

# Introduction à l'Ingénierie dirigée par les Modèles: au-delà des IA génératives

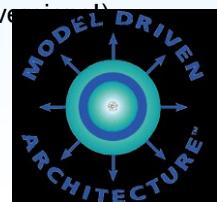
Crédit :

*Richard M. Soley, Ph.D., OMG Chairman  
Jean Bézivin, IRIN Université de Nantes  
Daniel Petisme, Michelin  
David Hill, ISIMA – Université Blaise Pascal*



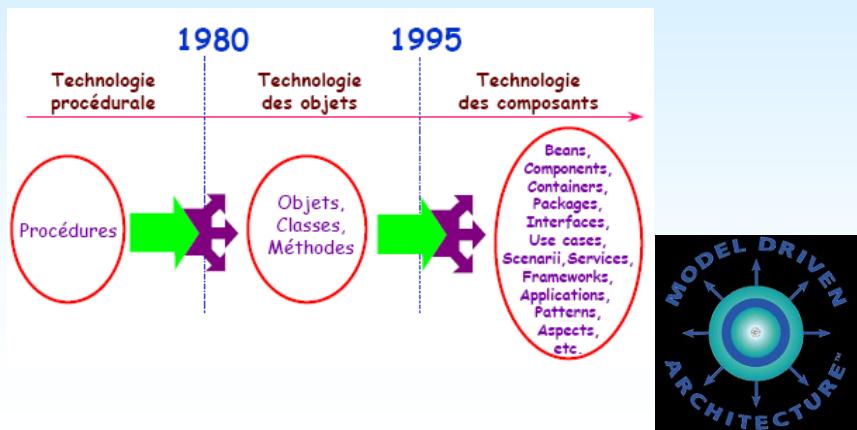
## Besoin d'automatisation et constat d'Hétérogénéité permanente

- Langages de programmation
  - ~3 millions de programmeurs COBOL,
  - ~1.6 million de programmeurs VB
  - ~1.2 millions de programmeurs Java, JS, Python
  - ~1.1 million de programmeurs C/C++
- Systèmes d'Exploitation
  - Unix, MVS, VMS, MacOS, OS X, Windows (8 versions)
  - Windows 3.1: est toujours utilisé !
  - Dispositifs embarqués (mobile, Palm)
- Réseaux
  - Ethernet, ATM, IP, SS7, Firewire, USB
  - Bluetooth, 802.11b, HomeRF



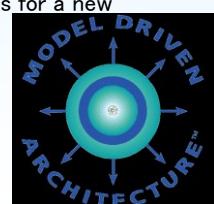


## Des modèles de complexité croissante



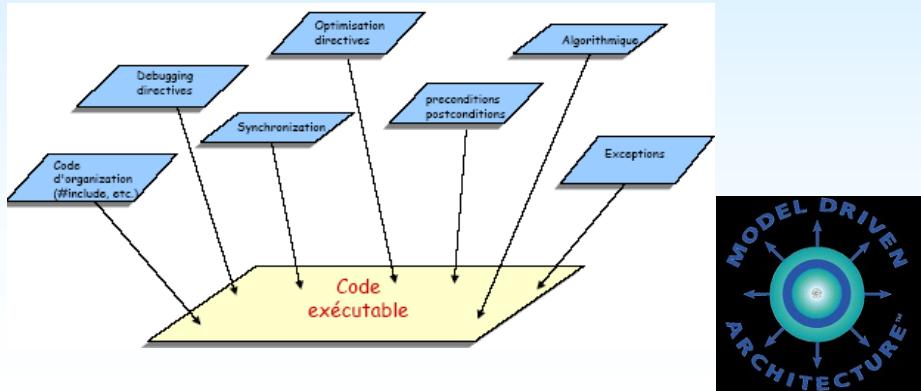
## Introduction de la programmation par aspects

« We have found many programming problems for which neither procedural nor object-oriented programming techniques are sufficient to clearly capture some of the important design decisions the program must implement. This forces the implementation of those design decisions to be scattered through-out the code, resulting in “tangled” code that is excessively difficult to develop and maintain. We present an analysis of why certain design decisions have been so difficult to clearly capture in actual code. We call the properties these decisions address **aspects**, and show that the reason they have been hard to capture is that they cross-cut the system’s basic functionality. We present the basis for a new programming technique, called **aspect-oriented programming**, that makes it possible to clearly express programs involving such aspects, including appropriate isolation, composition and re-use of the aspect code. » from Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, John Irwin. (ECOOP June 1997).



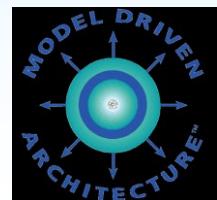


## AOP : Le code comme référentiel unique



## Où pouvons nous être d'accord ?

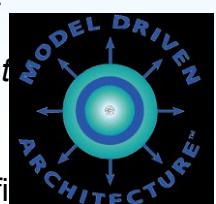
- Il n'y aura pas de consensus sur les plates-formes matérielles
  - Il n'y aura pas de consensus sur les systèmes d'exploitation
  - Il n'y aura pas de consensus sur les protocoles réseau
  - Il n'y aura pas de consensus sur les langages de programmation  
  - ***Il doit y a voir un consensus sur les interfaces et l'interopérabilité !***






## La mission de l'OMG depuis 1989

- Développer une architecture, utilisant la technologie des objects pour l'intégration d'applications distribuées garantissant :
  - La réutilisabilité des composants
  - Interopérabilité & la portabilité
  - L'emploi des logiciels commercialement disponibles
- Les spécifications doivent être *gratuitement* disponibles
- Des implémentations doivent exister
- Contrôlées par des membres « not-for-profit »



[HOME](#)
[SITE MAP](#)
[LEGAL](#)







+Q


  
OBJECT MANAGEMENT GROUP®
 



**30**  
 Years
 

**IGNITING INNOVATION**

**CELEBRATING 30 YEARS OF INNOVATION**

**OMG 30TH ANNIVERSARY**

 Watch a video, highlighting the past 30 years, and read what our members are saying about their experiences.
 

[READ MORE](#)

UPCOMING MEETING
OMG 30TH ANNIVERSARY
WHAT'S TRENDING AT OMG
HEALTHCARE & BPM
OMG CERTIFICATION
UNIFIED MODELING LANGUAGE
WEBINAR SERIES

**OMG**  
OBJECT MANAGEMENT GROUP

**ISIMA**

## Bien des entreprises du numérique ont défilées à l'OMG en + de 30 ans

Model Driven Architecture™

**OMG**  
OBJECT MANAGEMENT GROUP

**ISIMA**

## Les succès principaux de l'OMG

- Unified Modeling Language
  - UML™ reste le seul langage mondial de modélisation standardisé
- XML Metadata Interchange
  - XMT™, le standard XML-UML
- **Meta-Object Facility**
  - MOF™, standard de métamodélisation UML
  - ...

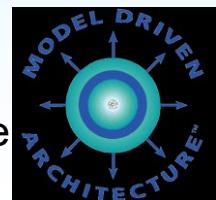
Some old or less known proposal

- Common Warehouse Metamodel
  - CWM™, intégration des deux dernières initiatives mondiales d'entreposé de donnée.
- Common Object Request Broker Architecture
  - CORBA® reste le seul langage et la seul plate-forme neutre pour l'interopérabilité...



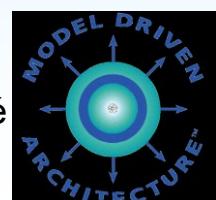
## Malgré tout la diversité continue...

- Le Middleware a proliféré :
  - CORBA®: Vendor, OS & language independent middleware
  - COM/DCOM/MTS (Microsoft Transaction Server)
  - Java/EJB
  - XML/SOAP
  - C#/Net
  - Quel sera le prochain intergiciel de pointe ???
- Constat : Il faut préserver son investissement logiciel malgré les changements d'infrastructure



## Comment protéger l'investissement logiciel ?

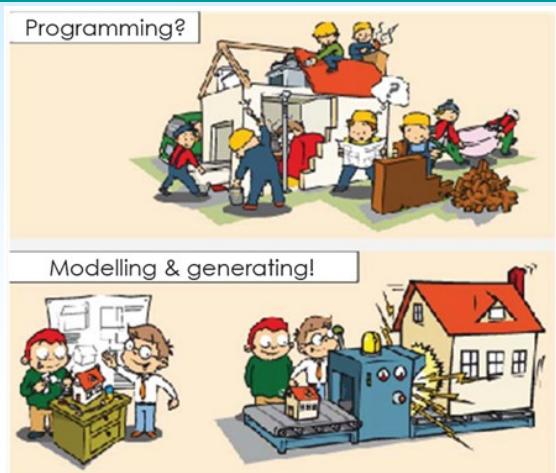
- De nombreux problèmes demeurent
  - Être à l'affût de la dernière technologie pour étudier son impact.
  - Protéger son investissement en s'appuyant sur la base logicielle existante
  - Garder le personnel qualifié
  - Maintenir le code existant
  - ...
- Intégrer ce que vous avez développé
  - Avec ce que vous développerez !





## **Une solution possible...**

# L'Ingénierie Générative



# « Everything is a model ? »

## L'Ingénierie Dirigée par les Modèles

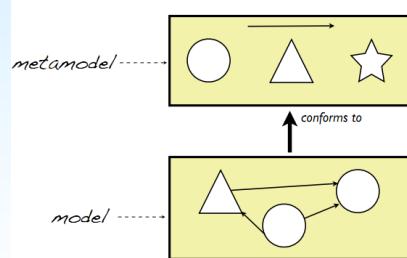
*« One and three chairs »*  
Joseph Kosuth, 1965



*« A model is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system »* Bézivin & Gerbé, 2001

# Métamodéliser ?

Layer	Description	Example
meta-metamodel	The infrastructure for a meta modeling architecture. Defines the language for specifying metamodels.	<i>MetaClass, MetaAttribute, MetaOperation</i>
Metamodel	An instance of a metametamodel. Defines the language for specifying a model.	<i>Class, Attribute, Operation, Component</i>
Model	An instance of a metamodel. Defines a language to describe an information domain.	<i>StockShare, askPrice, sellLimitOrder, StockQuoteServer</i>
user objects (user data)	An instance of a model. Defines a specific information domain.	<i>&lt;Acme_SW_Share_98789&gt;, 654.56, sell_limit_order, &lt;Stock_Quote_Svr_32123&gt;</i>



*« A meta-model is a model  
that defines the language for expressing a model »*

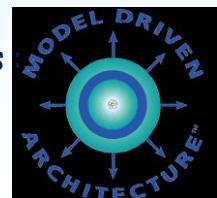
Bézibin, Gerbé, 2001

## Différentes vue d'un Métamodèle

Le **métamodèle** est un concept employé avec des sens différents suivant les domaines : informatique, statistique, Programmation Neuro-Linguistique (PNL : démarche en psychologie qui cherche à modéliser les compétences cognitives et comportementales des humains).

En l'informatique la notion de métamodèle est utilisée principalement dans 3 domaines

1. L'intelligence artificielle,
2. La programmation par objets (qui vient de la simulation)
3. Le génie logiciel.



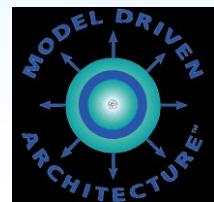


## Métamodèles en informatique 1/2

En **intelligence artificielle**, et plus précisément en représentation des connaissances, un métamodèle décrit la structure des modèles et permet de raisonner sur les modèles comme sur les connaissances de premier niveau.

Dans la **programmation par objets**, un métamodèle signifie littéralement modèle du modèle. Un modèle qui en décrit un autre.

Les métaclasses décrivant des classes, classes décrivant des objets (au sens méta-programmation).

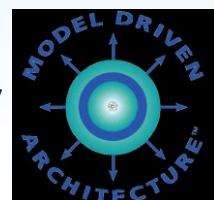


## Métamodèles en informatique 2/2

En **génie logiciel** avec UML que la notion de métamodèle décrit tous types de modèles et UML peut se décrire lui-même. (**mini exercice en cours**)

Un métamodèle qui se décrit lui-même est dit **réflexif**.

Cette propriété permet d'avoir une représentation complète des modèles sans régression infinie : modèle, métamodèle, métaméta-modèle (avec la notion de **méta-circularité**...)



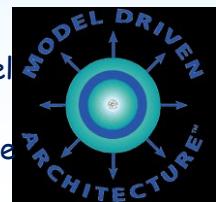
## Rappel sur la notion de modèle

Les modèles sont des **représentations** d'un point de vue particulier (d'une conception, d'une théorie, ...) sur un système sujet d'études. Ils sont conçus en fonction d'un ou plusieurs objectifs (les questions que l'on se pose)

Ils sont écrits dans un code (un langage, un formalisme...) approprié à l'expression et à l'usage des connaissances qu'ils véhiculent.

Une carte IGN, une partition musicale, une équation " $E=mc^2$ ", une maquette, un logiciel de simulation... sont des exemples de modèles.

La notion de "Modèle" est donc directement liée à celles de "Formalisme", de "Point de Vue/Théorie" et de "Sujet d'études/Système"



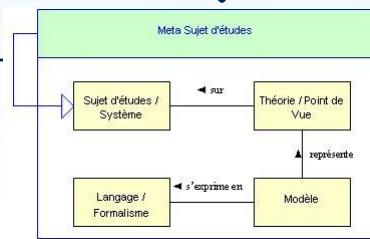
## Le point de vue théorique (1/4)

Soit une **structure S** composée des **4 notions**:

1. Le sujet d'études,
2. Le Formalisme,
3. Le point de Vue,
4. Le Modèle.

Considérons cette structure S comme un nouveau sujet d'études, un méta-sujet d'études.

Il est alors tout à fait concevable et légitime de bâtir des (méta)points de vue sur tout ou partie de ce nouveau sujet d'études et de les représenter par des (méta)modèles exprimés dans des (méta)langages.

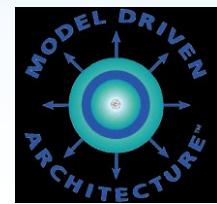
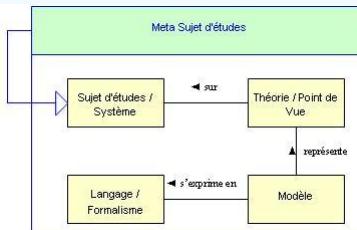


## Le point de vue théorique (2/4)

Un métamodèle peut donc être défini comme la représentation d'un point de vue particulier sur cette structure  $S$ .

Si on ne s'intéresse dans  $S$  qu'au couple {Langage, Modèle} on rejoint l'acception la plus courante de métamodèle :

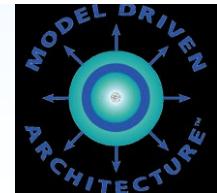
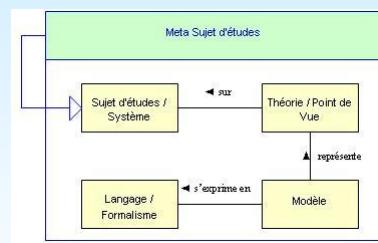
Un modèle de langage de description de modèles.



## Le point de vue théorique (3/4)

Il existe différents métalangages permettant l'écriture de tels métamodèles.

Parmi ceux-ci on peut citer entre autres le langage **BNF** (la forme de Backus-Naur) pour les grammaires des langages informatiques mais aussi le **MOF** le métalangage graphique d'inspiration UML.

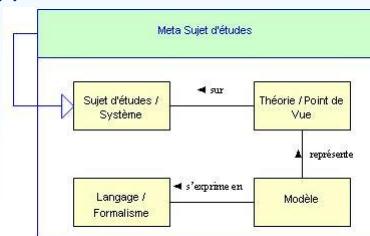


## Le point de vue théorique (4/4)

Si on s'intéresse par contre au couple {Sujet d'études, Point de Vue}, un métamodèle sera la représentation d'une opinion (d'une théorie...) portée sur des points de vue portés sur des sujets d'études (noter la récurrence).

L'épistémologie (Philo des Sc.) et l'histoire des Sciences sont des disciplines pourvoyeuses de ce type de métamodèles.

Les approches de conception d'applications informatiques (telles que l'ancien cycle en V, XP,...), leurs représentations formelles et les logiciels associés relèvent également de la métamodélisation.

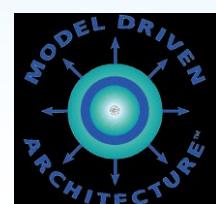


## Epistémologie ? + de détails en EDD ZZ3 TC

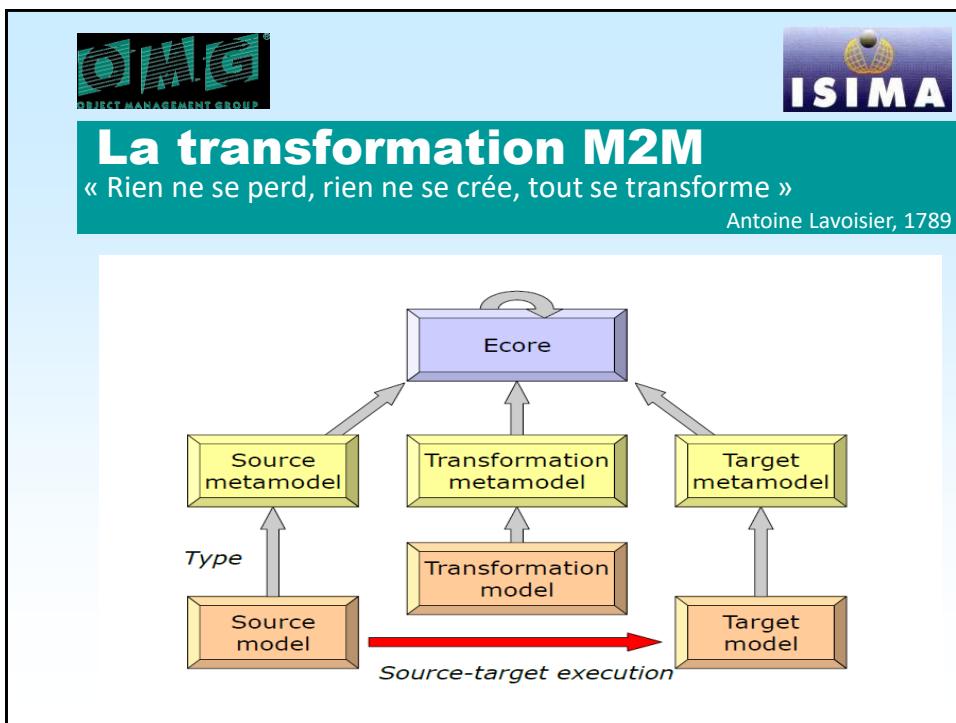
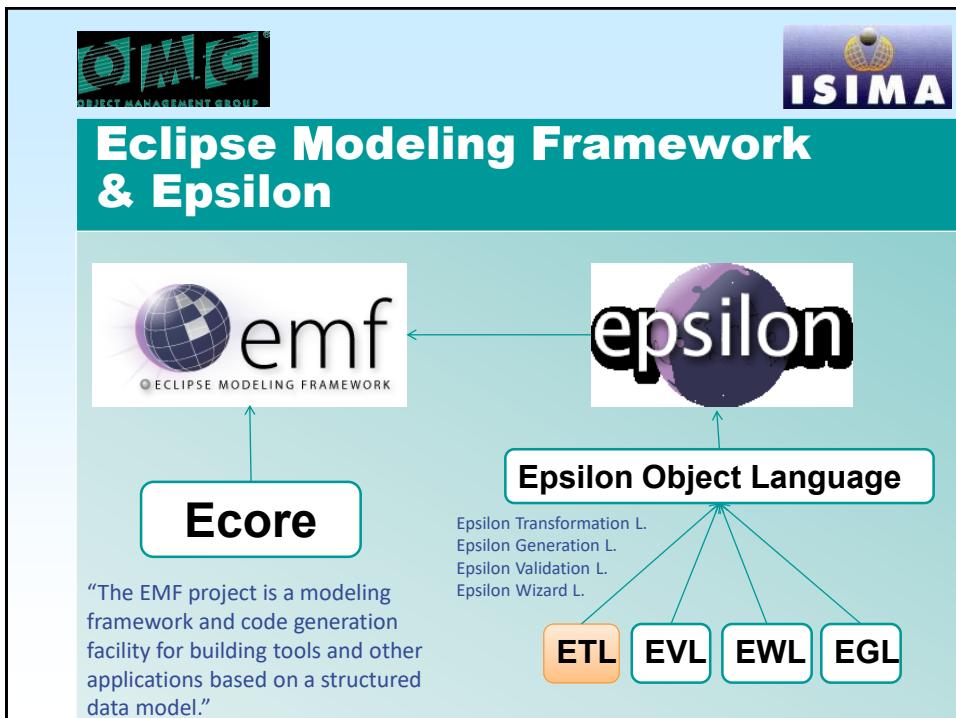
L'épistémologie: XXe siècle. Emprunté de l'anglais epistemology, lui-même composé à l'aide du grec ancien ἐπιστήμη / epistemē « connaissance, science » et de λόγος / λόγος « discours » traité.

Cela désigne en général :

- Soit: l'examen critique des principes et méthodes qui gouvernent les sciences en Philosophie des sciences.
- Soit la théorie de la connaissance en général.



La méthodologie une branche de l'épistémologie





## AI – for software Engineering : « Programs that write programs »

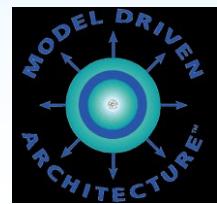
1. Compilers
2. Metaprograms

### FROM:

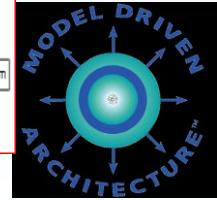
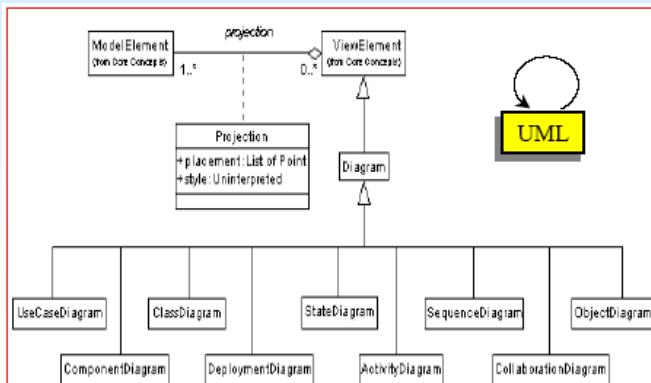
- Basic Shell scripts that write other scripts
- To C code writing
- Ending by advanced C++ TMP

### TO MDE :

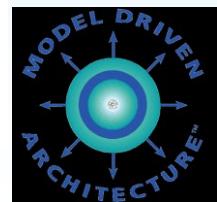
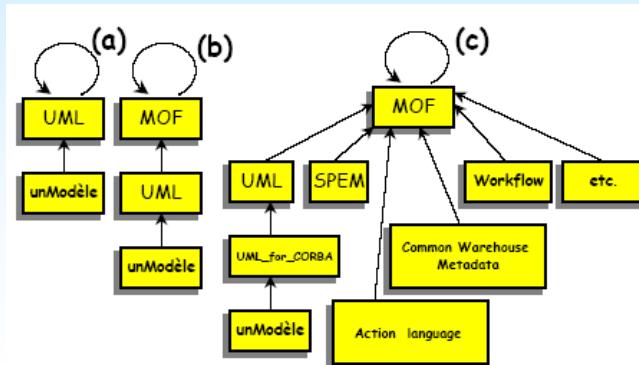
- Domain Specific Languages
- Visual programming



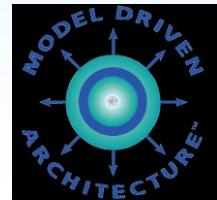
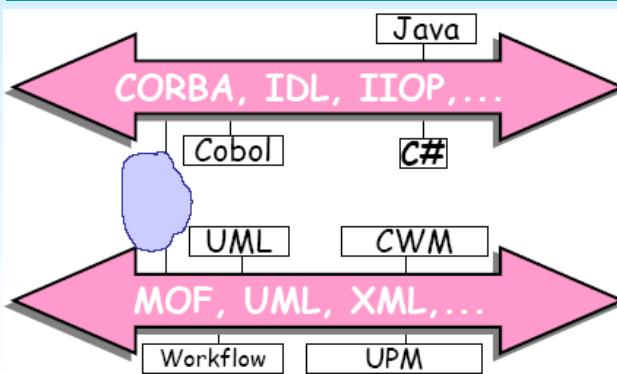
## Fragments d'un métamodèle UML



## Evolution des techniques à l'OMG

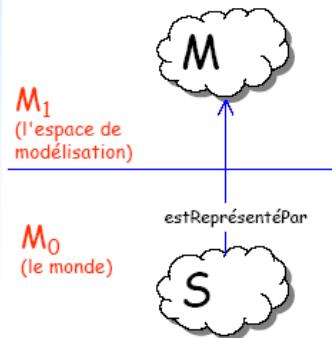


## Bus logiciel et bus de connaissance

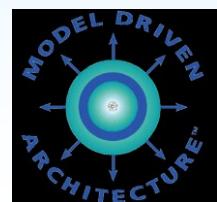


## La relation de « représentation »

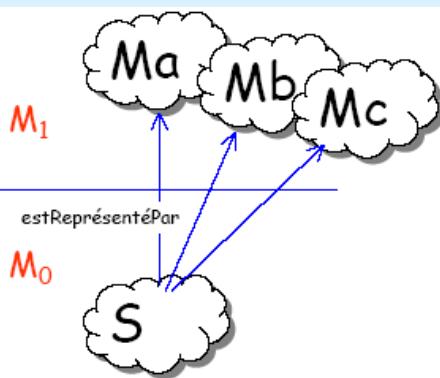
### Systèmes et modèles



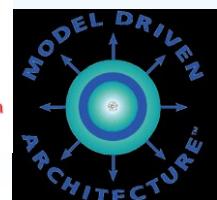
Un modèle **M** est une représentation simplifiée du monde; en fait d'une partie **S** du monde appelée le **système**.



## Aspect Oriented Software Engineering

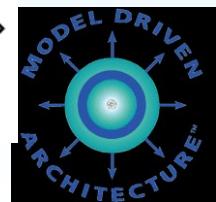
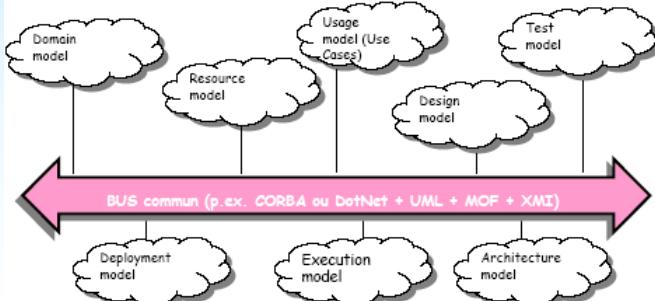


- Un système donné peut avoir plusieurs modèles différents
- Chaque modèle représente un **aspect** différent du système.
- L'appellation **AOM** (Aspect-Oriented Modeling) est donc un **pléonasme**.
- Inversement on peut combiner des modèles différents extraits du même système: **opération de tissage des aspects**.



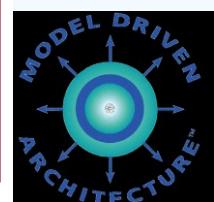
## Génie Logiciel orienté aspect

(AOSE : aspect-oriented software engineering)

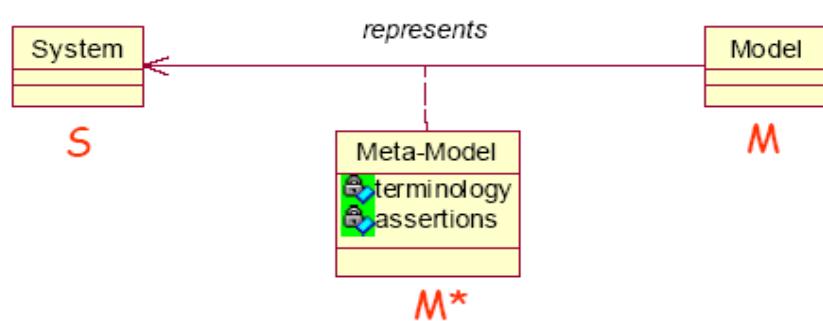


## Le but d'un modèle

- Le but d'un modèle est toujours d'être capable de répondre à certaines questions en lieu et place du système qu'il est censé représenter.
- Il doit répondre à ces questions exactement de la même façon que le système y aurait répondu lui-même, si on l'avait interrogé.

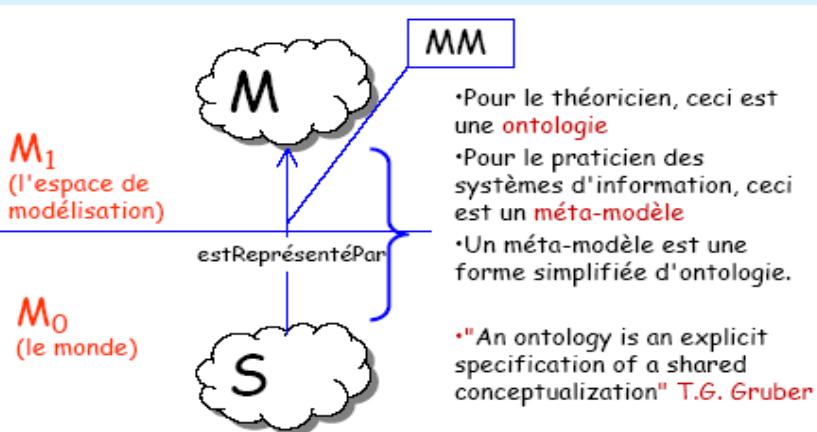


## Un méta-modèle ?

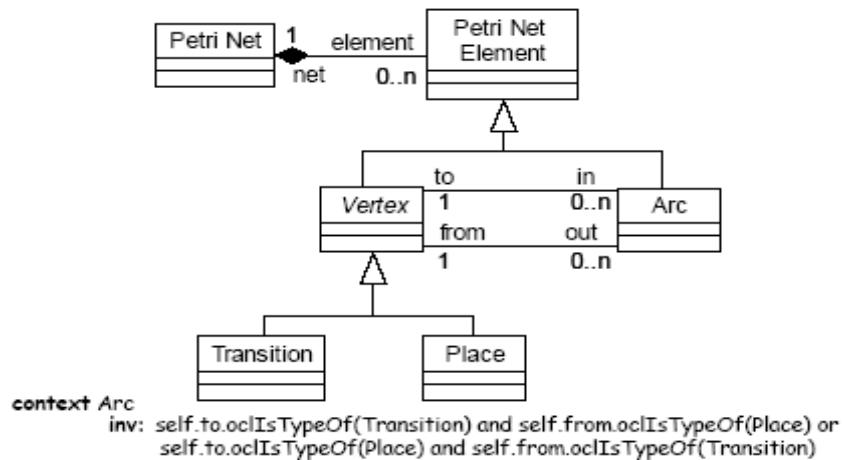


La correspondance entre un modèle et un système est définie par un méta-modèle.

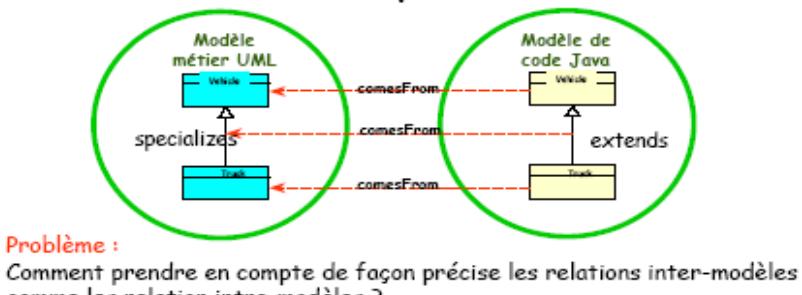
## Ontologies et méta-modèles



## Un métamodèle pour réseau de Pétri

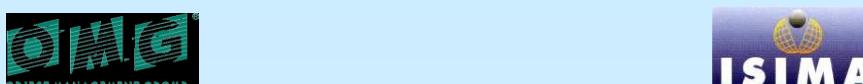


## Les éléments des différents modèles sont dépendants



**Constat :**  
Les problèmes de tracabilité, de transformation, de tissage d'aspects sont des cas particuliers de la mise en correspondance de modèles.  
Ils doivent se traiter au niveau des métamodèles.

The slide features a central title "Back to Computers... « Exécutabilité d'UML ? »" above a large blue cloud containing the text "Un modèle UML". Below the cloud is a screenshot of a terminal window showing multiple command-line interfaces. To the right of the terminal is a text box stating: "Il n'y a pas d'exécution canonique pour un modèle UML. Cependant le standard 'Action Semantics for UML' fournit un métamodèle pour définir des schémas d'exécution. Nombreuses applications (p.ex. Kennedy-Carter pour faire de la transfo de modèles et de la génération de code)". In the top left corner is the OMG logo, and in the top right corner is the ISIMA logo. On the right side of the slide are two book covers: "EXECUTABLE UML: A FOUNDATION FOR MODEL-DRIVEN ARCHITECTURE" by Stephen J. Mellor and Marc J. Balmer, and "EXECUTABLE UML: DESIGN AND PRACTICE" by Bertrand Coudert.



# Applications : Visual Paradigm,...

**Modeling and Code Generation**

```

classDiagram
    class ShoppingCart {
        @products
    }
    class Product {
        @JPA1StyledElement, TechnicalSupplements
        @products
        +productId: long ...
        +description: String ...
        +labeling: String ...
        +price: double ...
    }
    class ProductCategory {
        +district: String ...
        +country: String ...
    }
    ShoppingCart "1" -- "*" ProductCategory : parentCategory
    ProductCategory "*" -- "5" Product : category
    Product "*" -- "*" ProductCategory : <<JPAToMany>>
    ProductCategory "*" -- "*" Product : <<JPAToMany>>
    
```

```

if (!obj instanceof Product)
    return false;
final Product rhs = (Product) obj;
final EqualsBuilder builder = new EqualsBuilder();
builder.append(productId, rhs.productId);
return builder.isEquals();
}
@ManyToMany(mappedBy = "products")
public Set<ProductCategory> getCategory() {
    return this.category;
}
public String getDescription() {
    return this.description;
}
String getLabeling() {
    return this.labeling;
}
    
```

