les diagrammes de classes représentent les classes et les relations statiques entre ces classes : classe, attribut, opération, visibilité, interface, association, agrégation, héritage, dépendance...
- Les diagrammes d'objets (pas toujours considéré comme un diagramme) décrivent des objets et des liens. Les objets peuvent être actifs et définir leur flot de contrôle. Sur ces liens (réels ou virtuels) circulent des messages. Les envois de messages sont synchrones ou asynchrones, avec ou sans résultats.

La notation UML : diagrammes 2/3

- Les diagrammes d'objets se retrouvent sous deux formes dans UML :
  - Les diagrammes de séquence, qui donnent une vision temporelle des interactions en objets en mettant l'accent sur l'ordonnancement des échanges entre objets.
  - Les diagrammes de collaboration, qui donnent une vision spatiale des interactions en mettant l'accent sur les liaisons entre objets.

La notation UML : diagrammes 3/3

- Les diagrammes états-transitions modélisent le comportement des objets au cours du temps.
- Les diagrammes d'activités décrivent le flot de contrôle interne aux opérations. A grande échelle, ils représentent aussi les échanges entre objets.
- Les diagrammes de composants mettent en évidence les composants d'implémentation et leurs relations.
- Les diagrammes de déploiement définissent la structure matérielle et la distribution des objets et des composants.

En plus : stéréotypes, paquetages, notes, contraintes.

La notation UML2

Une notation encore plus complète : 13 diagrammes

- Objets et de Paquetages deviennent des diagrammes à part entière
- Diagramme de collaboration devient (une fois simplifié) Diagramme de communication
  La collaboration devient un élément des structures composites.
- Les diagrammes de structures composites placent la hiérarchie de composition au premier plan avec une nette orientation composants et architecture de logiciels (ADL).
- ...

La notation UML2

- ...
- Les diagrammes d'interaction sont un mélange d'activités et de séquence.
- Les diagrammes de temps (*timing*) permettent la description d'évolution temporelle usuelle en génie électrique.

Par ailleurs, les diagrammes d'activités sont fortement enrichis pour inclure les DFD.

Tour d'horizon de la notation UML

Exemples voir le cours UML de M1

La notation UML : modèles et vues

Les diagrammes décrivent des aspects complémentaires mais non disjoints du système.
Un modèle est un ensemble de diagrammes et de documents.

- Modèle des besoins, d'analyse, de conception, de réalisation, d'implantation, de déploiement, de test...
- Modèle statique, modèle dynamique, modèle fonctionnel
- Modèle de type, modèle d'instances...
- Méta-modèle
- etc.

Les vues sont des aspects des modèles. Les modèles ne dépendent pas de la notation mais de la méthode de développement.

La notation UML : multi-formalisme

Les relations entre diagrammes sont de plusieurs types :

- multi-aspect : classes, états-transitions, activités
- type/instance : classes et objets, UC et scénarios
- abstraction : classes et composant, classes d'analyse et classe d'implantation
- complémentaires : classes et UC
- chevauchement : séquences et collaborations, activités et états-transitions

Les concepts et notations sont variés : l'objet assure la cohésion sémantique.

La notation UML : classification

Les concepts (et diagrammes) peuvent être classés selon plusieurs axes :

- activité de développement : besoins, analyse, conception, réalisation
- abstraction : méta-type/type/instance : définitions, exemplaires
- aspect du système : statique, dynamique, fonctionnel
- degré de formalisme : cas d'utilisation, classe, états-transition...

Chaque axe reprend une préoccupation des acteurs du développement (méthodologiste, analyste, utilisateur, développeur...).

La notation UML : en résumé

UML est un langage complet mais complexe.

Contrairement à UML 1.x, dans UML 2.0 la sémantique est plus modulaire mais le recouvrement des diagrammes interdit toute sémantique commune.

OCL : état actuel

Specification Name:	**Object Constraint Language (OCL)**
Description:	Specifies the Object Constraint Language (OCL), a formal language used to describe expressions on UML models. These expressions typically specify invariant conditions that must hold for th e system being modeled or queries over objects described in a model. Note that when the OCL expressions are evaluated, they do not have side effects (i.e., their evaluation cannot alter the state of the corresponding executing system). OCL expressions can be used to specify operations / actions that, when executed, do alter the state of the system. UML modelers can use OCL to specify application -specific constraints in their models. UML modelers can also use OCL to specify queries on the UML model, which a re completely programming language independent.
Keywords:	
Latest / past specifications:	Current version: 2.2 Past versions: 2.0
Related OMG Specification s:	MOF, UML, XMI
Related Industry Standards:	
Most recent IPR and Implementation questionnaire responses:	BCQ Borland IBM

http://www.omg.org/spec/OCL/2.2/

Object Constraint Language.

Eléments clés

- Langage à objets déclaratifs (relativement) formel
- Inspiré de Syntropy (et donc de Z)
- Typage
- Navigation
- Assertions et contraintes
- Méta Object Protocol

Object Constraint Language.

Détails dans le chapitre 9 [AV01b]

- Types de base et MOP ⇒ section 2
- Navigation ⇒ section 3.2

Object Constraint Language

UML : Action Semantics

Intégré à UML depuis UML 1.5

	Product Plans	
Companies/Organizations with Veto Power	**Beta Available**	**General Available**
Kabira		
Telelogic AB	2001 (partly supported in current version)	2002
Rational Software, Inc.	3Q2002	4Q2002
Veto Power Expires	May 15, 2003	

réponse au RFP : http://www.kc.com/as_site/download/ActionSemantics.zip

UML1.4/AS : http://www.omg.org/cgi-bin/doc?ptc/02-01-09

UML : Action Semantics

- Un méta-modèle détaillé
- Pas de syntaxe concrète normalisée mais des "implantation"
  - le BridgePoint Action Language (AL)
    www.projtech.com.
  - le Kabira Action Semantics (Kabira AS)
    www.kabira.com
  - xUML (Kennedy-Carter) est un sous-ensemble d'UML
    compatible avec AS www.kc.com
  - un sous-ensemble de SDL
    ITU-T Recommendations Z.100 (SDL) and Z.109 (SDL UML profile)

OMG : normes de référence

1. MOF
2. XMI
3. QVT

MOF : état actuel

Specification Name:	**Meta Object Facility (MOF™)**			
Description:	MOF is an extensible model driven integration framework for defining, manipulating and integrating metadata and data in a platform independent manner. MOF -based standards are in use for integrat ing tools, applications and data.			
Keywords:	metadata, meta -model, modeling			
Latest / past specifications:	Current version: 2.0		Past versions	
Associated documen ts:	MOF 2 XMI, XML Schema, etc.			
Revision Information:	Status: 2.1 revision underway	Working Document: Version 2.0	Contact: MOF 2.0 Core RTF	
Related OMG Specifications:	Components , CWM, MOF QVT , MOF Versioning , UML, XMI			
Related Industry Standards:	ISO/IEC 19470:2005			
Most recent IPR and Implementation questionnaire responses:	BCQ	Ceira	IBM	MetaMatrix Sun Microsystems

http://www.omg.org/spec/MOF/2.0/PDF

MOF : les points clés 1/2

MetaObject Facility

- est un langage de description normalisé de modèles (type BNF)
- unifie les présentations de modèles à objets (Corba, UML)
- sous-ensemble d'UML : méta-modèle réflexif
- nombreux enrichissements : réflexion, OCL, JMI, SPEM...

MOF : les points clés 2/2

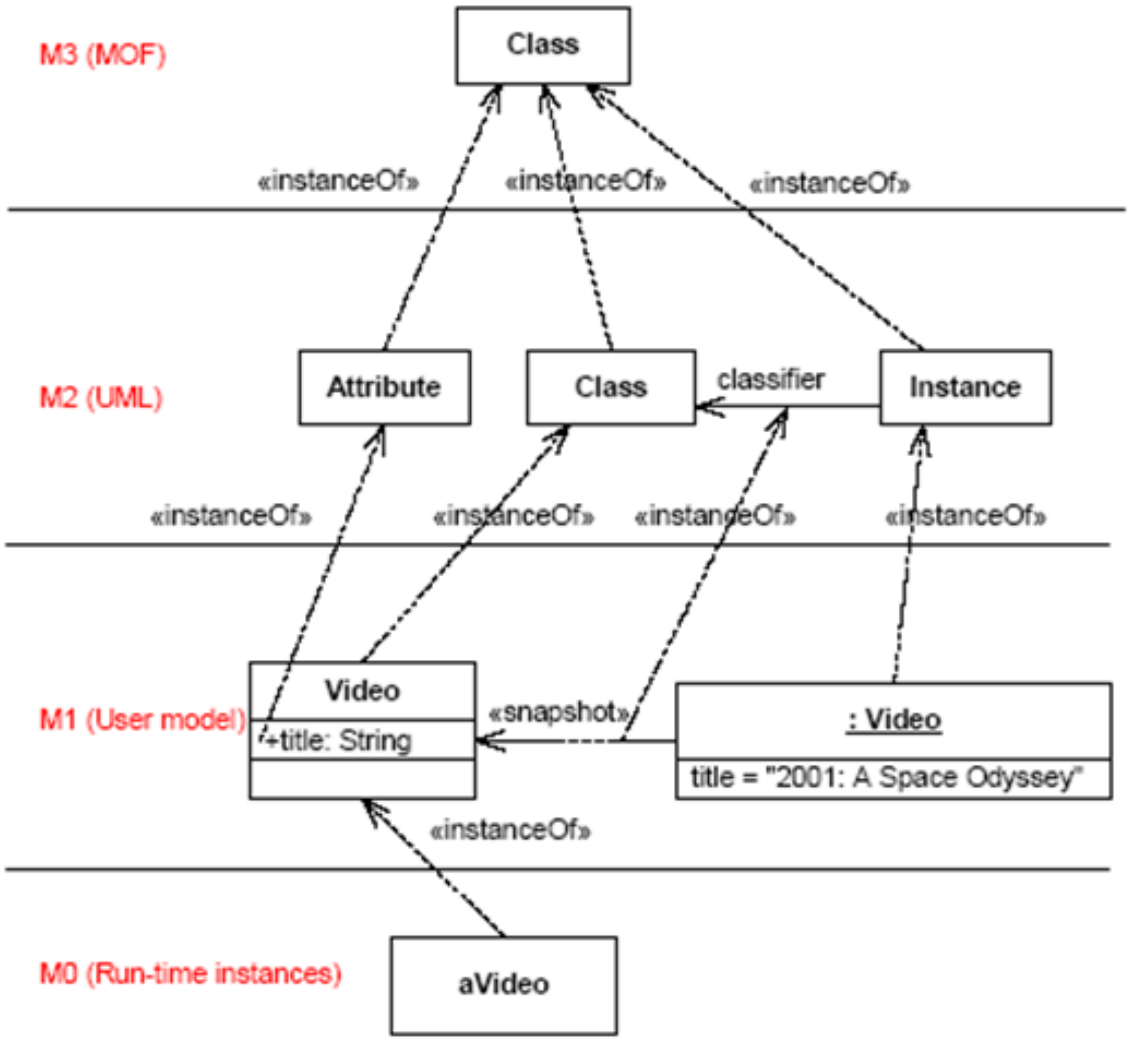
MetaObject Facility

- existe depuis 1997
- standardise le travail des fournisseurs d'outils compatibles OMG
- est à la base de l'architecture en 4 niveaux
- devient un pilier de l'architecture MDA

MOF : une architecture en 4 couches

- concepts de base :
  les réseaux sémantiques (les graphes conceptuels)
  = des concepts + des relations entre concepts + spécialisation
- architecture basée sur la relation d'instanciation :
  chaque élément d'une couche est une instance d'un élément de la couche supérieure
- modèle réflexif :
  le niveau supérieur est décrit à partir de lui-même
- architecture incrémentale :
  l'ajout de nouveaux langages doit limiter les perturbations des couches supérieures
- unifie (fédère) les langages de l'OMG

MOF : une architecture en 4 couches



MOF : les niveaux vis-à-vis d'UML

Les éléments d'un niveau

- M3 : méta-méta-modèle (d'UML) ou modèle du MOF
  décrit les éléments d'un langage de modèles UML, Merise...
- M2 : méta-modèle (d'UML) ou MOF
- M1 : modèle (d'UML) ou modèle de l'application
- M0 : instance d'un modèle (d'UML) ou système

MOF : les niveaux vis-à-vis d'UML

Les éléments d'un niveau

- M3 : méta-méta-modèle (d'UML) ou modèle du MOF
- M2 : méta-modèle (d'UML) ou MOF
  décrit les éléments d'un langage de modélisation.
- M1 : modèle (d'UML) ou modèle de l'application
- M0 : instance d'un modèle (d'UML) ou système

MOF : les niveaux vis-à-vis d'UML

Les éléments d'un niveau

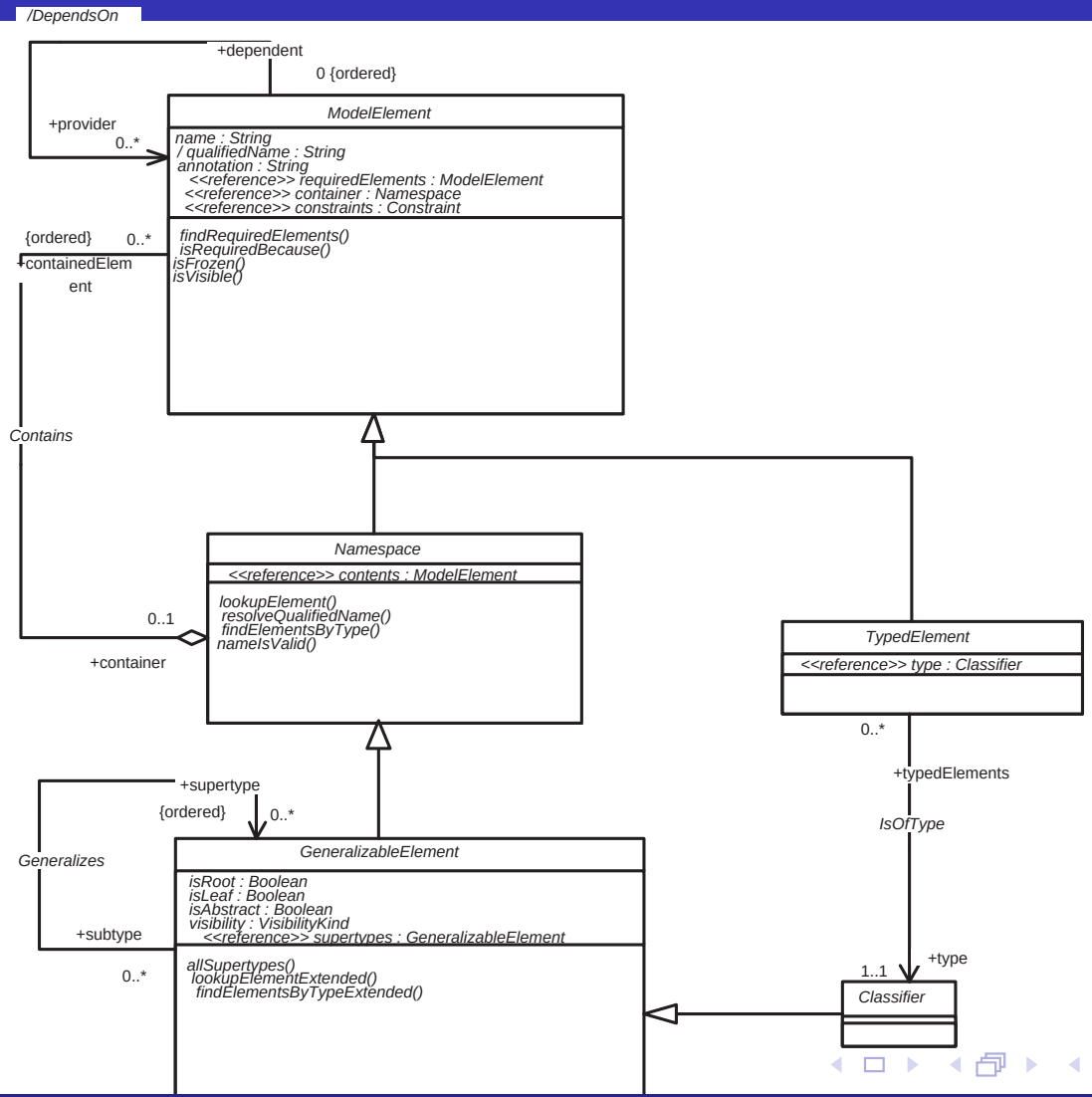
- M3 : méta-méta-modèle (d'UML) ou modèle du MOF
- M2 : méta-modèle (d'UML) ou MOF
- M1 : modèle (d'UML) ou modèle de l'application décrit un modèle de système
- M0 : instance d'un modèle (d'UML) ou système

MOF : les niveaux vis-à-vis d'UML

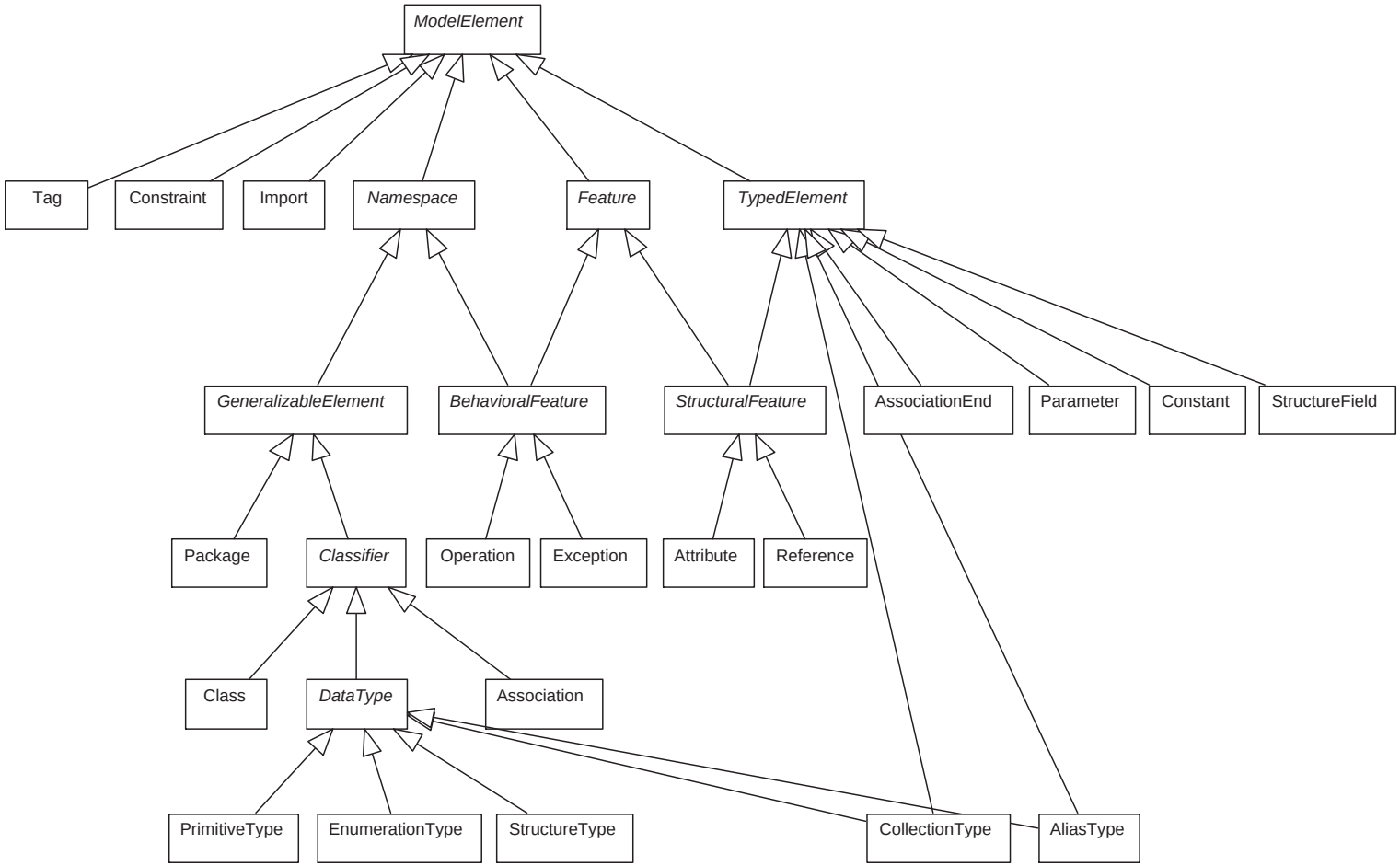
Les éléments d'un niveau

- M3 : méta-méta-modèle (d'UML) ou modèle du MOF
- M2 : méta-modèle (d'UML) ou MOF
- M1 : modèle (d'UML) ou modèle de l'application
- M0 : instance d'un modèle (d'UML) ou système
  instances du système

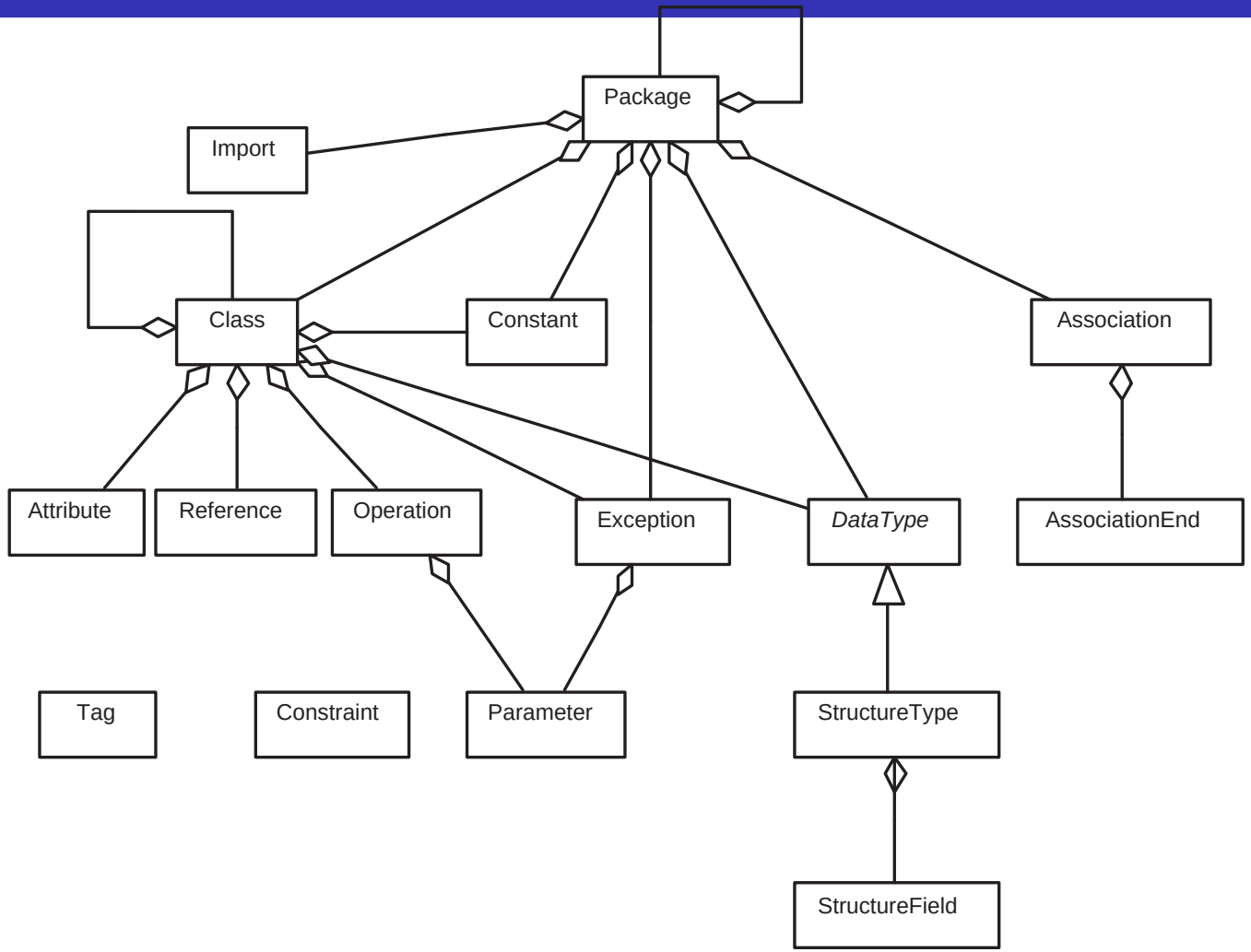
MOF : éléments clés



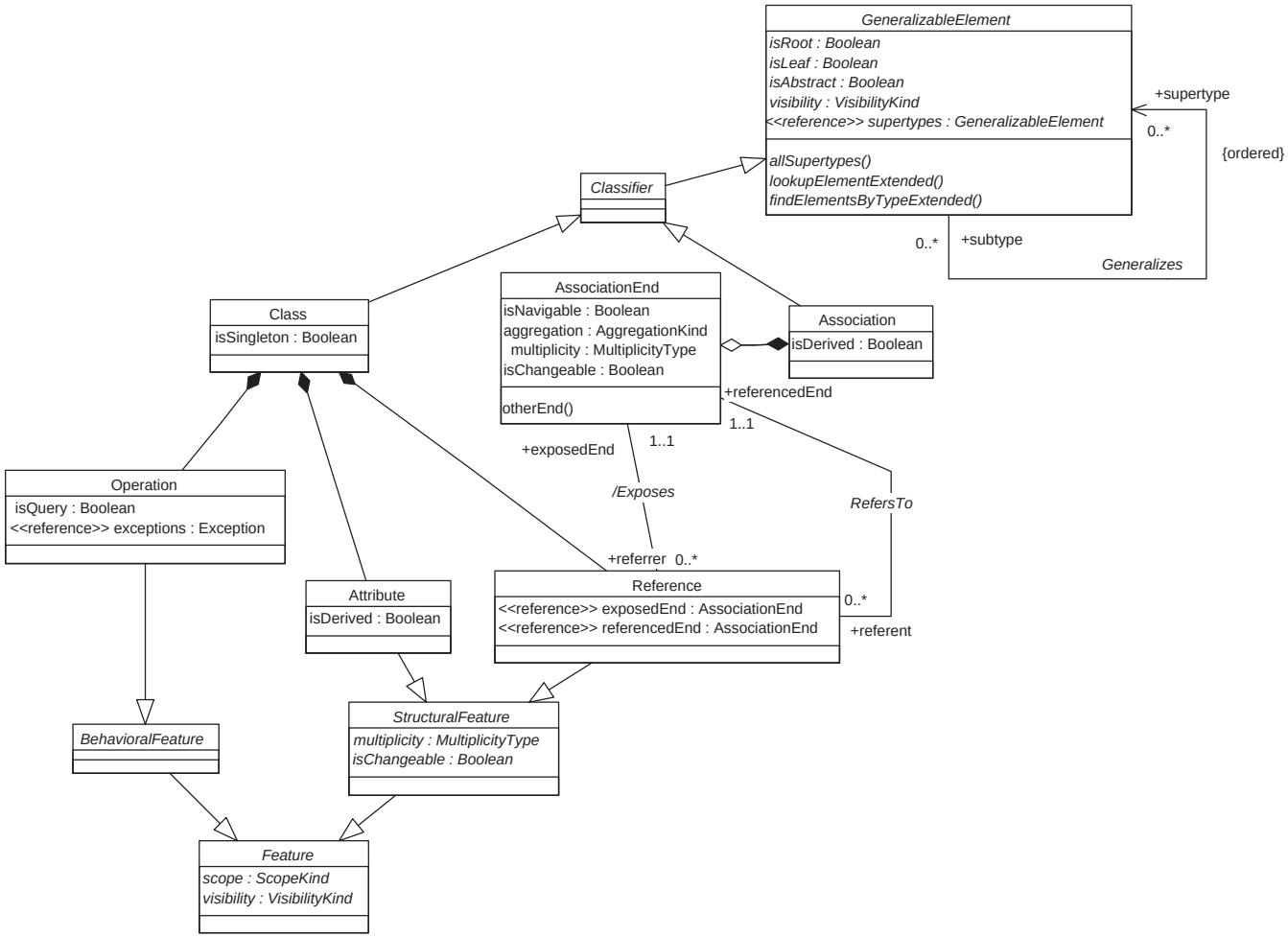
MOF : vue d'ensemble - spécialisation



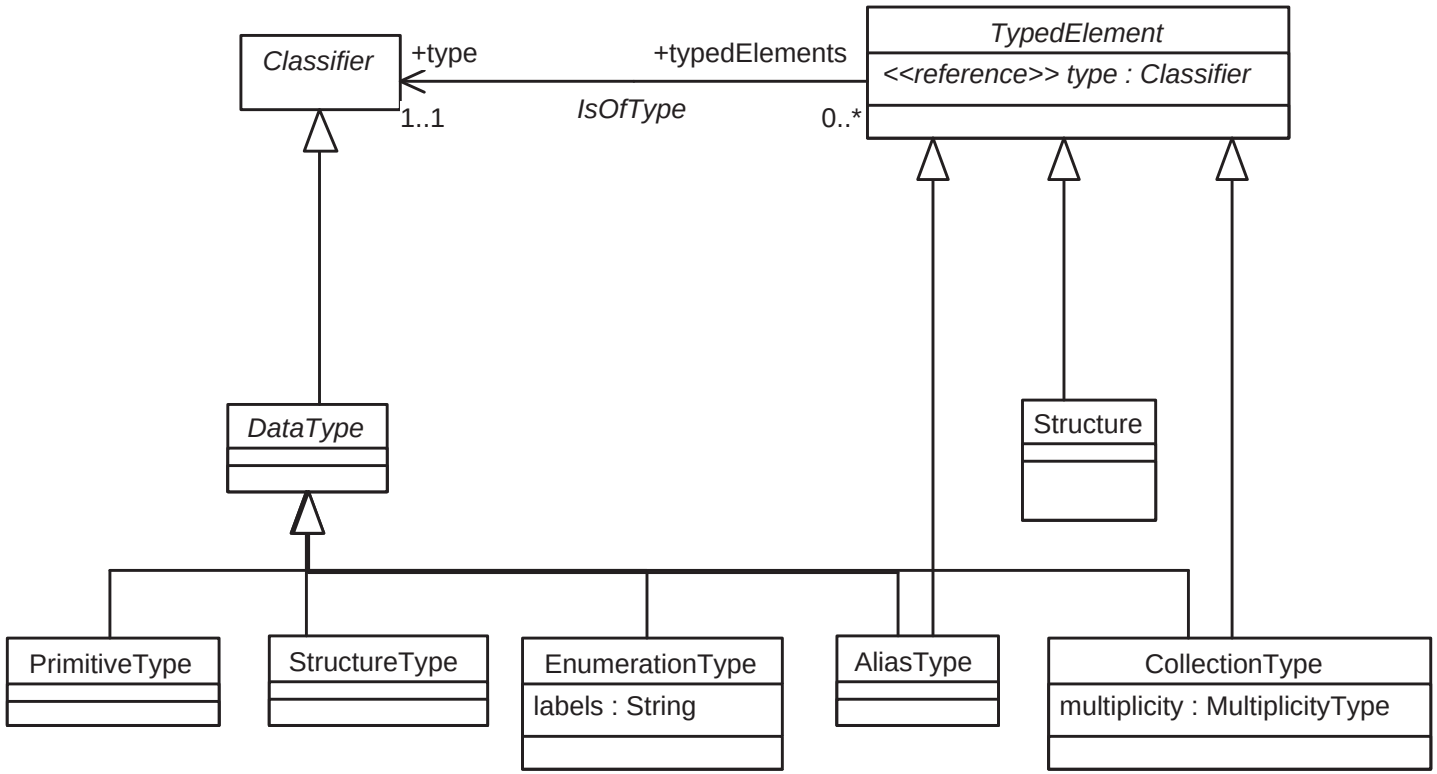
MOF : vue d'ensemble - inclusion



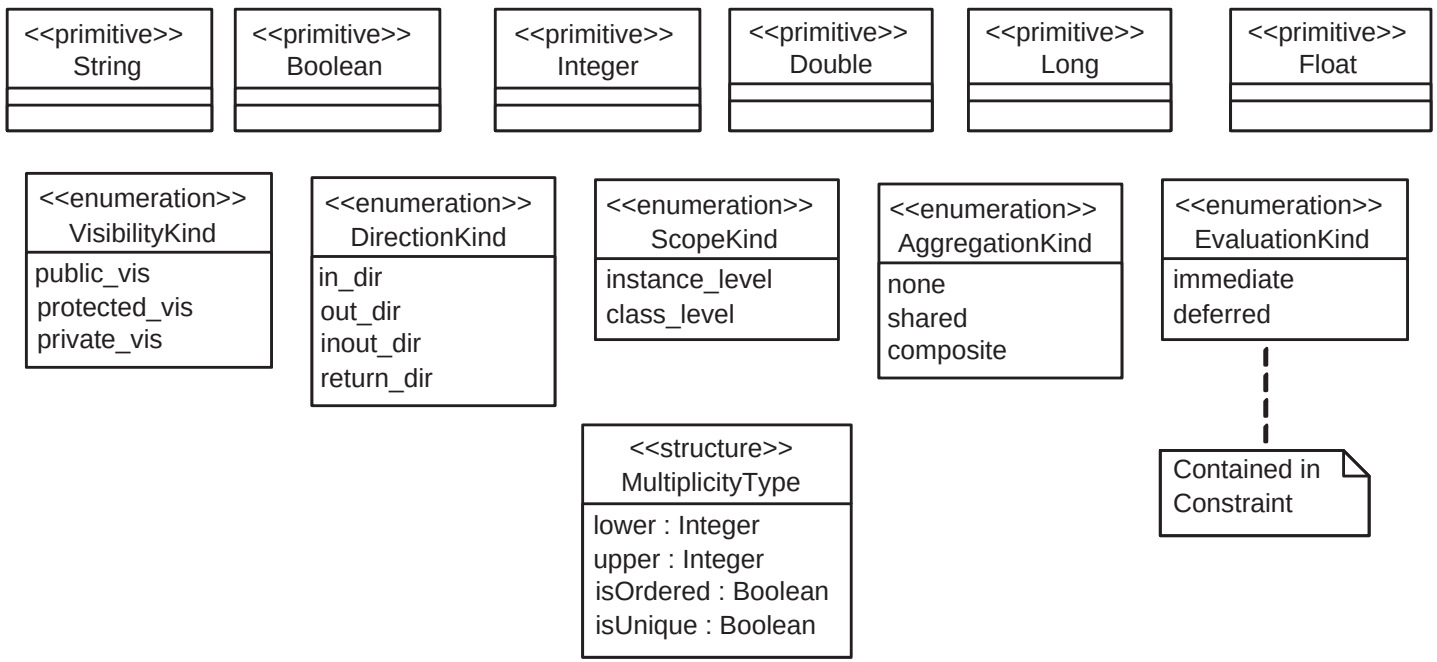
MOF : classes et associations



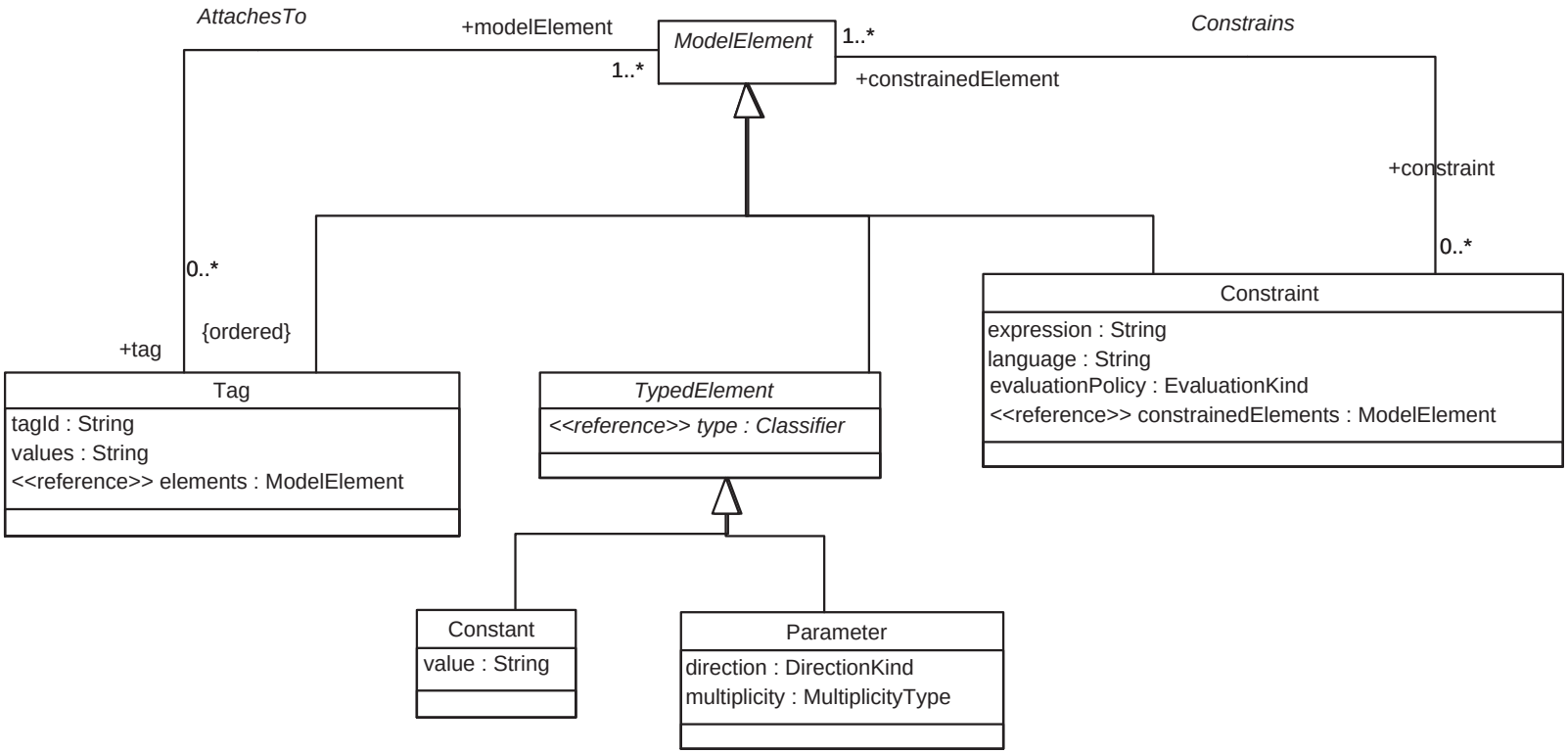
MOF : types de données 1/2



MOF : types de données 2/2



MOF : tag et contraintes



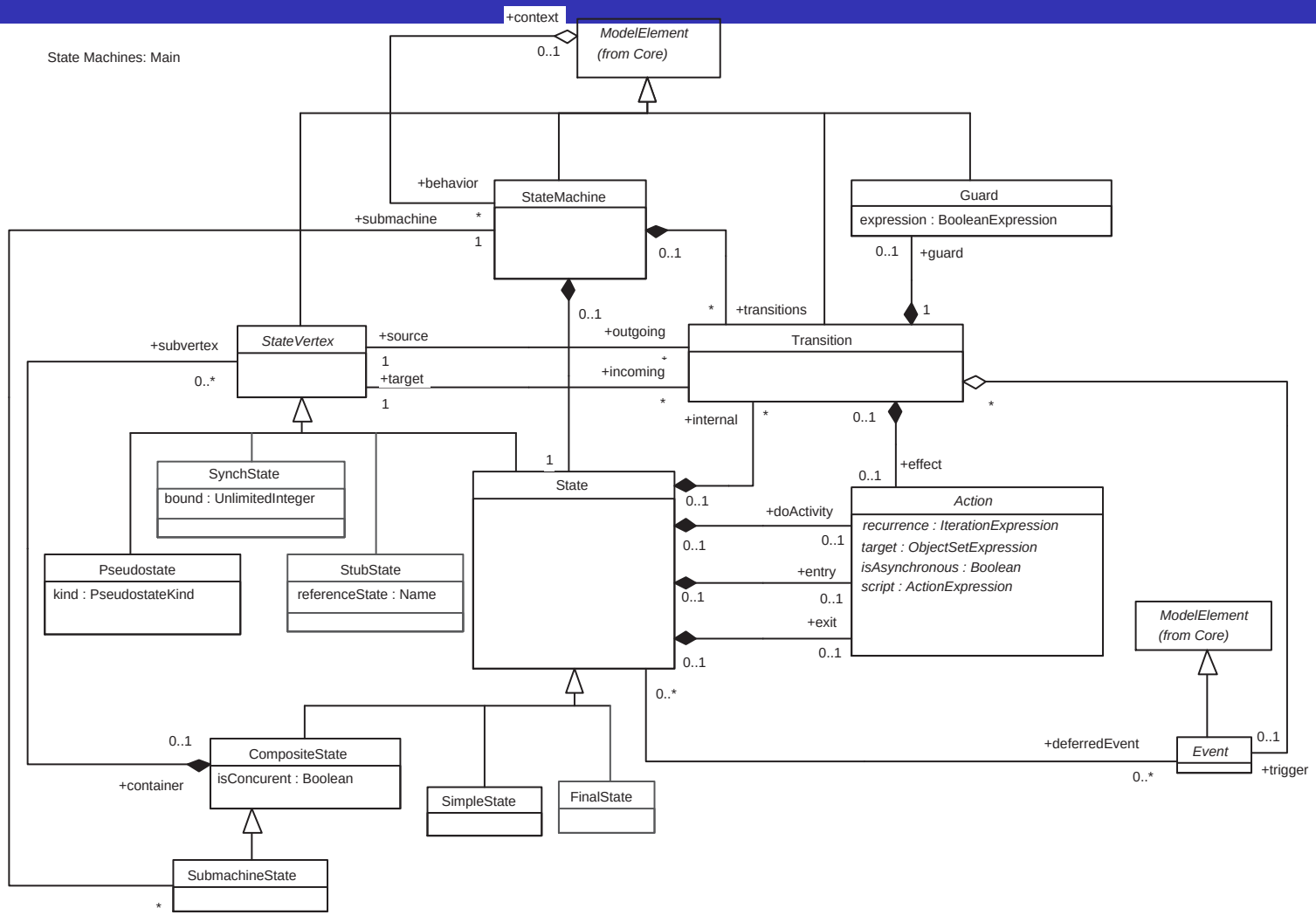
Exercice

Exercice 1 (UML-MOF)
Comparer les modèles UML et MOF.

Exercice 2 (UML-MOF)
Représenter le méta-modèle des automates en MOF.

Exercice 3 (UML)
Représenter les automates du commutateur selon le méta-modèle des automates.

MOF : métamodèle des automates



MOF : automate du commutateur

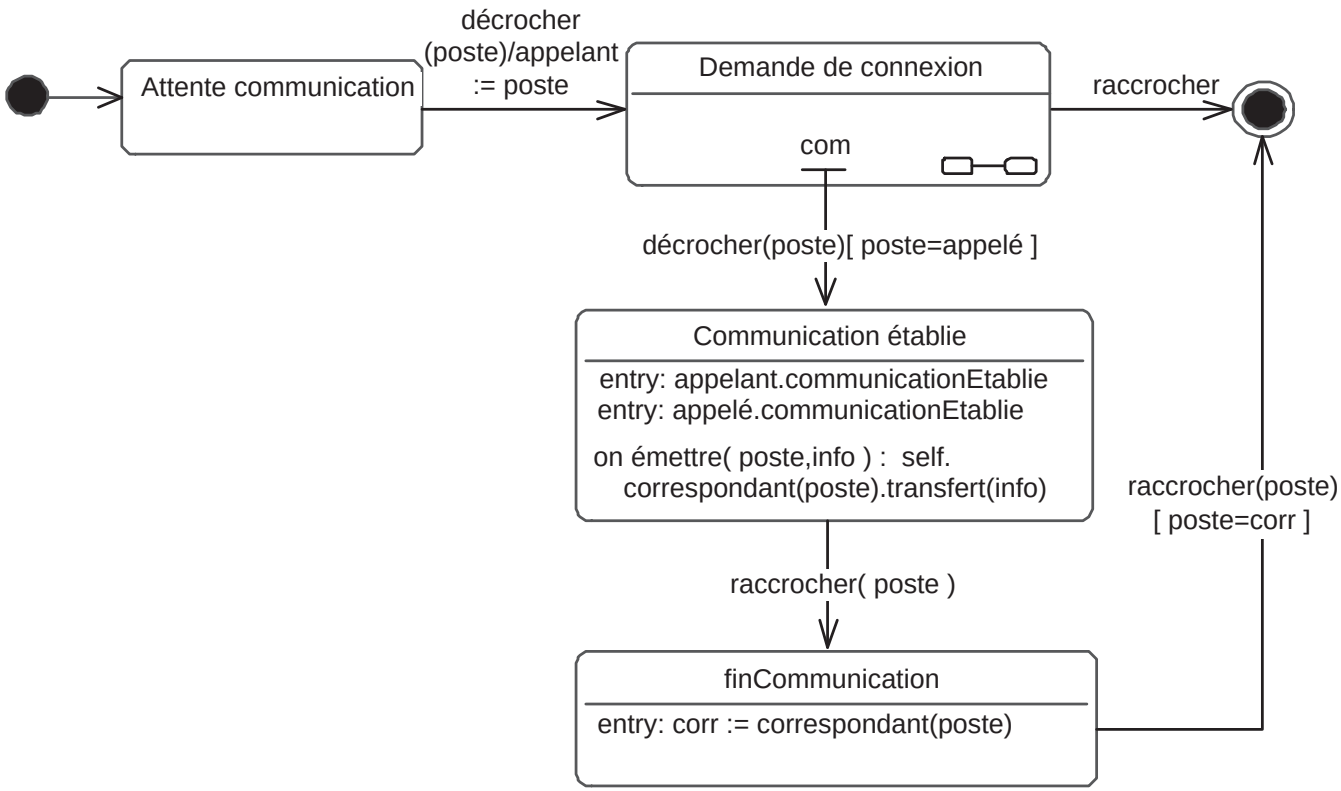


Diagramme états-transitions d'un objet `Commutateur`

MOF : automate du commutateur

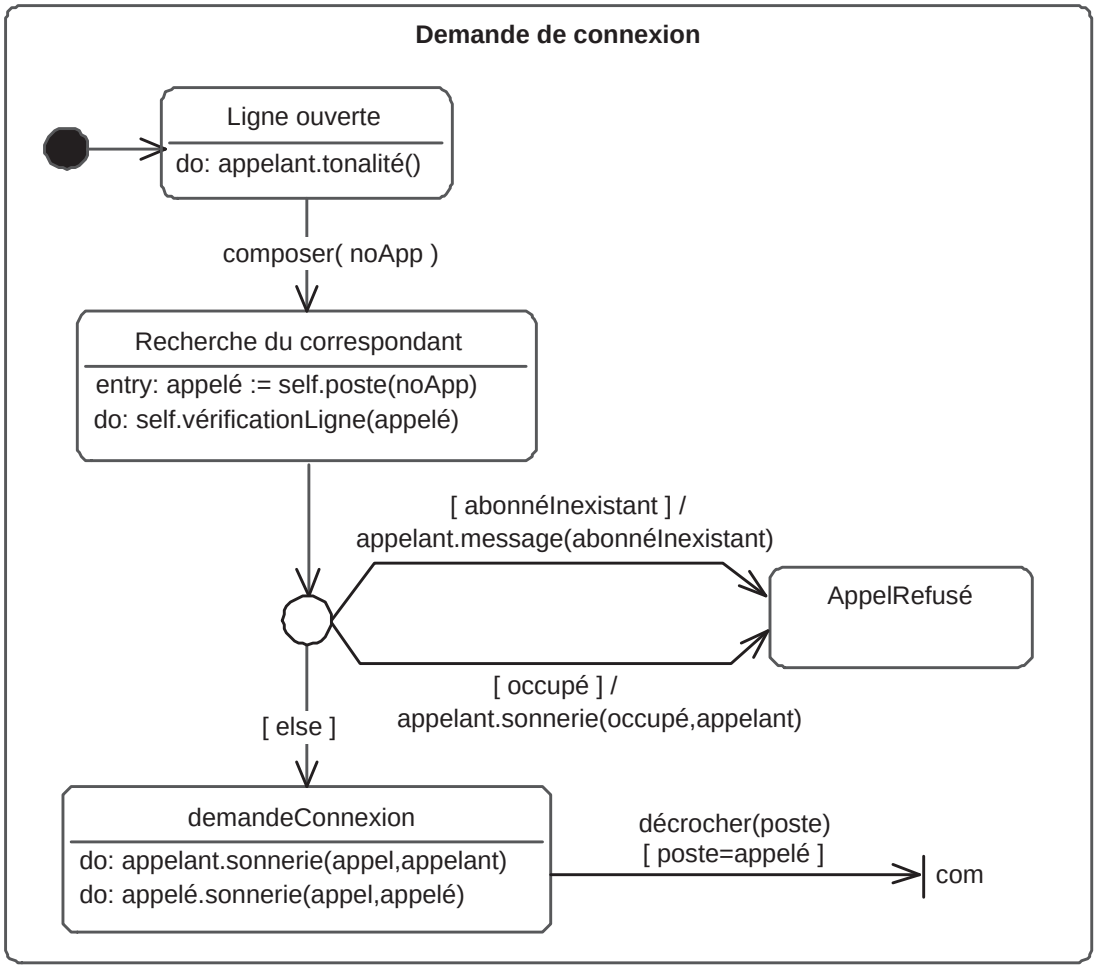
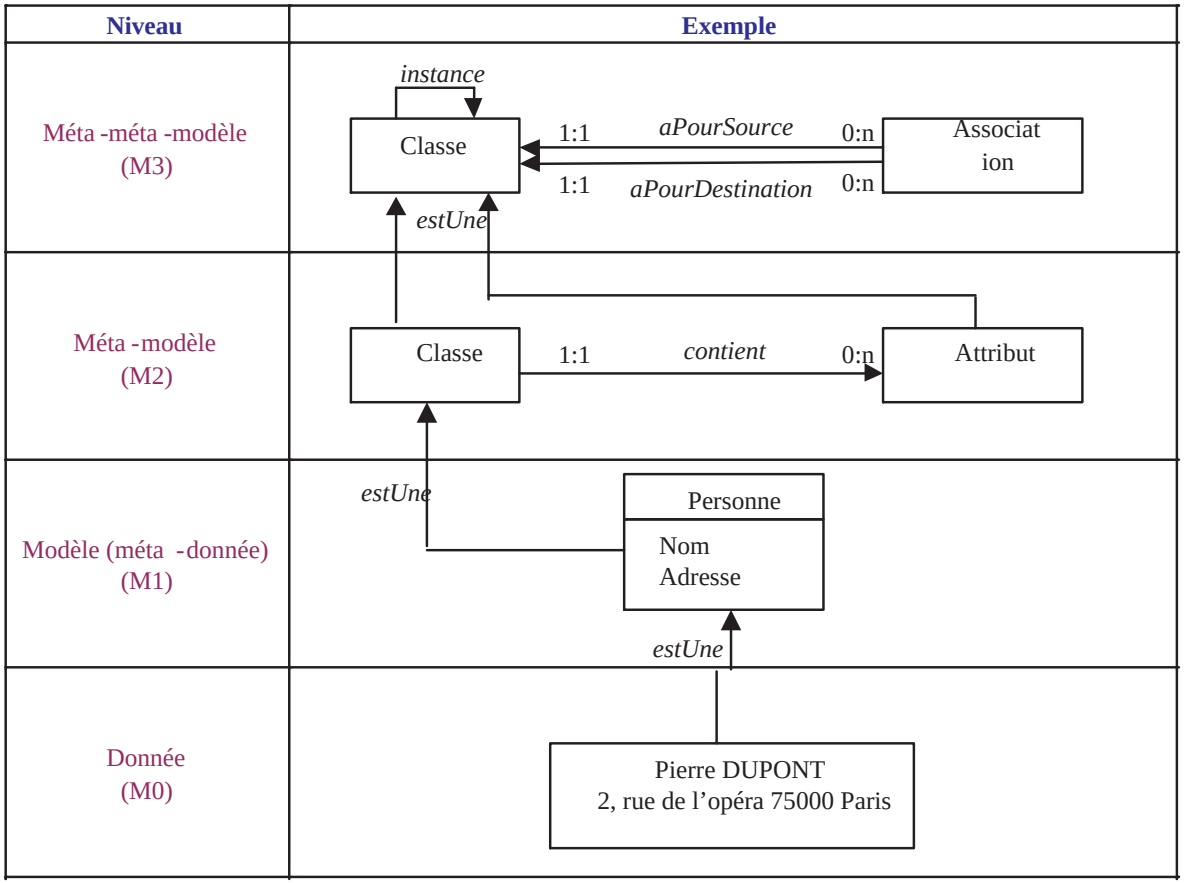


Diagramme états transitions d'un objet Commutateur, état Demande de

MOF : illustration 4 couches



Plan

OMG : normes de référence

1. MOF
2. XMI
3. QVT

XMI : évolution

Il y a longtemps eu séparation entre

1. XMI-UML
2. XMI-MOF

XMI contient maintenant quatre standards :

- XML - eXtensible Markup Language, a W3C standard.
- UML - Unified Modeling Language, an OMG modeling standard.
- MOF - Meta Object Facility, an OMG language for specifying metamodels.
- MOF Mapping to XMI

http://www.omg.org/spec/XMI/2.1/

XMI-MOF : état actuel

Specification Name:	**XML Metadata Interchange (XMI®)**
Description:	XMI is a model driven XML Integration framework for defining, interchanging, manipulating and integrating XML data and objects. XMI -based standards are in use for integrating tools, reposito ries, applications and data warehouses. XMI provides rules by which a schema can be generated for any valid XMI -transmissible MOF -based metamodel. XMI provides a mapping from MOF to XML. As MOF and XML technology evolved, the XMI mapping is being updated to comply with the latest versions of these specifications. Updates to the XMI mapping have tracked these version changes in a manner consistent with the existing XMI Production of XML Schema specification (XMI Version 2).
Keywords:	abstract class, archit ecture, CDATA, class diagram, composition, DTD, metadata, metamodel, modeling, production rules, schema, XML
Latest / past specifications:	Current versions: 2.1.1 Past versions
Revision Information:	Status: revision underway Working Document: Version 2.1.1 Contact: XMI for MOF 2 (XMI 2.1) RTF
Related OMG Specifications:	UML, MOF The following describes the relationship between releases of XMI and releases of MOF: XMI 1.1 corresponds to MOF 1.3 XMI 1.2 corresponds to MOF 1.4 XMI 1.3 (added Schema support) corresponds to MOF 1.4 XMI 2.0 (adds Schema support and changes document format) corresponds to MOF 1.4 XMI 2.1 corresponds to MOF 2.0
Related Industry Standards:	ISO/IEC 19503:2005; W3C DOM, EIA CDIF, W3C SAX, Web -DAV, W3C XML
Most recent IPR and Implementation questionnaire responses:	BCQ IBM

http://www.omg.org/spec/XMI/2.1/

XMI : Aperçu

OMG XML Metadata Interchange

- Objectif : permettre l'interopérabilité des méta-données
  - entre outils de modélisation (modèles UML)
  - entre répertoires de méta-modélisation (modèles MOF)
- Normes : basé sur 3 standards (XML, UML, MOF)
- Autres normes d'échange : EDOC, SMIF...

XMI : Aperçu

OMG XML Metadata Interchange

- Objectif : permettre l'interopérabilité des méta-données
  - entre outils de modélisation (modèles UML)
  - entre répertoires de méta-modélisation (modèles MOF)
- Normes : basé sur 3 standards (XML, UML, MOF)
- Autres normes d'échange : EDOC, SMIF...

Cours d'introduction à XML

http://aristote1.aristote.asso.fr/Presentations/CEA-EDF-2003/Cours/VincentQuint/IndexVincentQuint.html

XMI : quelques balises de la DTD 1/3

XMI.header : identification du modèle, métamodèle et métamétamodèle

<!ELEMENT XMI.header (XMI.documentation?,
XMI.model∗,
XMI.metamodel∗,
XMI.metametamodel∗,
XMI.import∗) >

XMI.content : données à transmettre

<!ELEMENT XMI.content ANY >

XMI : quelques balises de la DTD 2/3

`XMI.model` : identification du modèle (il peut y en avoir plusieurs)

```
<!ELEMENT XMI.model ANY>
<!ATTLIST XMI.model
%XMI.link.att;
xmi.name CDATA #REQUIRED
xmi.version  CDATA #REQUIRED
>
```

Les attributs `xmi.name` et `xmi.version` correspondent au modèle décrit dans `XMI.content`.

idem pour XMI.metamodel et XMI.metametamodel

XMI : quelques balises de la DTD 3/3

- XMI.extensions : extensions du métamodèle
- XMI.extension : extension du métamodèle
- XMI.documentation : auteur, outils...
- XMI.import : document externe
- XMI.difference : modification de la base
- XMI.delete : modification de la base
- XMI.add : modification de la base
- XMI.replace : modification de la base
- XMI.reference : référence à d'autres éléments XML (ex: types de données)

XMI : DTD UML 1/3

ModelElement : élément de modélisation

Listing 3: DTD UML

```
<!ELEMENT UML:ModelElement.taggedValue (UML:TaggedValue)* >
<!ATTLIST UML:ModelElement
name CDATA #IMPLIED
 visibility  (public | protected | private) #IMPLIED
binding IDREFS #IMPLIED
template IDREFS #IMPLIED
templateParameter IDREFS #IMPLIED
implementation IDREFS #IMPLIED
view IDREFS #IMPLIED
 presentation  IDREFS #IMPLIED
namespace IDREFS #IMPLIED
 constraint  IDREFS #IMPLIED
requirement IDREFS #IMPLIED
 ...
```

XMI : DTD UML 1/3

ModelElement (suite)

Listing 4: DTD UML

```
 provision  IDREFS #IMPLIED
 stereotype  IDREFS #IMPLIED
elementReference IDREFS #IMPLIED
 collaboration  IDREFS #IMPLIED
behavior  IDREFS #IMPLIED
 partition  IDREFS #IMPLIED
XMI.element.att;
XMI.link.att;
>
```

XMI : DTD UML 2/3

Classifier

Listing 5: DTD UML

```
<!ELEMENT UML:Classifier (UML:ModelElement.name |
UML:ModelElement.visibility |
UML:GeneralizableElement.isRoot |
UML:GeneralizableElement.isLeaf |
UML:GeneralizableElement.isAbstract |
XMI.extension | UML:ModelElement.binding |
UML:ModelElement.template |
UML:ModelElement.templateParameter |
...)
>
```

XMI : DTD UML 2/3

Classifier (suite)

Listing 6: DTD UML

```
<!ATTLIST UML:Classifier
name CDATA #IMPLIED
  visibility  ( public | protected | private ) #IMPLIED
isRoot ( false | true ) #IMPLIED
 isLeaf ( false | true ) #IMPLIED
 isAbstract ( false | true ) #IMPLIED
binding IDREFS #IMPLIED
 ... >
<!ELEMENT UML:Classifier.feature
   (UML:Feature | UML:StructuralFeature |
   UML:BehavioralFeature | UML:Attribute |
   UML:Operation | UML:Method |
   UML:Reception)* >
```

XMI : DTD UML 3/3

StateMachine

Listing 7: DTD UML

```
<!ELEMENT UML:StateMachine.top (UML:State | UML:CompositeState |
UML:SimpleState | UML:SubmachineState |
UML:ActionState | UML:ObjectFlowState | UML:ActivityState)* >
<!ELEMENT UML:StateMachine.transitions (UML:Transition)* >
<!ELEMENT UML:Transition.guard (UML:Guard)* >
<!ELEMENT UML:Transition.effect (UML:ActionSequence)* >
<!ELEMENT UML:State.internalTransition (UML:Transition)* >
<!ELEMENT UML:State.entry (UML:ActionSequence)* >
<!ELEMENT UML:State.exit (UML:ActionSequence)* >
```

XMI : DTD UML 3/3

StateMachine (suite)

Listing 8: DTD UML

```
<!ELEMENT UML:CompositeState.substate (UML:StateVertex |
UML:PseudoState | UML:State |
UML:CompositeState | UML:SimpleState |
UML:SubmachineState | UML:ActionState |
UML:ObjectFlowState | UML:ActivityState)* >
```

XMI : DTD MOF 1/3

Les éléments

Listing 9: DTD MOF

```
<!ELEMENT MOF:Namespace.contents (MOF:ModelElement |
MOF:Feature | MOF:Namespace | MOF:Constraint |
MOF:Import | MOF:TypedElement |
MOF:Tag | MOF:StructuralFeature |
MOF:BehavioralFeature |
MOF:MofAttribute | MOF:Reference |
MOF:Operation | MOF:MofException |
MOF:GeneralizableElement |
MOF:Classifier | MOF:Package |
MOF:Class | MOF:Association |
MOF:DataType | MOF:Parameter |
MOF:Constant | MOF:AssociationEnd |
MOF:TypeAlias)*
>
```

XMI : DTD MOF 2/3

Classifier MOF

Listing 10: DTD MOF

```
<!ELEMENT MOF:Classifier (MOF:ModelElement.name |
MOF:ModelElement.annotation |
MOF:GeneralizableElement. visibility |
MOF:GeneralizableElement.isAbstract |
MOF:GeneralizableElement.isRoot |
MOF:GeneralizableElement.isLeaf |
XMI.extension | MOF:ModelElement.container |
MOF:ModelElement.constraints |
MOF:GeneralizableElement.supertypes |
MOF:Namespace.contents)*
>
```

XMI : DTD MOF 3/3

Classifier MOF (suite)

Listing 11: DTD MOF

```
<!ATTLIST MOF:Classifier
name CDATA #IMPLIED
annotation CDATA #IMPLIED
 visibility ( public_vis | protected_vis | private_vis ) #IMPLIED
isAbstract (true | false ) #IMPLIED
isRoot (yes | no | dont_care) #IMPLIED
isLeaf (yes | no | dont_care) #IMPLIED
container IDREFS #IMPLIED
constraints IDREFS #IMPLIED
supertypes IDREFS #IMPLIED
%XMI.element.att;
%XMI.link.att;
>
```

Exercice

Exercice 1 (XMI-UML)
Représenter en UML le document XMI donné en annexe.

Exercice 2 (XMI-MOF)
Représenter en MOF le document XMI donné en annexe.

XMI et MOF

Table de correspondance MOF - XMI

Version MOF	**Version XMI**
MOF 1.3	XMI 1.1
MOF 1.4 (current)	XMI 1.2
MOF 1.4 (current)	XMI 1.3 (adds Schema support)
MOF 1.4 (current)	XMI 2.0 (current; new format)
MOF 2.0 (in process)	XMI 2.1 (in process)

Source OMG, Last updated on 03/24/2010

XMI et UML

Table de correspondance UML - XMI

UML	**XMI 1.0 DTD**	**XMI 1.1 DTD**	**XMI 1.2 DTD**	**XMI 1.2 Schema**	**XMI 2.0 Schema**	**XMI 2.1 Schema**
1.3	×	×	×	×		
1.4	×	×	×	×	×	
1.5	×	×	×	×	×	
2.0						XMI 2.1 (in process)

Source MDA en Action p. 108

Voir aussi : The XMI Hackers' Homepage :
http://homepages.inf.ed.ac.uk/perdita/XMI/

XMI et DI

DI = Diagram Interchange

- Objectif : permettre l'interopérabilité des représentations graphiques
  - Format SVG - Scalable Vector Graphics (W3C)
  - SVG est un langage XML
- DI : Transformations XML en SVG
- Association UML - DI dans le métamodèle *SemanticModelBridge*

UML-DI ne semble pas évoluer depuis 2006.

OMG : normes de référence

1. Object Management Group
2. Normes
3. UML
4. MOF
5. XMI
6. QVT

QVT : état actuel

Specification Name:	**MOF™ Query / Views / Transformations**
Description:	This specification is one of a series related to developing the 2.0 revision of the OMG Meta Object Facility specification, referred to as MOF 2.0. This specification addresses a techno logy neutral part of MOF and pertains to: 1.) queries on models; 2.) views on metamodels; and 3.) transformations of models.
Keywords:	Area, Bottom Pattern, Core Domain, Core Transformation, Domain, Guard Pattern, Identifying Property, Incremental Update, Key, Mapping (Core), Mapping Operation, Model Type, Operational Transformation, Relation, Relational Transformation, Relation Domain,Template Pattern, Trace Class, Trace Instance
Latest / past specifications:	Current version: 1.1 Beta 2 specification — Past versions: 1.0
Revision Information:	Contact: QVT 1.1 RTF
Associa ted documents:	XMI files
Related OMG Specifications:	MOF, OCL
Related Industry Standards:	
Most recent IPR and Implementation questionnaire responses:	BCQ — France Tel ecom — Softeam — TCS

http://www.omg.org/spec/QVT/1.1/Beta2/

spécification en cours

QVT : Aperçu

OMG QVT Query View Transformation

- Objectif : unifier les langages de transformation
  - par programmation
  - par patron, règle, template
  - par modélisation
- Normes : basé sur 3 standards (XML, UML, MOF)
- Autres approches : ATL, Kermeta, UMT, Bosco...

QVT : Aperçu

OMG QVT Query View Transformation

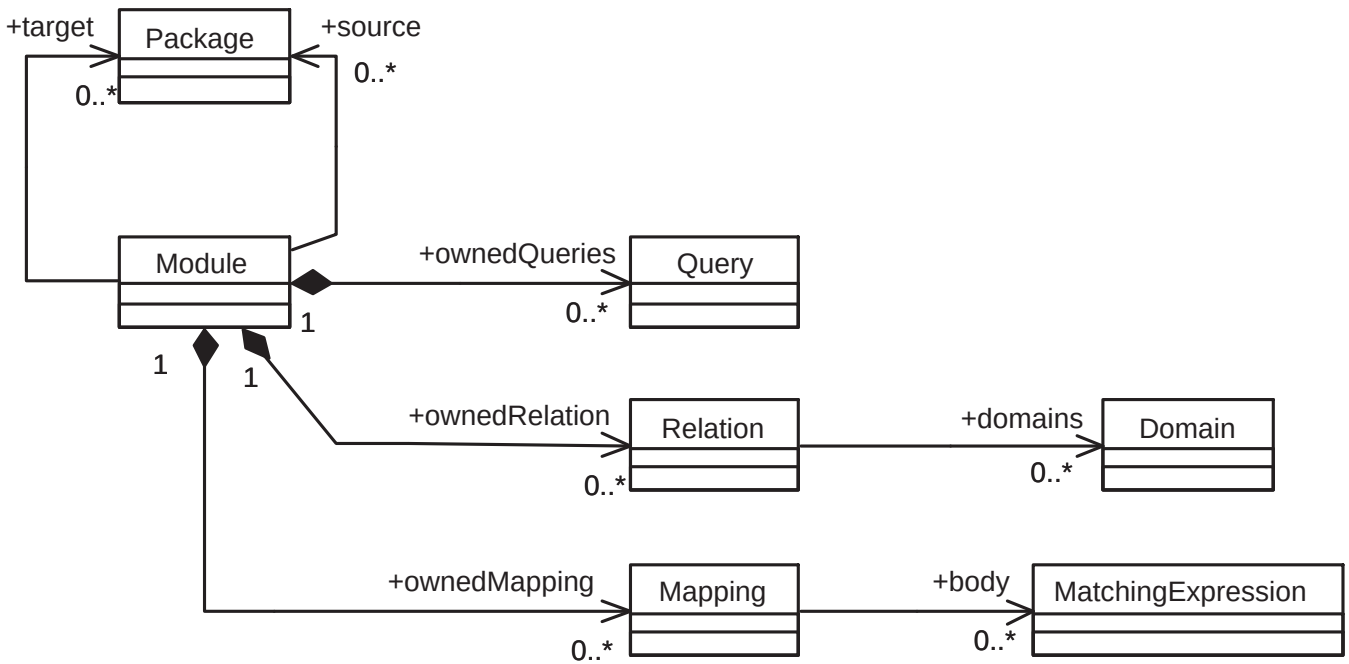
- Objectif : unifier les langages de transformation
  - par programmation
  - par patron, règle, template
  - par modélisation
- Normes : basé sur 3 standards (XML, UML, MOF)
- Autres approches : ATL, Kermeta, UMT, Bosco...

Référence de MOF2.0 QVT

http://www.omg.org/spec/QVT/1.1/Beta2/

QVT : métamodèle simplifié

[Bla05]

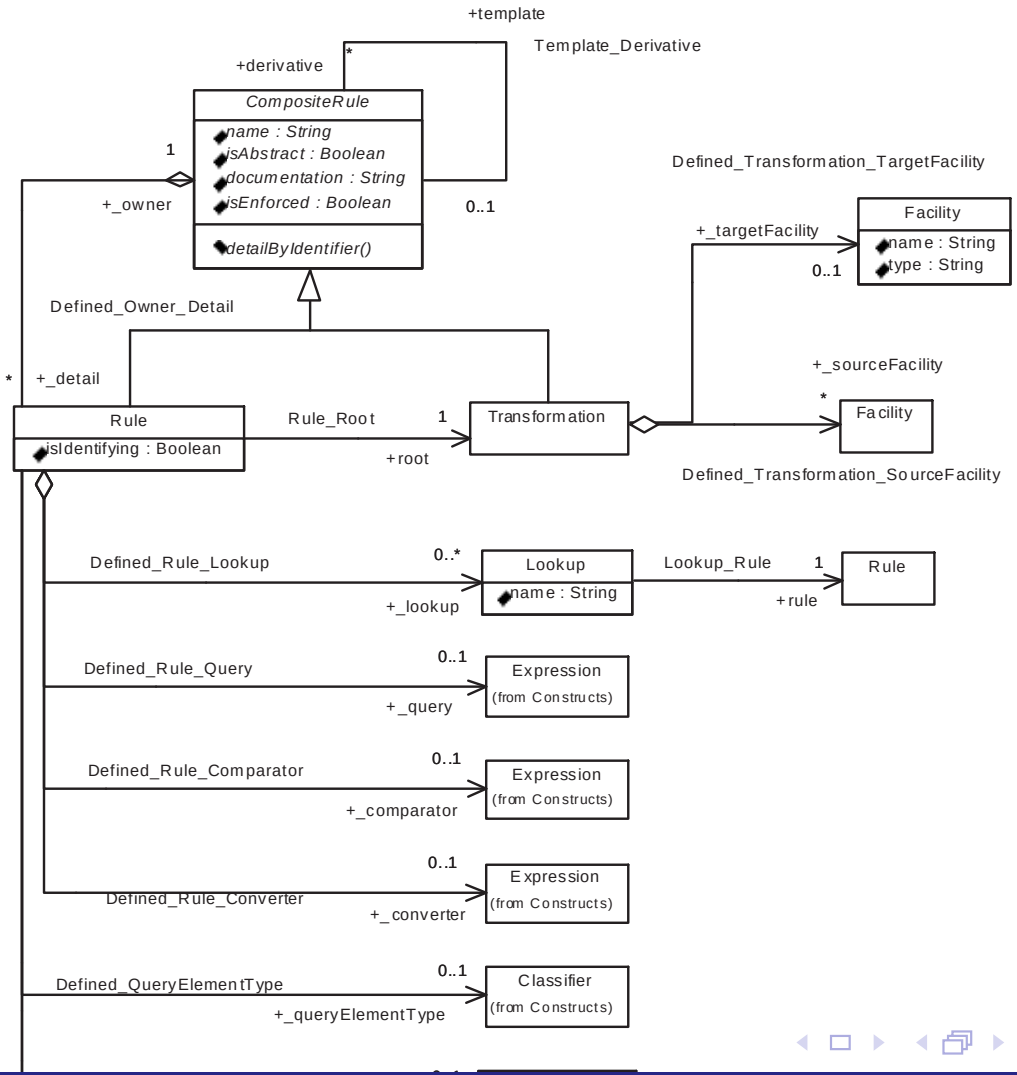


QVT : Principes

1 module = 1 transformation

- source, target : métamodèles (MOF2.0) sources et cibles de la transformation
- Query : requêtes de la transformation (OCL)
- Relation : règles de correspondance entre source et cible
  - Domain : sous-ensemble d'un métamodèle
- Mapping : règles de construction (correspondances structurelles entre métamodèles), déclaratif
  - MatchingExpression : actions de construction (impératif)

QVT : partiel (Source réponse Unisys)



Plan

MDA : remarque préliminaire

Outils de Génie Logiciel (+/- AGL) = automatiser

- édition de spécification
- analyse de spécification
- étude de propriétés, typage...
- manipulations : vérifications, preuves, tests, génération de code

La génération de "code" est un changement de représentation en vue d'utiliser d'autres outils GL.

MDA : la vision de l'OMG 1/3

Constat OMG (2000)

- fin du mythe de l'application autonome, sans maintenance, sans évolution

MDA : la vision de l'OMG 1/3

Constat OMG (2000)

- fin du mythe de l'application autonome, sans maintenance, sans évolution
- évolution rapide des technologies (XML, Web, langages, environnements)

MDA : la vision de l'OMG 1/3

Constat OMG (2000)

- fin du mythe de l'application autonome, sans maintenance, sans évolution
- évolution rapide des technologies (XML, Web, langages, environnements)
- remise en cause permanente des infrastructures techniques

MDA : la vision de l'OMG 1/3

Constat OMG (2000)

- fin du mythe de l'application autonome, sans maintenance, sans évolution
- évolution rapide des technologies (XML, Web, langages, environnements)
- remise en cause permanente des infrastructures techniques
- remise en cause permanente des besoins

MDA : la vision de l'OMG 2/3

Solutions

- abstraction : travailler sur des modèles et non des implantations

Model Driven

MDA : la vision de l'OMG 2/3

Solutions

- abstraction : travailler sur des modèles et non des implantations

  Model Driven

- automatisation : outils de production du code

MDA : la vision de l'OMG 2/3

Solutions

- abstraction : travailler sur des modèles et non des implantations

Model Driven

- automatisation : outils de production du code
- intégration : fusion des applications (existantes ou non)

MDA : la vision de l'OMG 2/3

Solutions

- abstraction : travailler sur des modèles et non des implantations

Model Driven

- automatisation : outils de production du code
- intégration : fusion des applications (existantes ou non)
- test et validation : générés depuis les modèles

MDA : la vision de l'OMG 2/3

Solutions

- abstraction : travailler sur des modèles et non des implantations

Model Driven

- automatisation : outils de production du code
- intégration : fusion des applications (existantes ou non)
- test et validation : générés depuis les modèles
- séparer les préoccupations (domaines, bridges)

MDA : la vision de l'OMG 2/3

Solutions

- abstraction : travailler sur des modèles et non des implantations

Model Driven

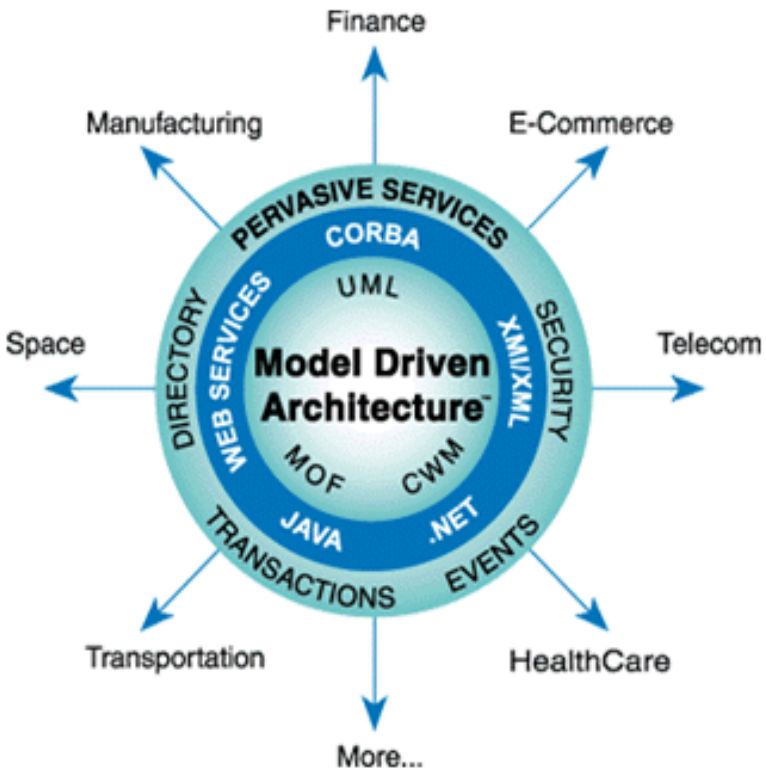
- automatisation : outils de production du code
- intégration : fusion des applications (existantes ou non)
- test et validation : générés depuis les modèles
- séparer les préoccupations (domaines, bridges)

nouvelle génération ?

une nouvelle étape dans les compilateurs

MDA : la vision de l'OMG 3/3

Fédère les activités de l'OMG



MDA : références

Sources bibliographiques = MDA

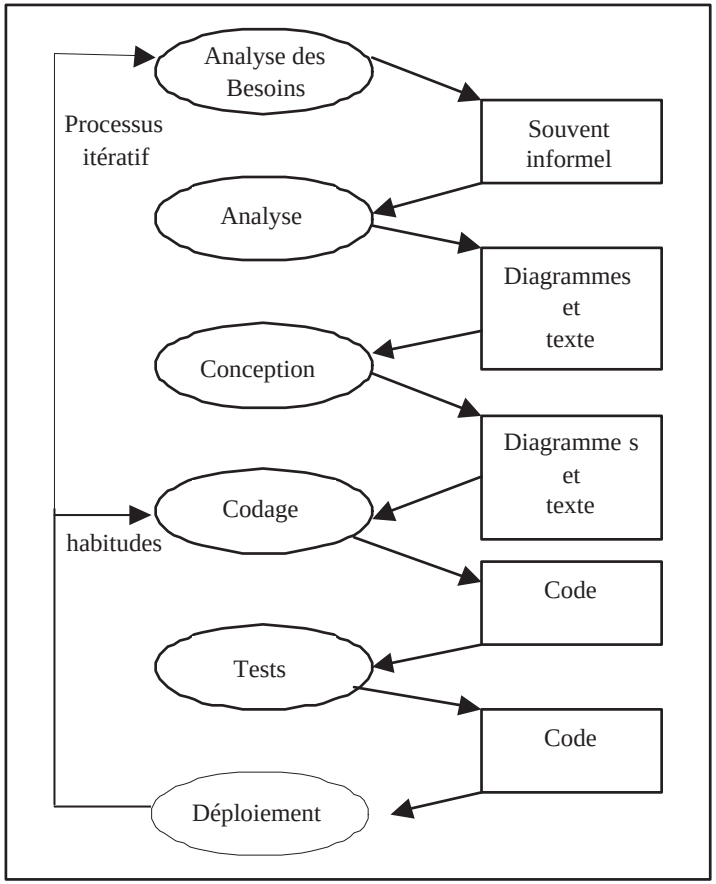
- la référence OMG [ME03]
- des approches MDA [KWB03, MSUW04, Lan05]
- ingénierie MDA [Bla05, Kad05]
- implantation [MB02, RFW+04]
  `http://www.kc.com/MDA/xuml.html`
- rapport bibliographique
  `http://www.sciences.univ-nantes.fr/info/perso/permanents/andre/COURS/DESS/ExposeDESS04/mda.pdf.zip`

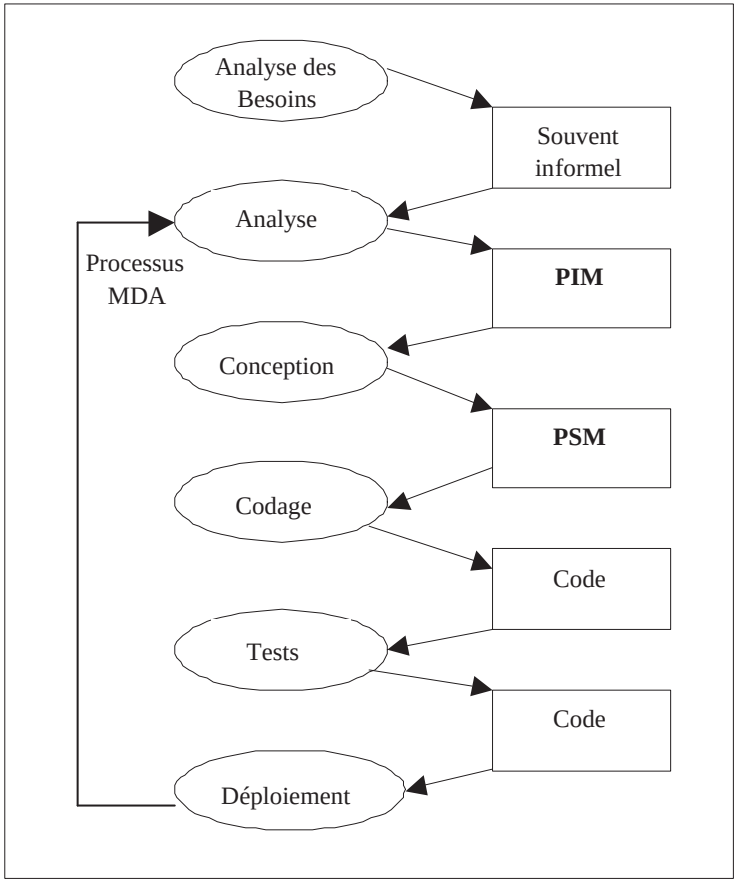
MDA : objectifs

- Productivité : réduire la distance modèle/code

MDA : objectifs

- Productivité : réduire la distance modèle/code





MDA : objectifs

- Productivité : réduire la distance modèle/code
- Portabilité : s'adapter aux technologies
  - chaque année apparaissent de nouvelles technologies (Java, Linux, XML, HTML, SOAP, UML, J2EE, .NET, JSP, ASP, Flash, Web Services,...)
  - problèmes d'investissements, de réutilisation

MDA = découpler modèles et infrastructure

MDA : objectifs

- Productivité : réduire la distance modèle/code
- Portabilité : s'adapter aux technologies
- Interopérabilité
  - faire communiquer anciennes et nouvelles applications
  - systèmes modulaires

MDA = utiliser des ponts (*bridges*) entre modèles

MDA : objectifs

- Productivité : réduire la distance modèle/code
- Portabilité : s'adapter aux technologies
- Interopérabilité
- Maintenance et documentation
  - éternel problème de la mise à jour des logiciels et des documentations
  - génération de documentations adaptées (haut niveau d'abstraction)

MDA = travail au niveau du PIM

MDA : objectifs

- Productivité : réduire la distance modèle/code
- Portabilité : s'adapter aux technologies
- Interopérabilité
- Maintenance et documentation

$\Rightarrow$ gagner en abstraction (PIM)

$\Rightarrow$ formaliser les technologies (PSM) et automatiser les transformations

MDA : objectifs

- Productivité : réduire la distance modèle/code
- Portabilité : s'adapter aux technologies
- Interopérabilité
- Maintenance et documentation

⇒ gagner en abstraction (PIM)

⇒ formaliser les technologies (PSM) et automatiser les transformations

Pas nouveau, mais l'originalité réside dans la tentative de formalisation (normalisation)

⇒ niveau d'abstraction de Merise (MCD, MLD, MPD)

⇒ générateurs de code

MDA : concepts de base 1/5

Définition (Système)

Le terme **système** *correspond à la désignation courante (organisation, personne, système information, programme...).*

Définition (Modèle)

Un **modèle** *d'un système est une spécification de ce système et de son environnemet selon des objectifs donnés (une interprétation).*

Définition (Architecture)

*L'***architecture** *d'un système est une spécification modulaire du système avec la description des modules et de leurs interactions (connecteurs).*

MDA : concepts de base 2/5

Définition (Plateforme)

Une **plateforme** *est un ensemble de sous-systèmes et de technologies fournissant un ensemble cohérent de fonctionalités décrites par des interfaces.*

- platesformes génériques : objet/batch/flots de données
- platesformes technologiques : CORBA, J2EE...
- platesformes propriétaires : Iona Orbix, Borland Visibroker, IBM Websphere, BEA Weblogic...

MDA : concepts de base 3/5

Définition (Vue)

Une **vue** *(un point de vue) constitue une abstraction selon certains critères de sélection (concepts architecturaux et règles de structuration).*

Le modèle MDA spécifie 3 points de vue :

- *Computation Independent Viewpoint* : environnement et besoins du système
- *Platform Independent Viewpoint* : opérations du système, invariant vis-à-vis des plateformes
- *Platform Specific Viewpoint* : ajoute des éléments spécifiques à une plateforme au PIM

MDA : concepts de base 4/5

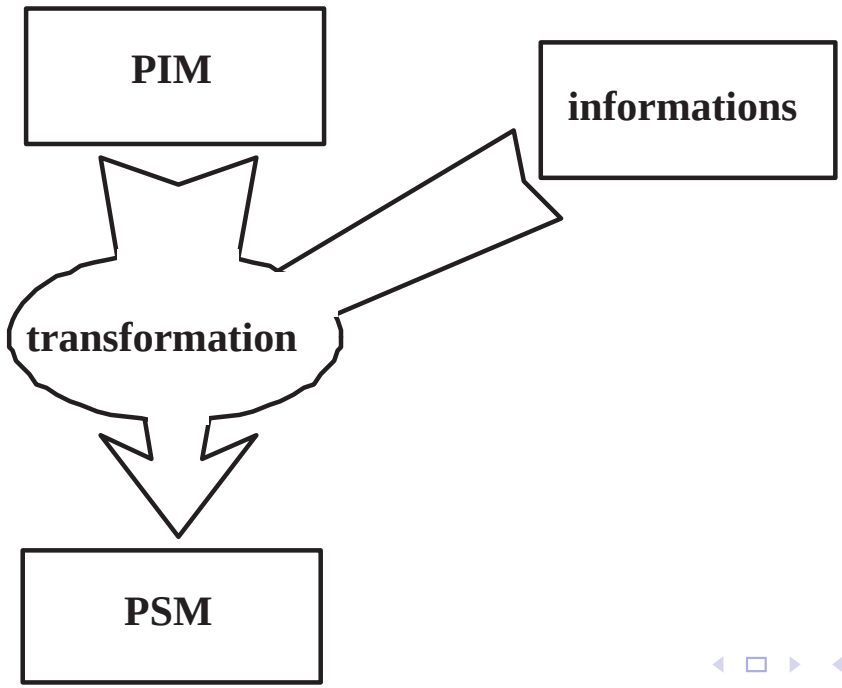
Aux 3 points de vue correspondent trois modèles :

- *Computation Independent Model* : le modèle selon la vue CIV, c'est-à-dire sans les détails de la structure du système. Il correspond aussi au modèle du domaine.
- *Platform Independent Viewpoint* : le modèle selon la vue PIV, c'est-à-dire le système indépendamment de toute réalisation. Par exemple, on utilisera une machine virtuelle comme un ensemble de service invocables. opérations du système, invariant vis-à-vis des plateformes
- *Platform Specific Viewpoint* : le modèle selon la vue PSV, c'est-à-dire dans les termes de la plateforme cible (ex: CCORBA, J2EE...)

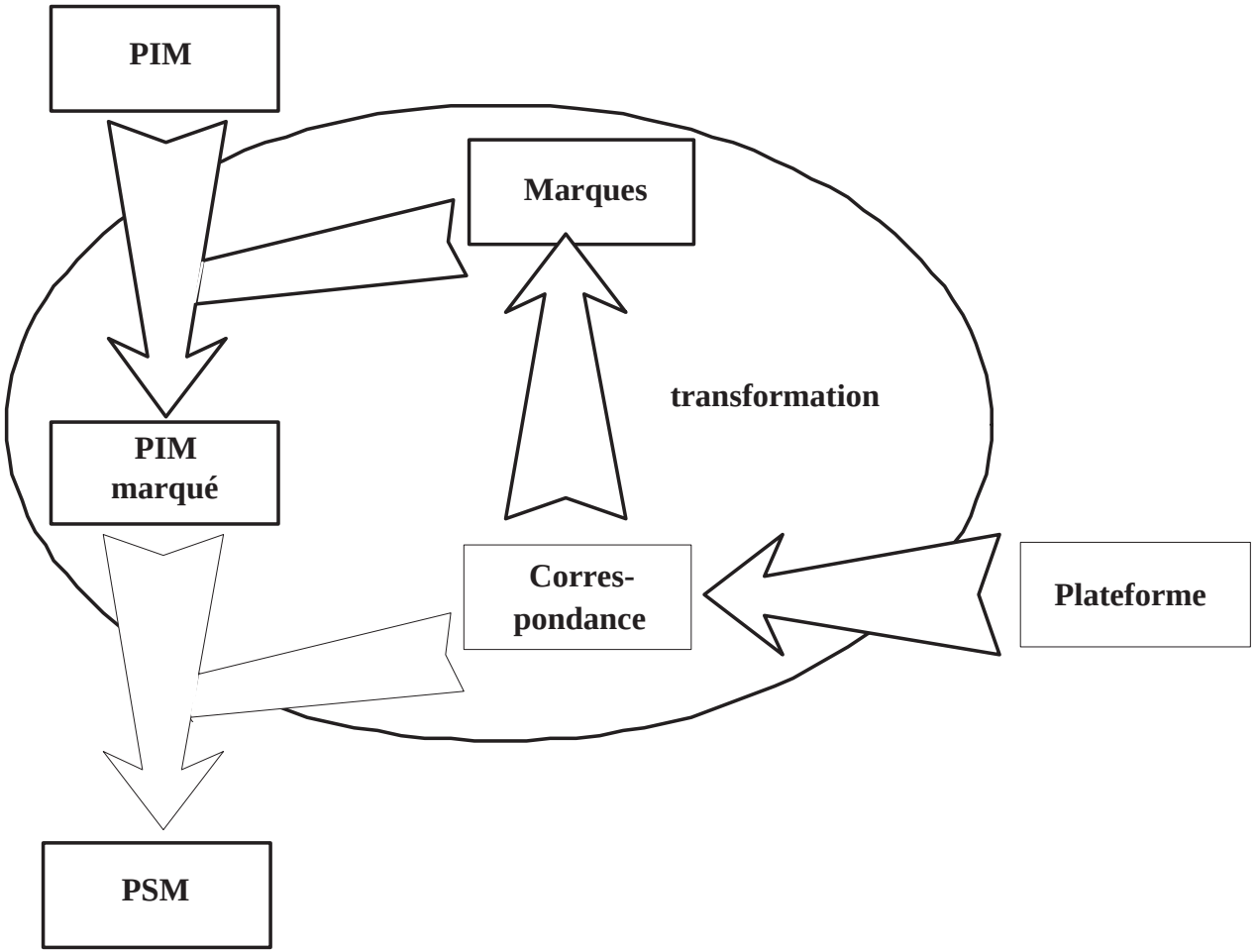
MDA : concepts de base 5/5

Définition (Vue)

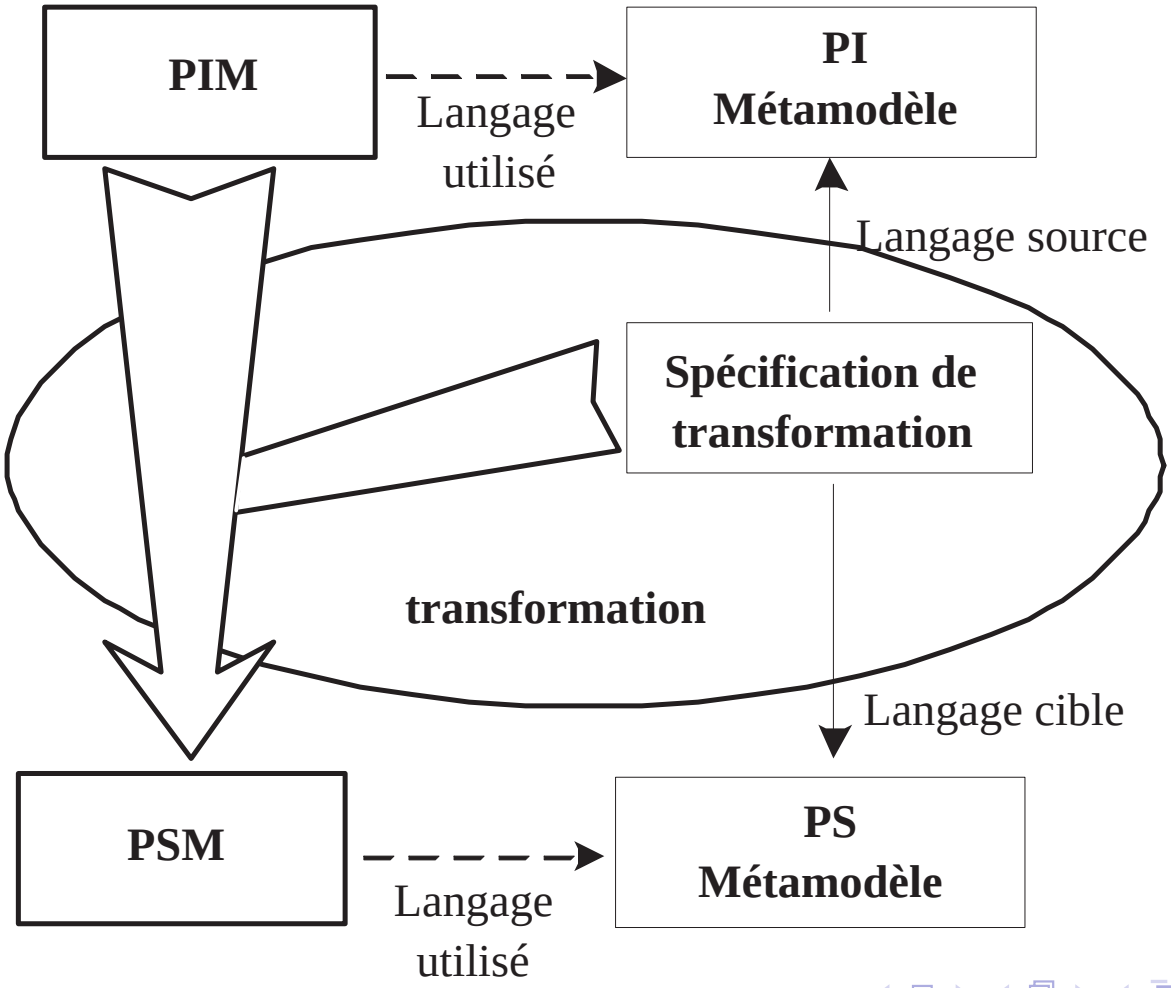
Une **transformation de modèle** *(un point de vue) convertit un PIM en un PSM selon des règles de transformation. C'est une sorte de patron réplicable.*



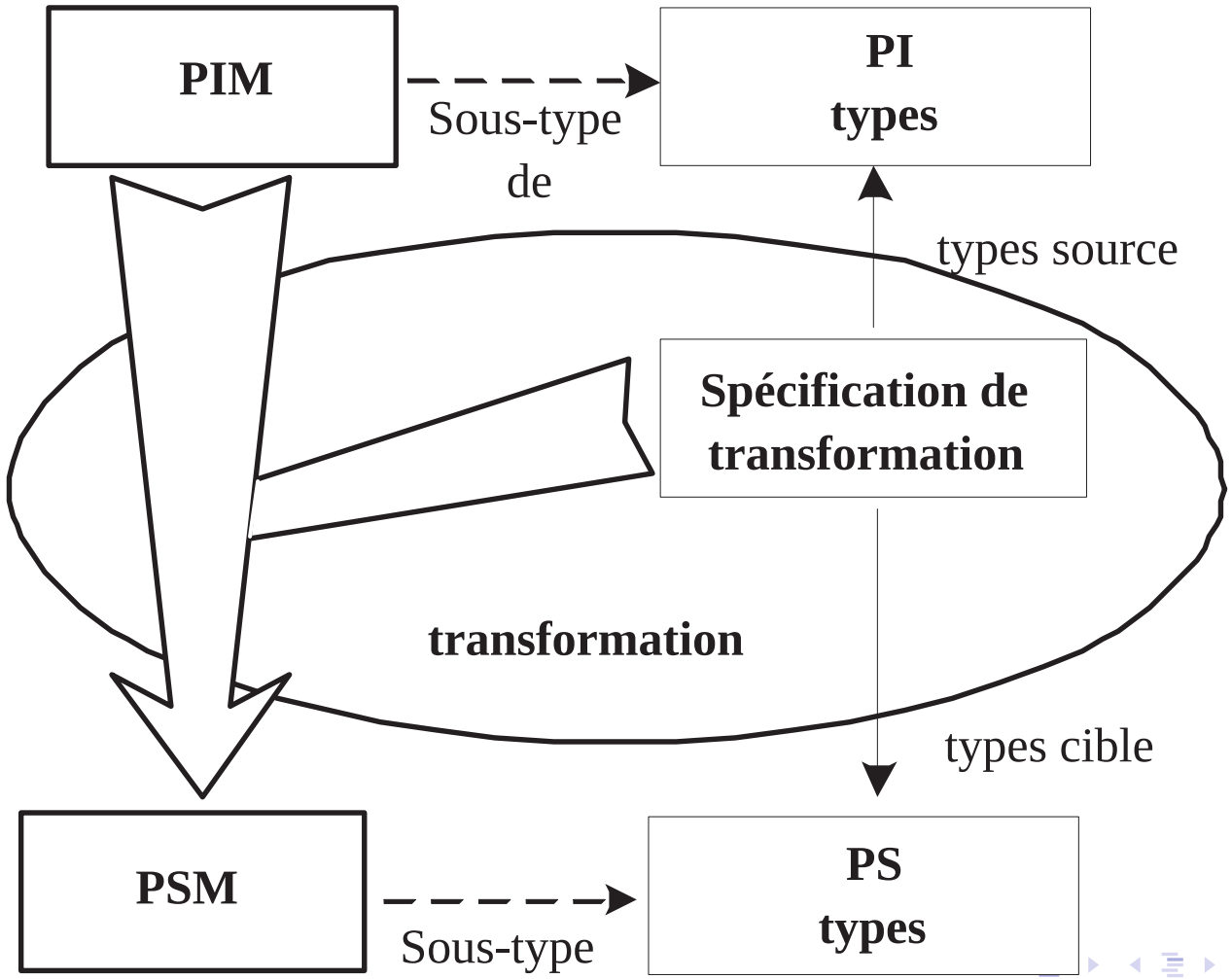
MDA : transformations par marquage



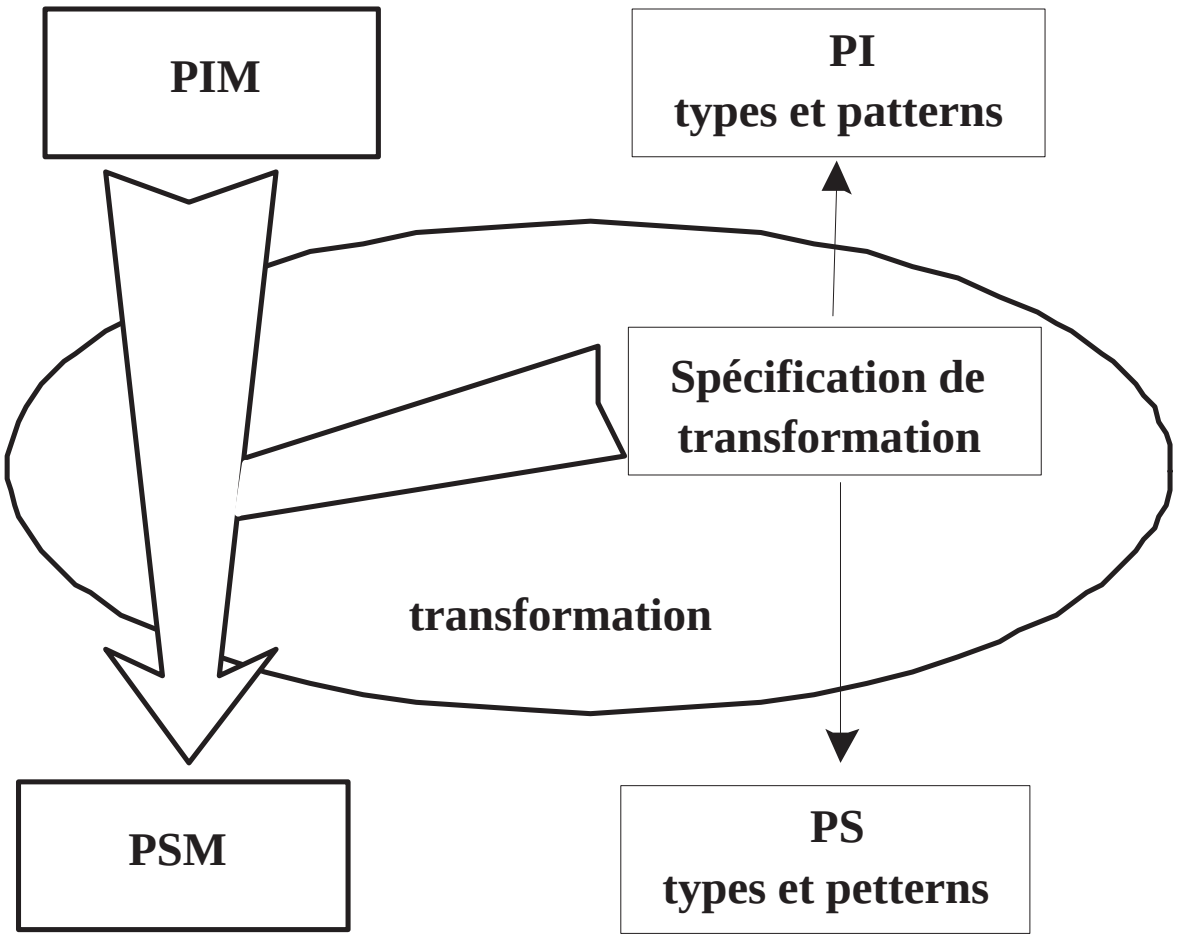
MDA : transformations de métamodèle



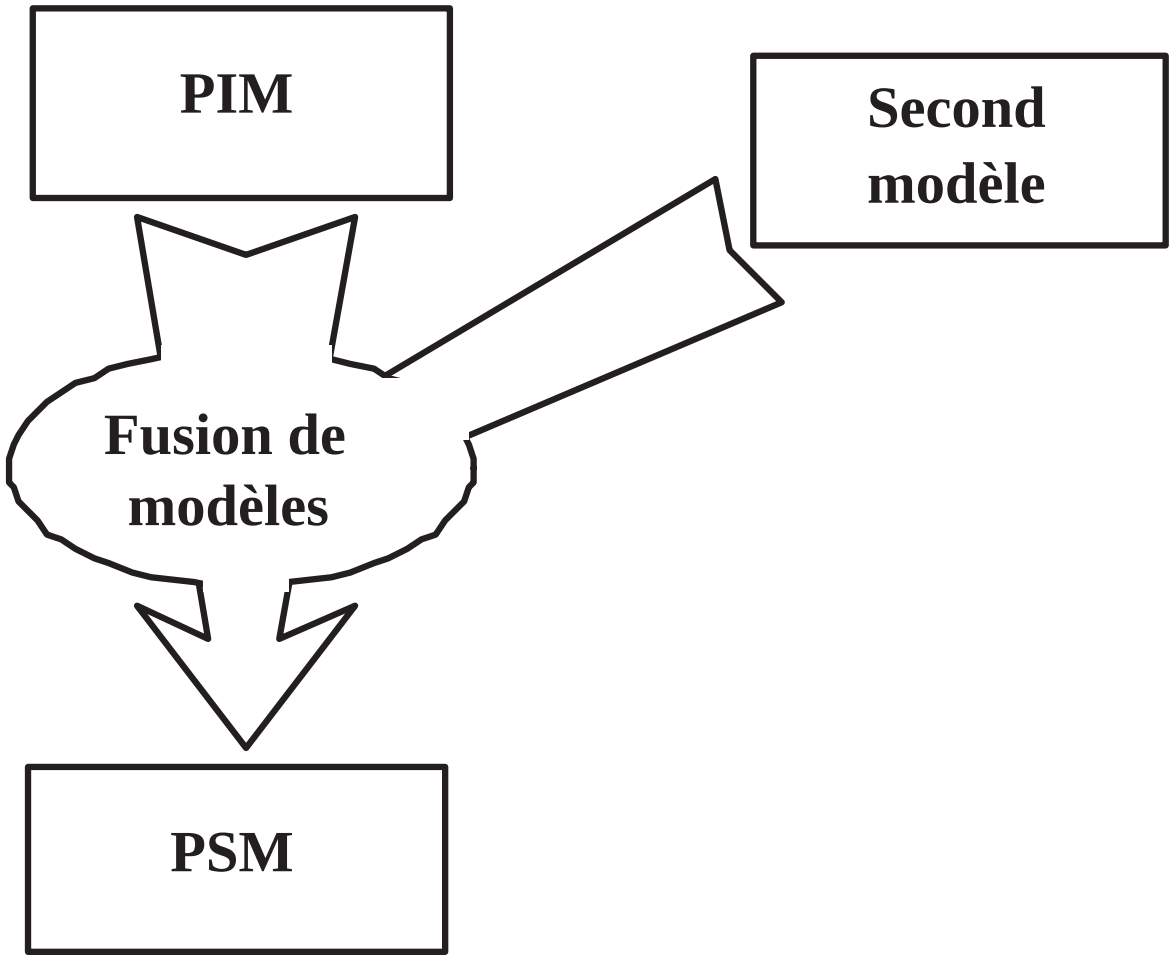
MDA : transformations de modèle



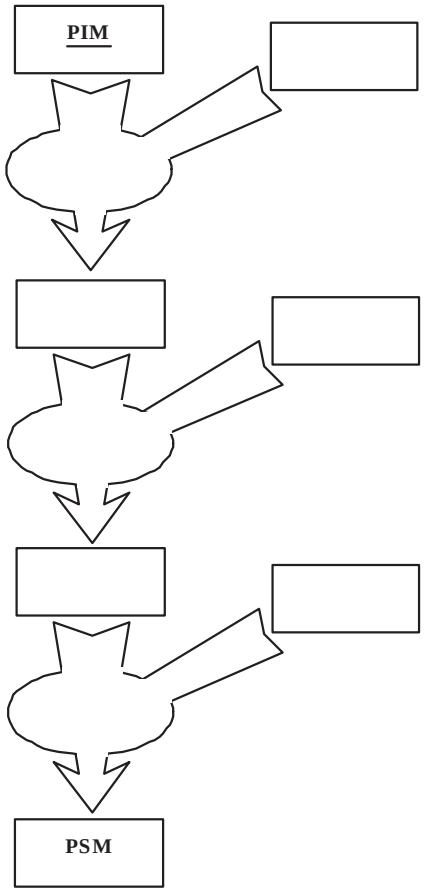
MDA : application de patron



MDA : fusion de modèles



MDA : combinaison de transformations



raffinement dans les méthodes formelles

MDA : vers un processus de transformations

Propriétés des transformations

- contrôler : paraméter, personnaliser, adapter, conditionner
- traçabilité
- cohérence incrémentale
- bidirectionnel

Les transformations deviennent des entités à part entière.

$\Rightarrow$ disposer d'un langage de transformation

$+$ connecté aux modèles (donc aux langages de modélisation)

$=$ métalangage pour les transformations

MDA : application

Les applications actuelles se focalisent sur 3 niveaux d'abstraction (*MDA Core*) :

- *PIM* : un modèle UML
- *PSM* : plusieurs modèles, chacun relevant d'une technologie particulière (transformations en parallèle + ponts)
  Le langage peut être spécifique ou un profil UML.
- *Code* : le code source de l'application

Le niveau *CIM* est ignoré car difficilement transformable directement en *PIM*.

MDA : application (suite)

Les applications actuelles se focalisent sur 3 niveaux d'abstraction (*MDA Core*) Remarques

- Il n'est pas fait mention de niveaux intermédiaires. Autrement dit, si on compose des transformations en série, on ne sait pas de quoi sont faits les modèles intermédiaires (l'entrée est un PIM, la sortie est un PSM).
- On a besoin d'abstractions sur les outils et environnements de développement (l'axe technique d'un cycle en Y).

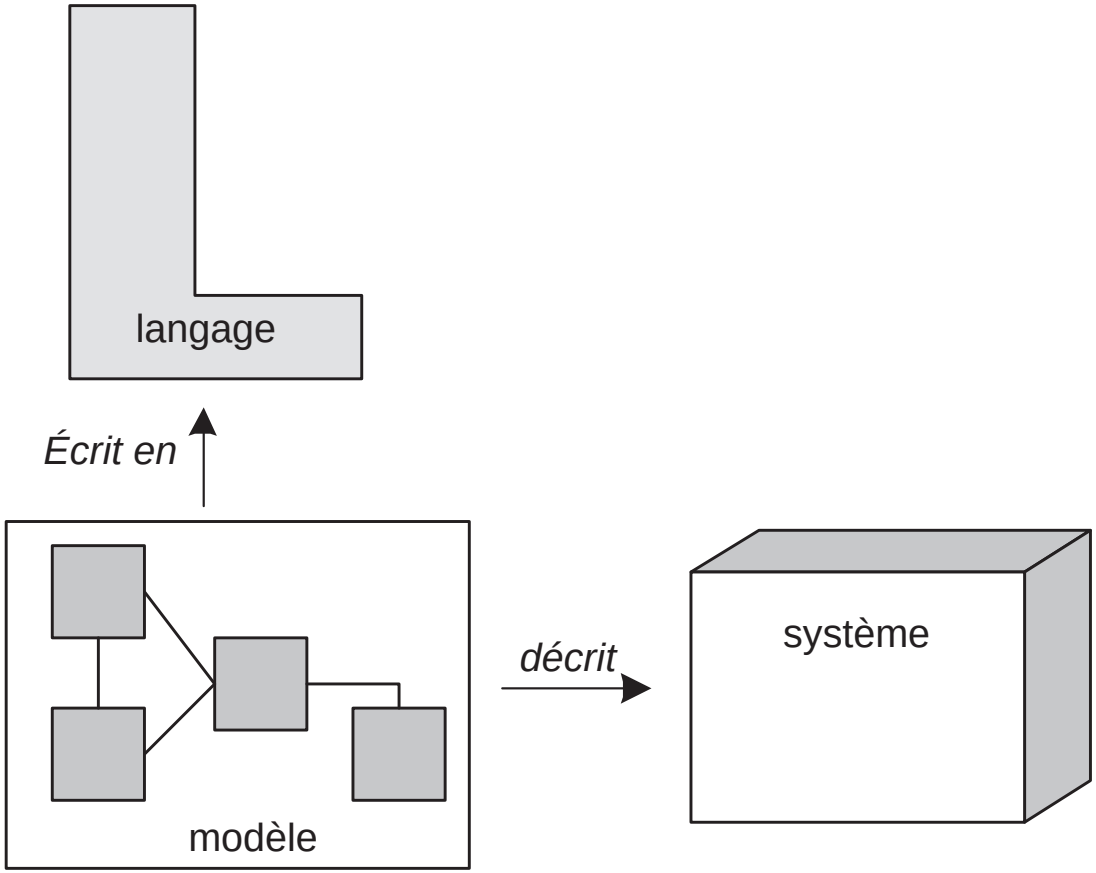
MDA : application (suite)

Les applications actuelles se focalisent sur 3 niveaux d'abstraction (*MDA Core*)

- Dans un processus de développement classique, à chaque étape (analyse, conception, conception détaillée, implantation), on ajoute ou transforme des éléments de modélisation et on prend des décisions.
  - Le développement MDA doit inclure ces décisions dans les transformations.
  - Pour transformer, il faut généralement une proximité sémantique : on ne transforme pas n'importe quoi en n'importe quoi.
  - Lorsque les transformations sont automatisées, toutes les informations essentielles se trouvent dans le PIM : on n'invente rien sur l'application, uniquement sur les représentations (exemple des actions).

MDA : un cadre de travail

[KWB03] Principe de modélisation



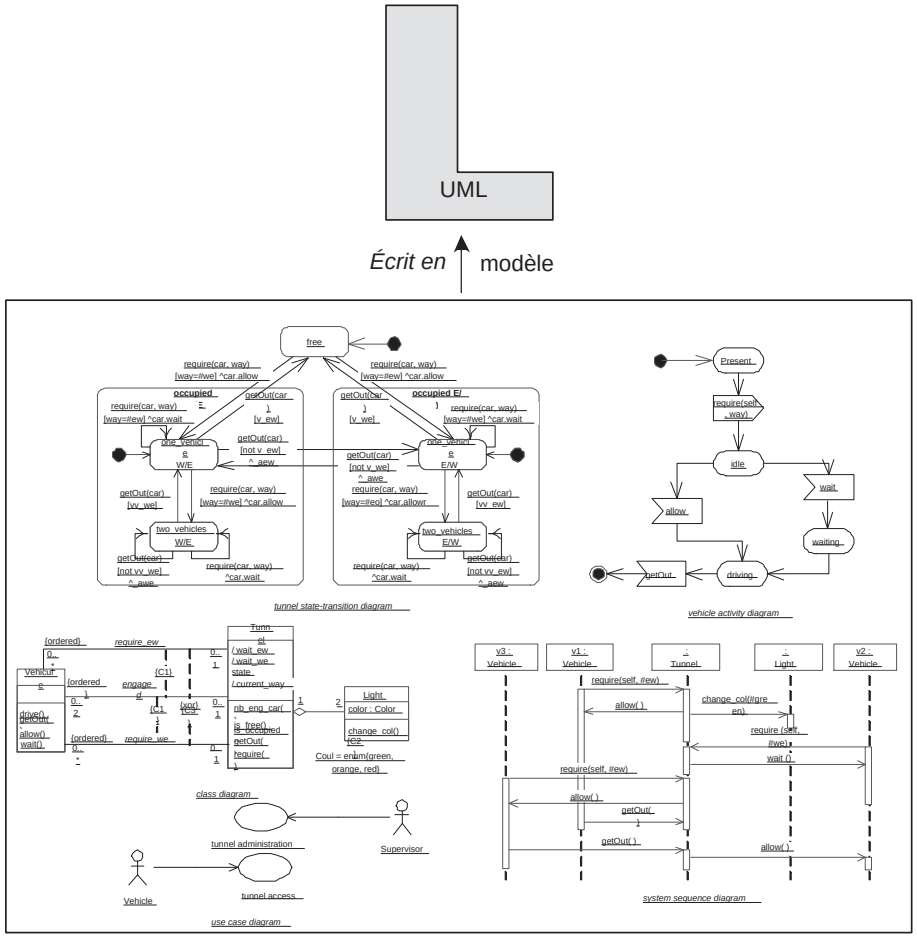
MDA : un cadre de travail(suite)

Différents types de modèles

- Métier / Application (*business and software models*) : CIM/PIM
- Structurel / dynamique (aspect, vue) : UML (différents diagrammes), Merise (E-A-P/Petri)
- PIM ou PSM : dénomination très relative
- Plateforme cible : profil UML, couche abstraite, couche propriétaire

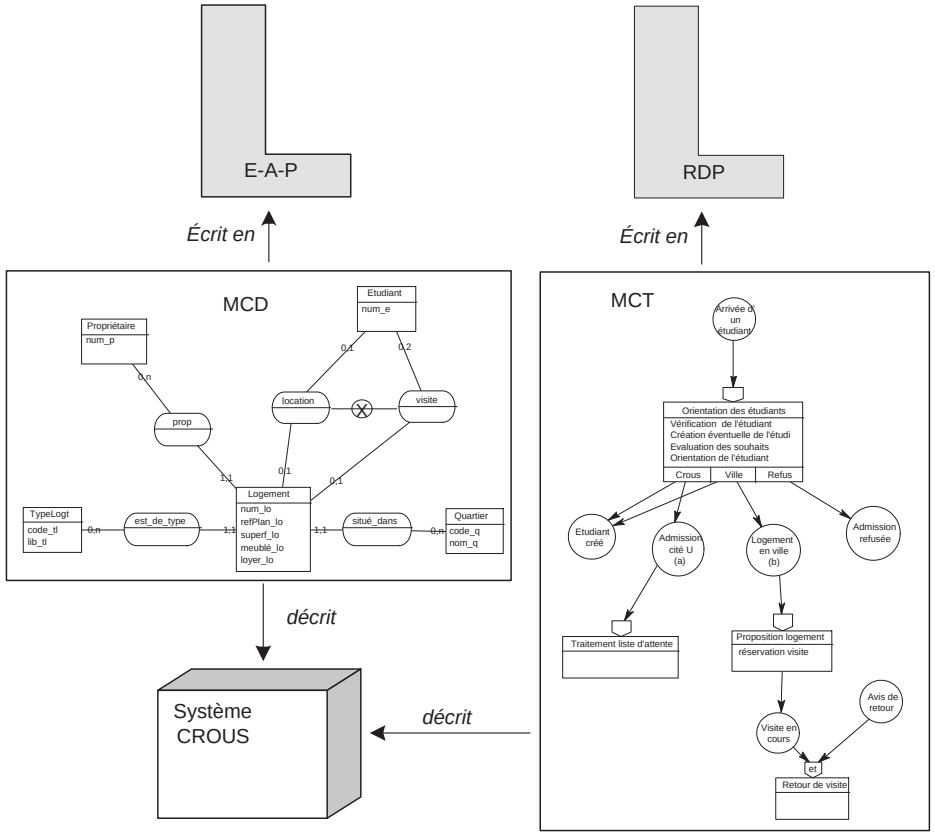
MDA : un cadre de travail(suite)

Langage multi-aspect



MDA : un cadre de travail(suite)

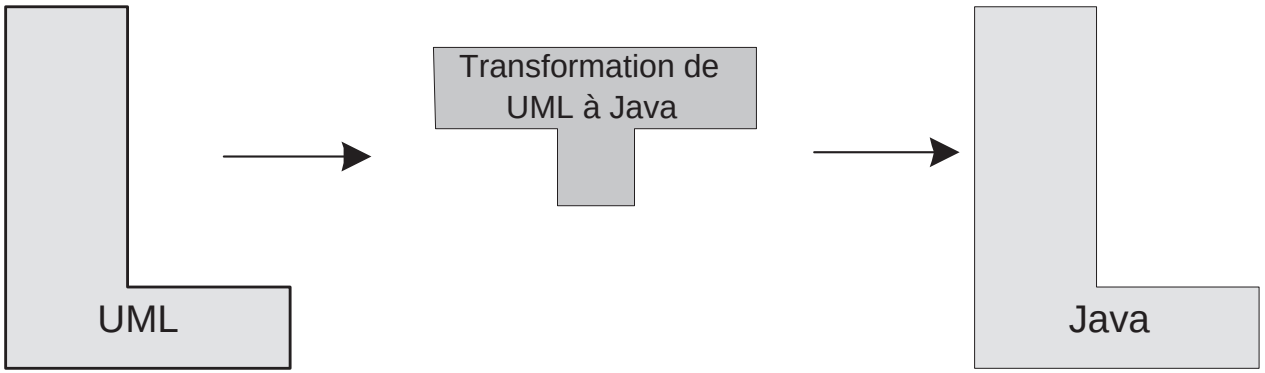
Un langage par aspect



MDA : un cadre de travail(suite)

Définition (Transformation)

Une **transformation** *est la génération automatique d'un modèle cible à partir d'un modèle source, selon la définition de la transformation.*



Définition (Définition de transformation)

Une **définition de transformation** *est un ensemble de règles de transformation qui décrivent comment un langage source est transformé en un langage cible.*

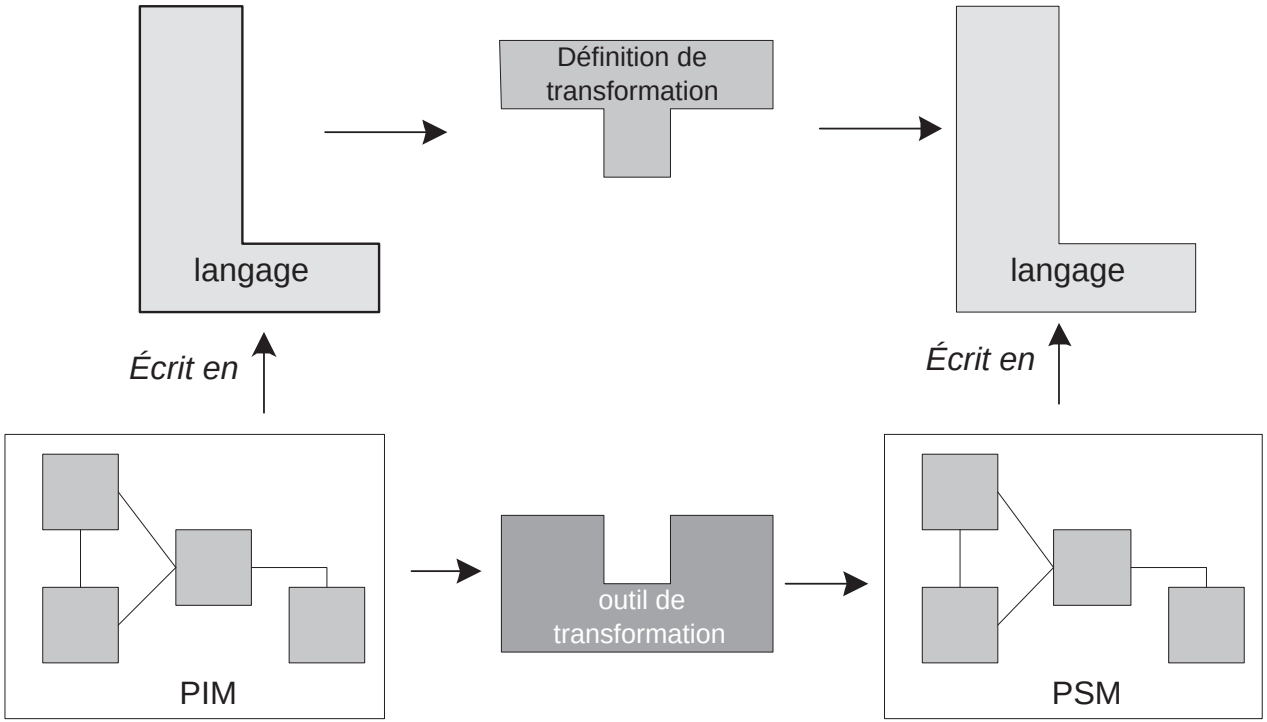
MDA : un cadre de travail(suite)

Définition (Règle de transformation)
Une **règle de transformation** *décrit la transformation de constructions du langage source en constructions du langage cible.*

- Le langage source et le langage cible peuvent être identiques : *refactoring*, normalisation
- La représentation concrète n'est isomorphe à la représentation abstraite. En termes de langage, une représentation XML (concrète) est une syntaxe pour différents langages abstraits.

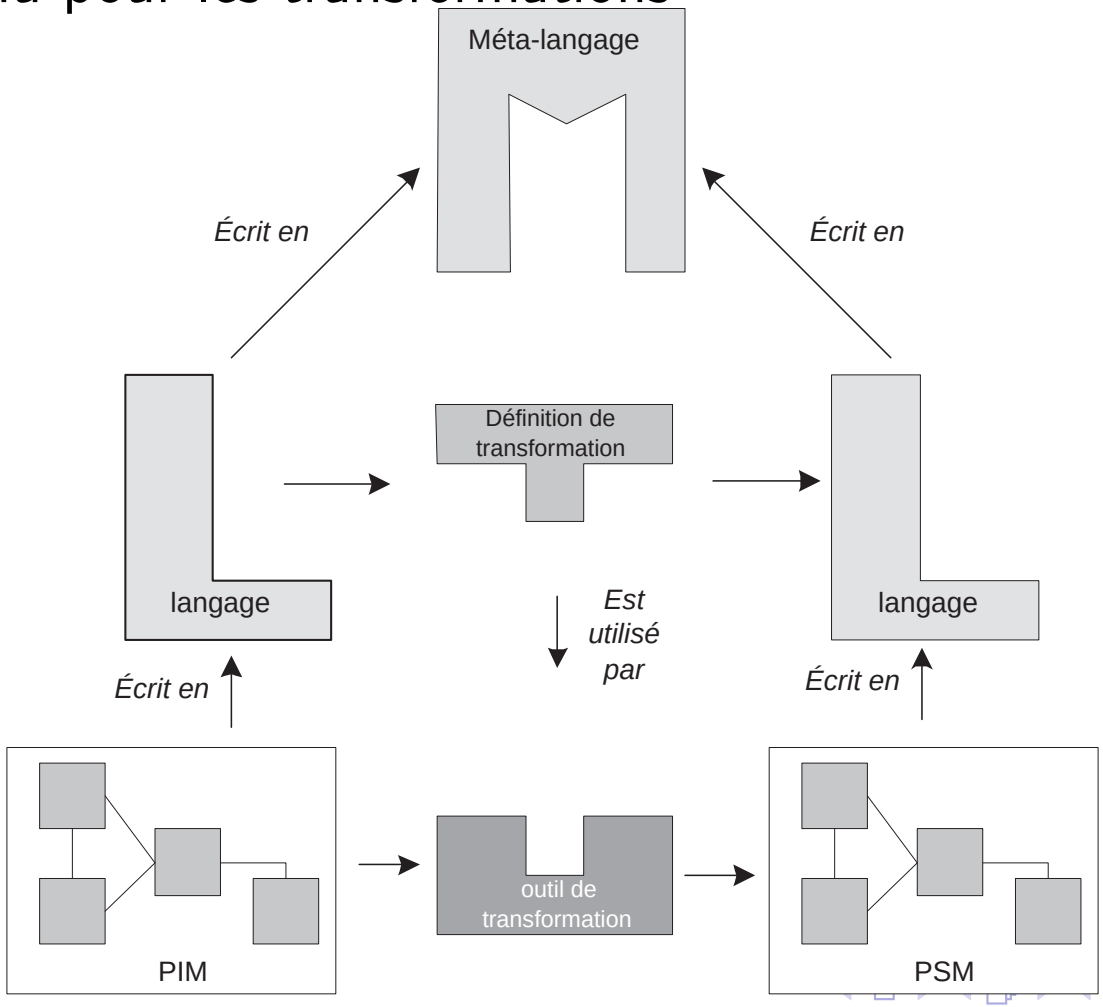
MDA : un cadre de travail(suite)

Cadre de base pour les transformations



MDA : un cadre de travail(suite)

Cadre étendu pour les transformations

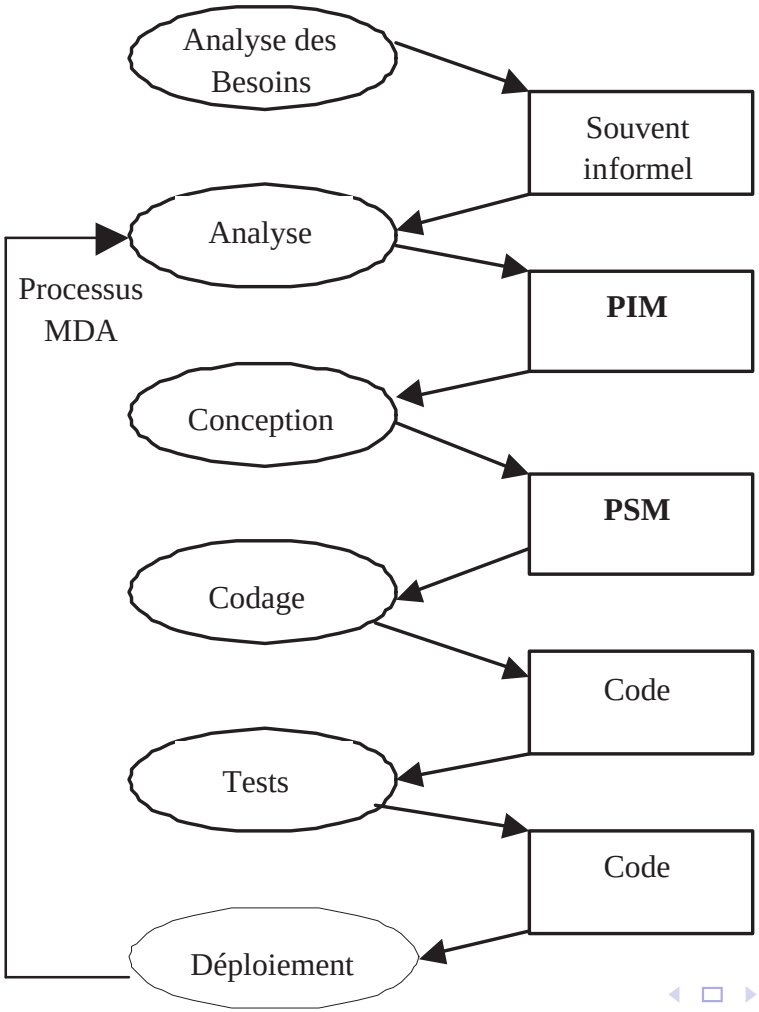


MDA : un cadre de travail(suite)

Langage de transformations
Règles

- nom
- paramètres
- cible
- condition
- actions de correspondance
- itérations

MDA : un cadre de travail (processus)



Plan

MDA : état des lieux 1/3

Base de l'OMG

- Langages modélisation : UML, OCL, AS, profils
- Langages de transformation : MOF, à venir QVT (Query, view, transformation)

PIM

- *Plain UML* : classique
- *Executable UML* : plusieurs versions basées sur la sémantique des actions (AS non standard)
- *UML-OCL* : différents compilateurs existent

MDA : état des lieux 2/3

Processus

- Processus agiles (e.g. eXtreme Programming)
- Processus unifié (e.g. RUP)

Outils de transformation (complexe car couvre tous les AGL)

- de PIM à PSM : rare
- de PSM à code : générateurs de code actuels, roundtrip ?
- de PIM à code : idem
- Transformateur paramétrable : scrips, QVT, ATL ?
- Définition de transformations : QVT, ATL ?

MDA : état des lieux 3/3

AGL + orientation MDA

- IBM Rational Software Modeler
- Softeam MDA Modeler

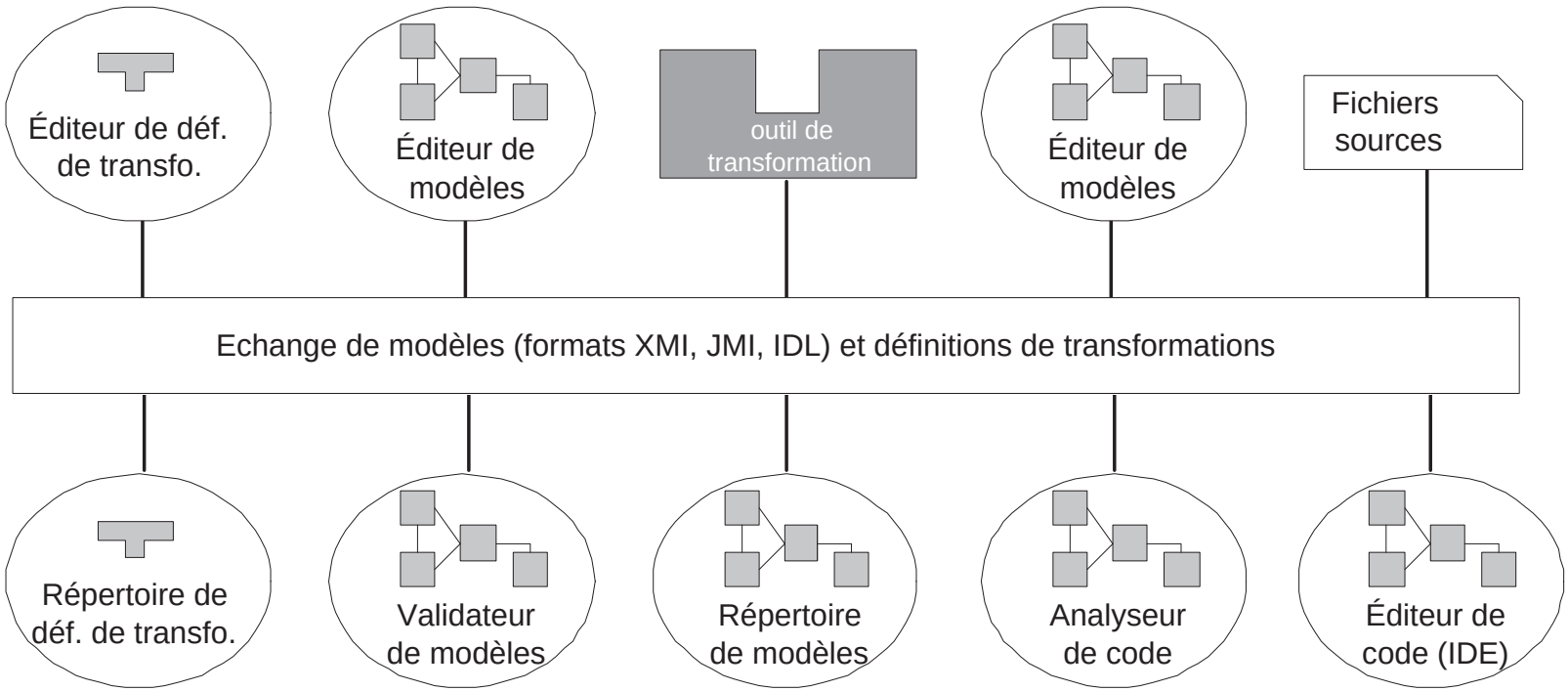
Outils plus spécifiques

- OMG : UMT, GMT, MotionModeling, AndroMDA...
- EMF (Eclipse) : ATL, Kermeta...

Committed Companies & Their Products
http://www.omg.org/mda/committed-products.htm
Chaque jour, de nouveaux outils...

MDA : environnement de développement

[KWB03]

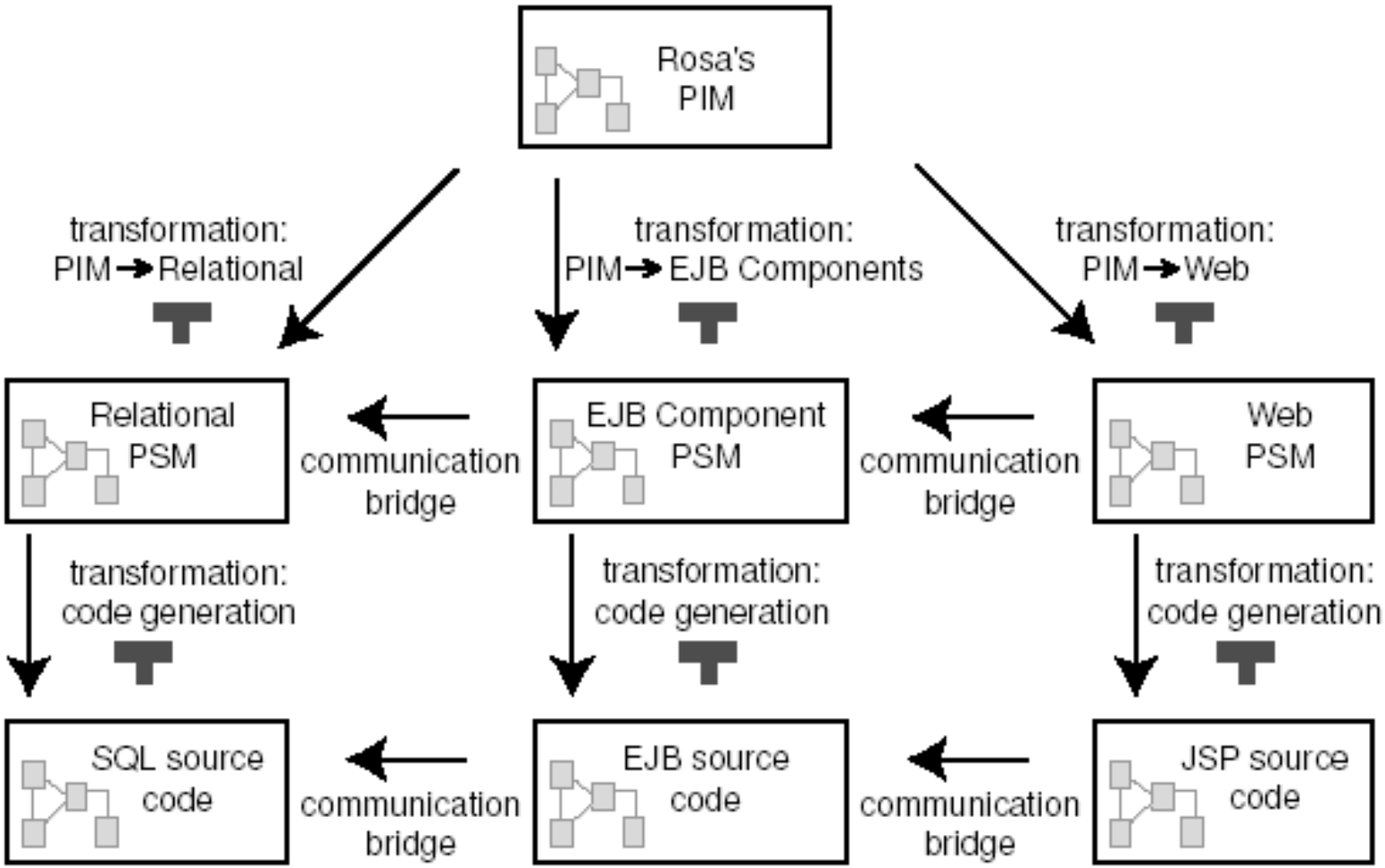


MDA : exemple 1

[KWB03] Kleppe/Warmer & Bast - Compuware's Optimal/J
http://www.compuware.fr/products/optimalj/

1. PIM = diagramme UML (classes métier) + OCL
2. Transformation PIM-PSM
   - de PIM à relationnel : connu
   - de PIM à EJB :
     - classe UML -¿ classe EJB Key
     - classe UML -¿ classe EJB Data
     - classe UML non composant -¿ composant EJB
     - association UML -¿ association avec un schéma EJB Data
     - ...
   - de PIM à Web : similaire EJB (informations)

MDA : exemple 1 (suite)



MDA : exemple 1 (suite)

3 Ponts : relient les classes des PSM

4 Transformation PSM-Code

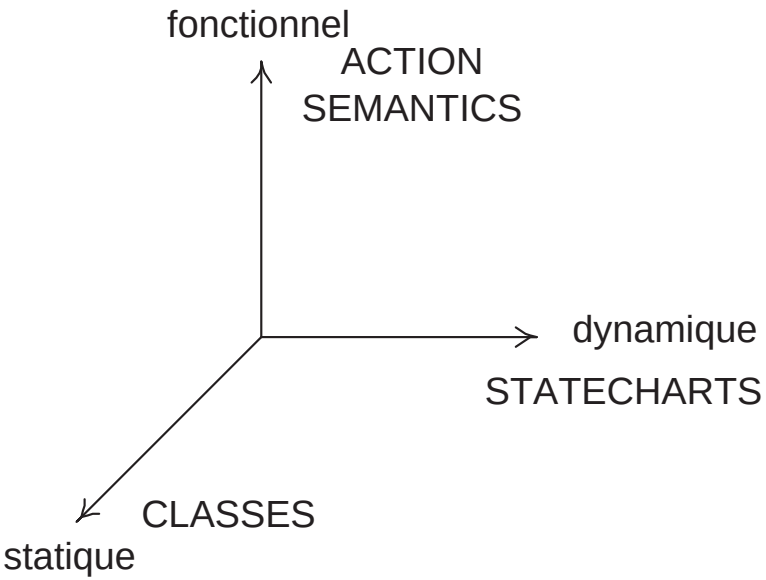
- de PSM/Rel à code : SQL
- de PSM/EJB à Java : spec. Sun
- de PSM/Web à JSP : JSP query

Commentaires

- Spécifications des PSM, règles de transformation
- On n'invente pas : dispose des éléments de base
- Au mieux on enrobe ou on plonge dans une architecture tout faire (*framework*) : ex : générateurs Access.
- Compilateur de modèles, *Pattern*

MDA : exemple 2

[MB02] XtUML, Balcer & Mellor
Project Technology, Nucleus BridgePoint - http://www.projtech.com



1. PIM = Executable UML
   - UML restreint
   - inspiré de Shlaer & Mellor (OOAD, TR, domaines)
   - restrictions
   - précision
   - BridgePoint Action Language

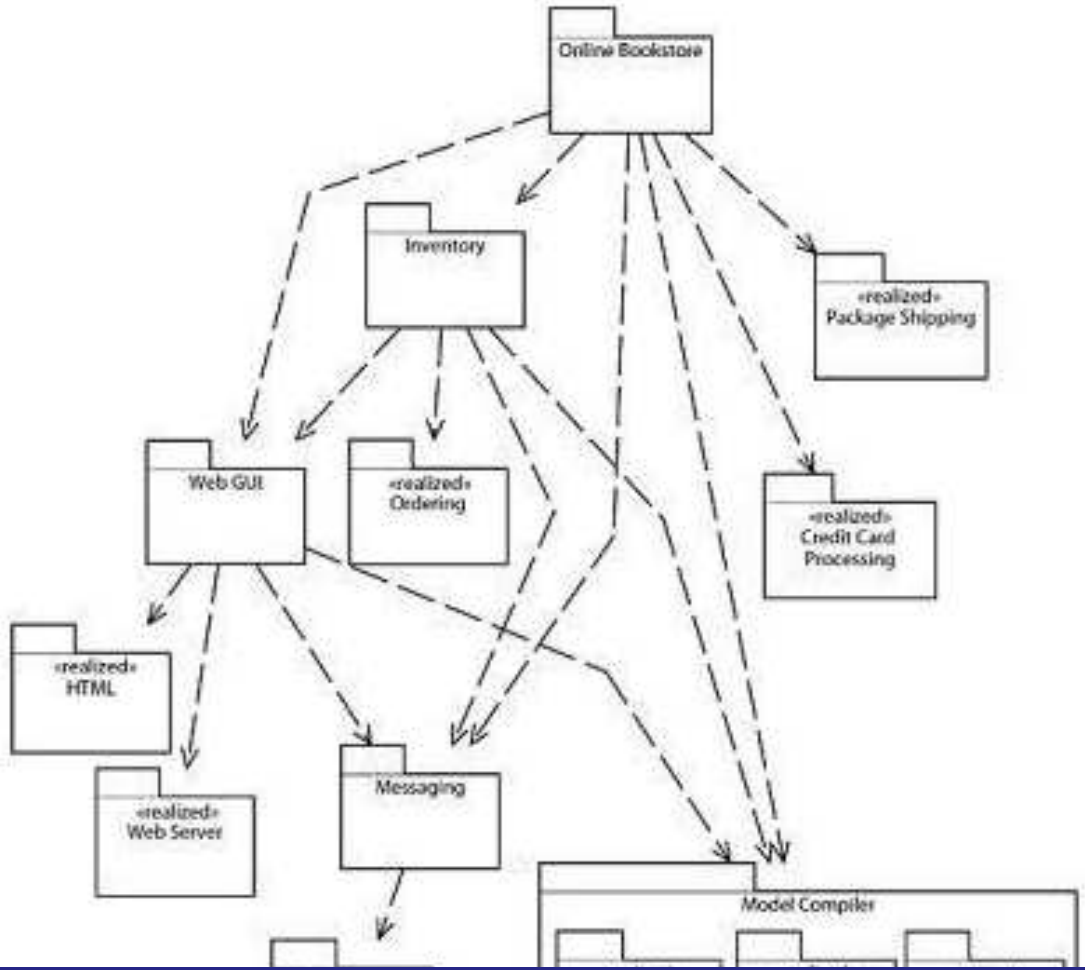
MDA : exemple 2 (suite)

Principes de modélisation

1. découper modulairement le modèle du système en domaines
   - modulaire : le domaine est cohérent et autonome
   - pas de séparation métier/technique : domaines "métier", GUI, BD, Programmation...
   - domaines substituables
   - domaines "existants"
   - étude des cas d'utilisation globale

MDA : exemple 2 (suite)

domaines



MDA : exemple 2 (suite)

Principes de modélisation (suite)

1. découper modulairement le modèle du système en domaines
2. modélisation individuelle des domaines (3 axes)
3. ponts = dépendance (hypothèse-besoins) en couche entre domaines
4. itérations intra-domaines et inter-domaines (modèles)
5. vérifications de modèles
   - statique
   - dynamique par exécution de scénarios
6. compilation de modèles $\Rightarrow$ modèles exécutables

MDA : exemple 2 (suite)

2 PSM : compiled model

3 Transformation PIM-PSM : Compilateur de modèles
⇒ plonger dans un *framework* technique

- C++ multi-tâche pour systèmes embarqués
- C++ multi-processus pour systèmes transactionnels
- C++ multi-processus pour systèmes critiques
- C sans OS pour systèmes intégrés
- C++ multi-OS pour systèmes événementiels
- Java Byte Code pour EJB et XML
- Machine virtuelle UML (?)
- ...

MDA : exemple 2 (suite)

domaine / ponts / aspects / points de jonction

- un domaine pose des hypothèses (sur un pont)

Domaine : IHM Web
Objectif : fournir un accès en ligne aux utilisateurs.
Hypothèses non résolue : les menus de sélection acceptent une quantité "15 exemplaires de ce livre".
Pont vers le domaine Serveur Web : Les communications sont sécurisées.
Compilateur : Les communications sont au format XML.
Les instances sont persistantes.

MDA : exemple 2 (suite)

- au niveau exécution ⇒ AOP
  - un domaine est un aspect
  - un point est un ensemble de points de jonction
- Ponts explicites : références intra-domaine
  - entités externes (acteurs)
  - messages (signaux) externes
  - opérations de pont : définie dans une entité externe (paquetage)
- Ponts implicites : points de jonction entre domaines
  - pas de dénomination externe mais des correspondances entre éléments de domaines différents
  - tables de correspondance (e.g. la classe C1 du domaine D1 correspond à la classe C2 du domaine 2)

MDA : exemple 2 (suite)

Commentaires

- Approche "programmation" : le PIM est très fourni, le compilateur implante
- Séduisant : effort de modélisation et non de développement
- Problème : trouver les compilateurs et framework techniques

la balle est dans le camp des fournisseurs d'outils de développement et non pas des utilisateurs

MDA : exemple 3

[RFW+04] xUML, Raistrick and al.
Kennedy-Carter, iUMLLite - http://www.kc.com

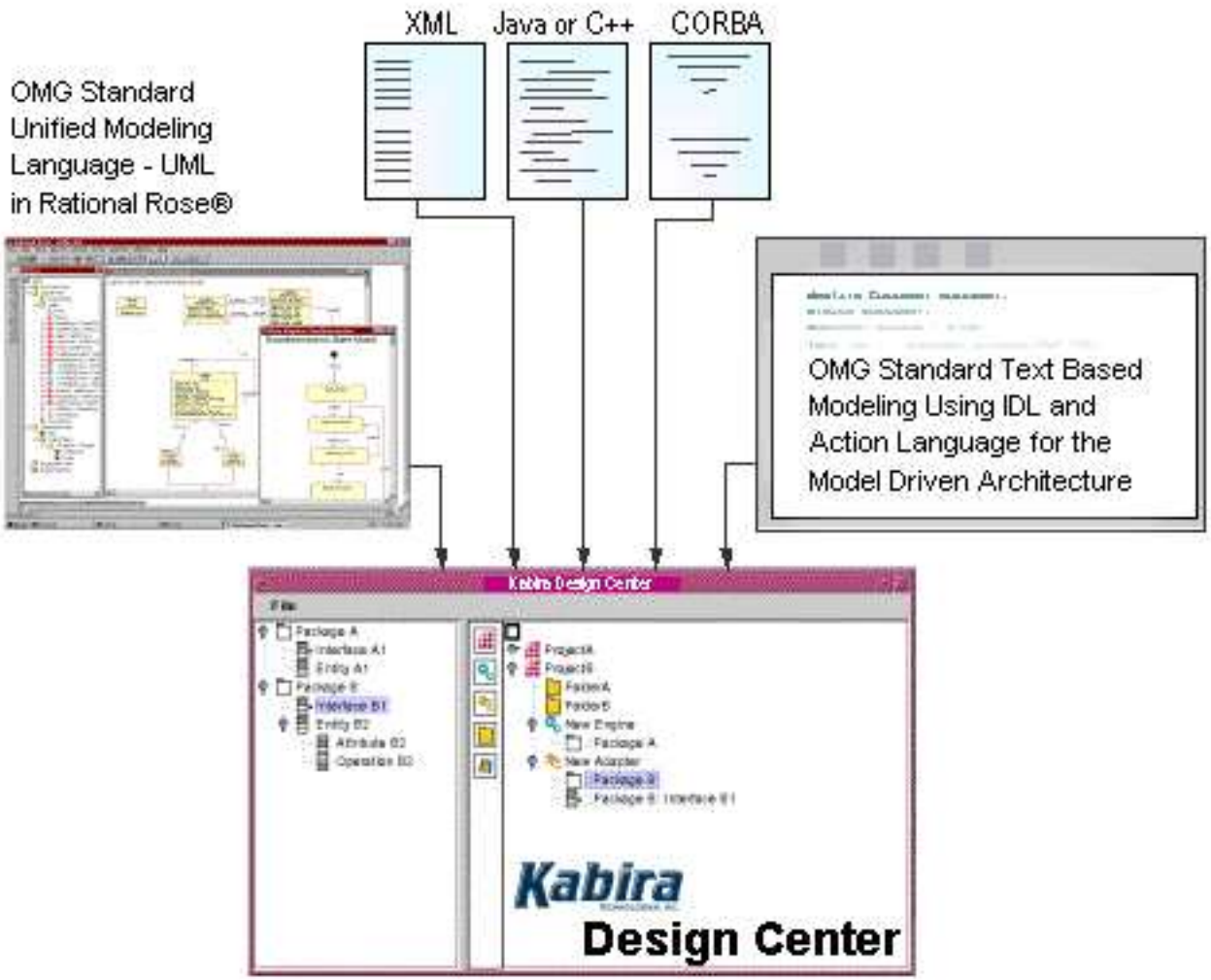
- similaire XtUML : domaines, ponts, modèles, 3 axes de modélisation,
- générateur de code configurable
- Action Specification Language
-

MDA : exemple 4

Suite de produits Kabira Design Center, Kabira Business Accelerator (KBA) Kabira, KDC - http://www.kabira.com

- Importation de modèles UML, Couplage Rational Rose
- Kabira Action Semantics (donné à part)
- Infrastructures d'accueil (solutions distribuées sur le réseau) : ObjectSwitch, Corba
  - Distribution, Thread-Management, Query , State Management,
  - Caching, Indexing, Concurrency, Memory Management,
  - Automatic Application Recovery, in-memory Transaction Management,
  - Logging, Queuing, Persistence, Deadlock Correction,
  - On the fly swapping of application logic with no loss of transactions.

MDA : exemple 4(suite)

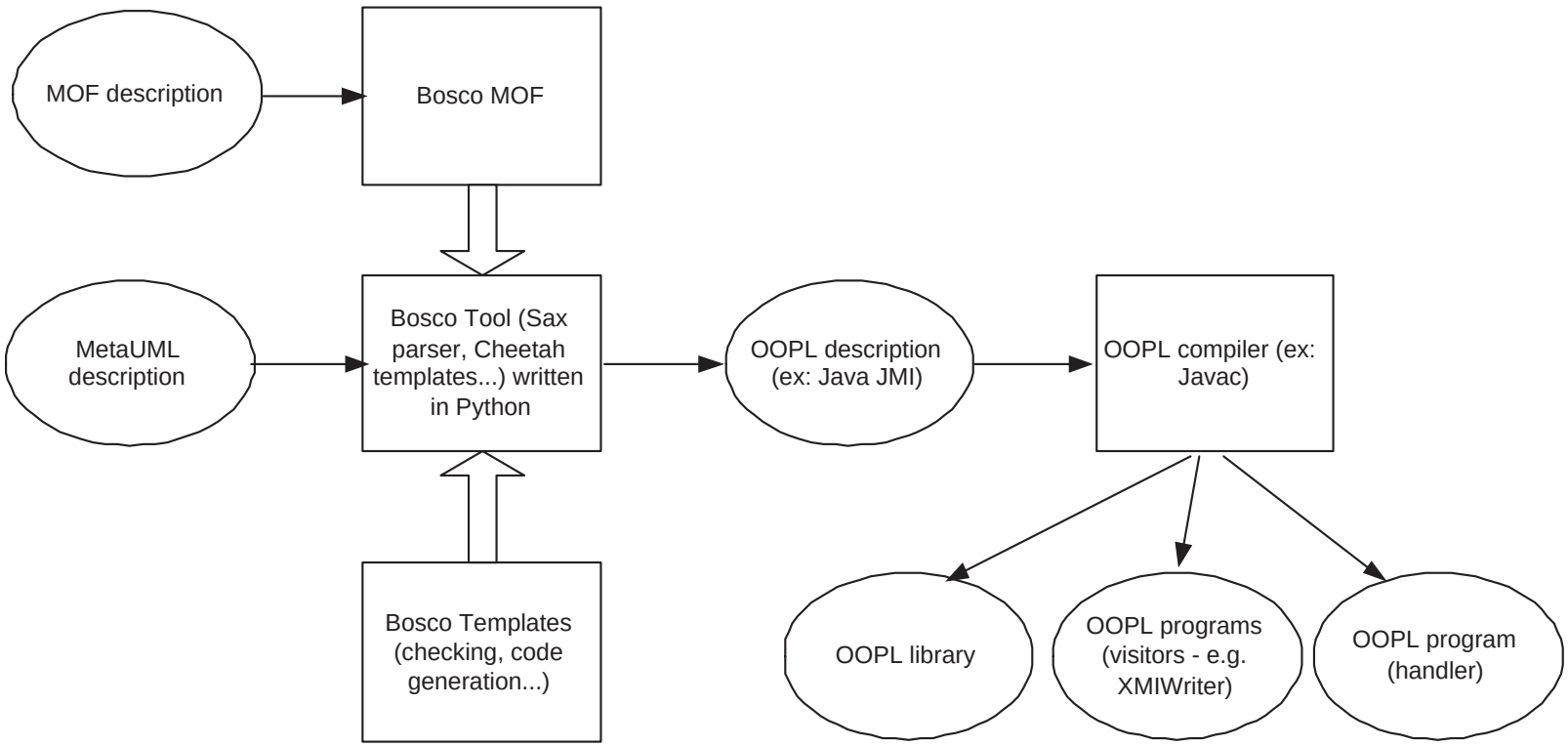


MDA : exemple 5

[AAS04] Bosco, source libre
Université de Nantes, - http://bosco.tigris.org/

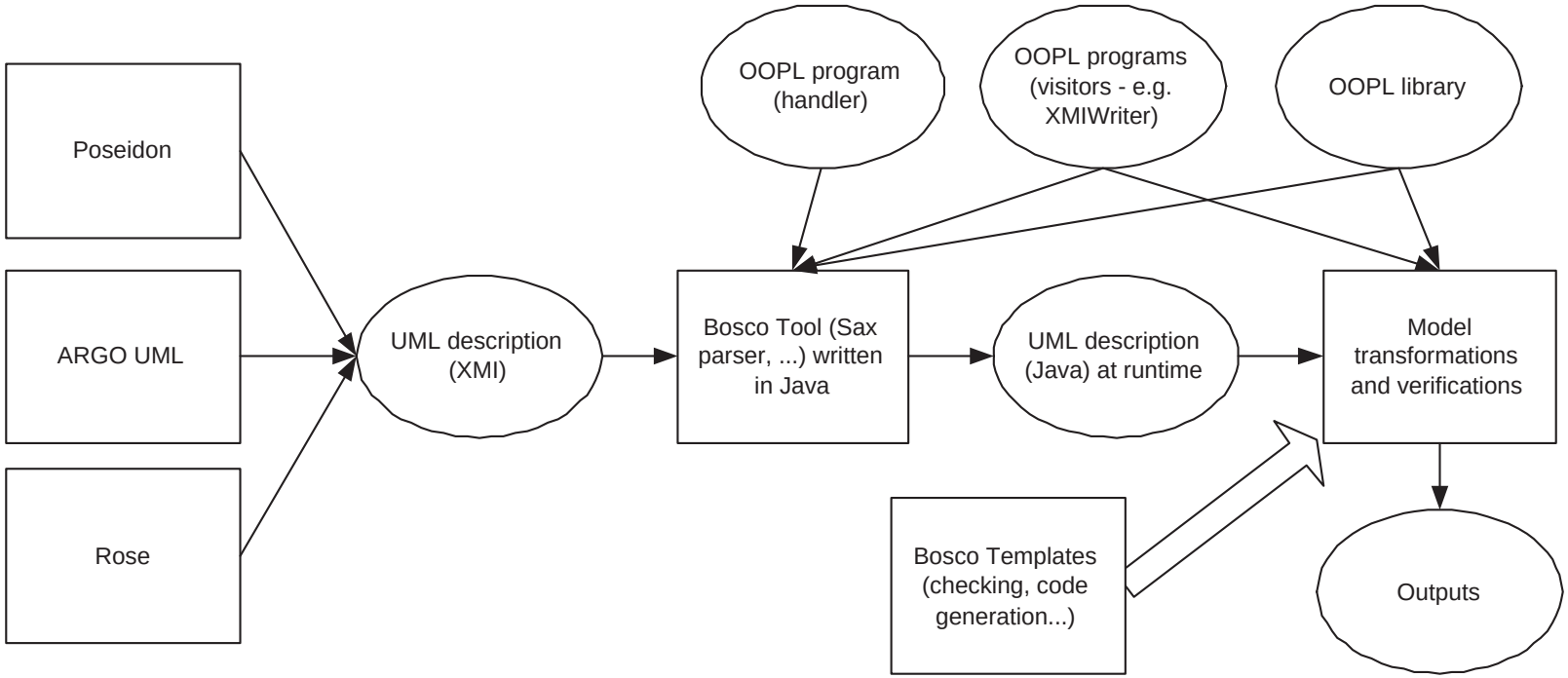
- outil de transformation de modèles
- flexibilité de modèles (acquisition de métamodèles) et évolutivité
- fusion (transformation) de méta-modèles
- efficacité et flexibilité de la génération de code (template)
- conforme aux standards (XMI, UML, OCL, MOF, JMI)
- implantation des transformations dans les métamodèles
- indépendance vis-à-vis des outils de modélisation
- ouvert (vérification, transformation...)

MDA : exemple 5 (suite)



architecture

MDA : exemple 5 (suite)



environnement généré

MDA : exemples

Outils de transformation sous Eclipse

- Kermeta http://www.kermeta.org/
- ATL http://www.eclipse.org/atl/

Comparaison [Wan05]

Autres exemples

- http://www.omg.org/mda/committed-products.htm
- http://www.lifl.fr/~bonde/exploration.html

Bibliographie sommaire

UML, OCL
[AV01b, AV03]
[MSUW04, KWB03, Bla05, Kad05, Lan05]

Outils
[MB02]

Evolution
http://www.omg.org/technology/documents/vault.htm


Pascal André, Gilles Ardourel, and Gerson Sunye.
The Bosco Project, A JMI-Compliant Template-based Code Generator.
In W. Dosch and N. Debnath, editors, Proceedings of the 13th International Conference on Intelligent and Adaptive Systems and Software Engineering, pages 157–162, July 2004.
ISBN 1-880843-52-X.


Pascal André and Alain Vailly.
Conception de systèmes d'information ; Panorama des méthodes et des techniques, volume 1 of Collection Technosup.
Editions Ellipses, 2001.
ISBN 2-7298-0479-X.


Pascal André and Alain Vailly.
Spécification des logiciels ; Deux exemples de pratiques récentes : Z et UML, volume 2 of Collection Technosup.
Editions Ellipses, 2001.
ISBN 2-7298-0774-8.


Pascal André and Alain Vailly.
Exercices corrigés en UML ; Passeport pour une maîtrise de la notation., volume 5 of Collection Technosup.
Editions Ellipses, 2003.
ISBN 2-7298-1725-5.


Mokrane Bouzeghoub, George Gardarin, and Patrick Valduriez.
Les Objets.
Eyrolles, 1997.
2e édition, ISBN 2-212-08957-0.


Xavier Blanc.
MDA en action - Ingénierie logicielle guidée par les modèles.
Architecte logiciel. Eyrolles, 1 edition, 2005.
ISBN 2-212-11539-3.


Hubert Kadima.
MDA - Conception orientée objet guidée par les modèles.
InfoPro. Dunod, 1 edition, 2005.
ISBN 2-10-007356-7.


Anneke Kleppe, Jos Warmer, and Wim Bast.
MDA Explained: The Model Driven Architecture: Practice and Promise.
Object Technology Series. Addison-Wesley, 1 edition, 2003.
ISBN 0-321-19442-X.


Kevin Lano.
Advanced Systems Design with Java, UML and MDA.
Computer Science. Elsevier, 1 edition, 2005.
ISBN 0-7506-6496-7.


Stephen J. Mellor and Marc J. Balcer.
Executable UML: A Foundation for Model-Driven Architecture.
Object Technology Series. Addison-Wesley, 1 edition, 2002.
ISBN 0-201-74804-5.


Joaquin Miller and Jishnu Mukerji. (Eds).
Model Driven Approach, MDA Guide Version 1.0.1.
Technical report, Object Management Group, available at http://www.omg.org/docs/omg/03-06-01.pdf, June 2003.


Stephen J. Mellor, Kendall Scott, Axel Uhl, and Dirk Weise.
MDA Distilled.
Object Technology Series. Addison-Wesley, 1 edition, 2004.
ISBN 0-201-78891-8.


Chris Raistrick, Paul Francis, Ian Wilkie, John Wright, and Colin B. Carter.
Model Driven Architecture with Executable UML.
Cambridge University Press, 2004.
ISBN 0-521-53771-1.


W. Wang.
Evaluation of UML Model Transformation Tools.
Master's thesis, Tools. University of Vienna, 2005.