Models Everywhere

ICEIS'2001

Jean Bézivin

Université de Nantes - LRSG
Faculté des Sciences et Techniques
2, rue de la Houssinière BP 92208
44322 Nantes cedex 3, France
Jean.Bezivin@sciences.univ-nantes.fr
http://www.sciences.univ-nantes.fr/info/lrsg/Recherche/mda/

Main point of the presentation

- ~~Objects~~ everywhere
  Models

- The latest paradigm shift in software engineering:
  - ✓ Object technology

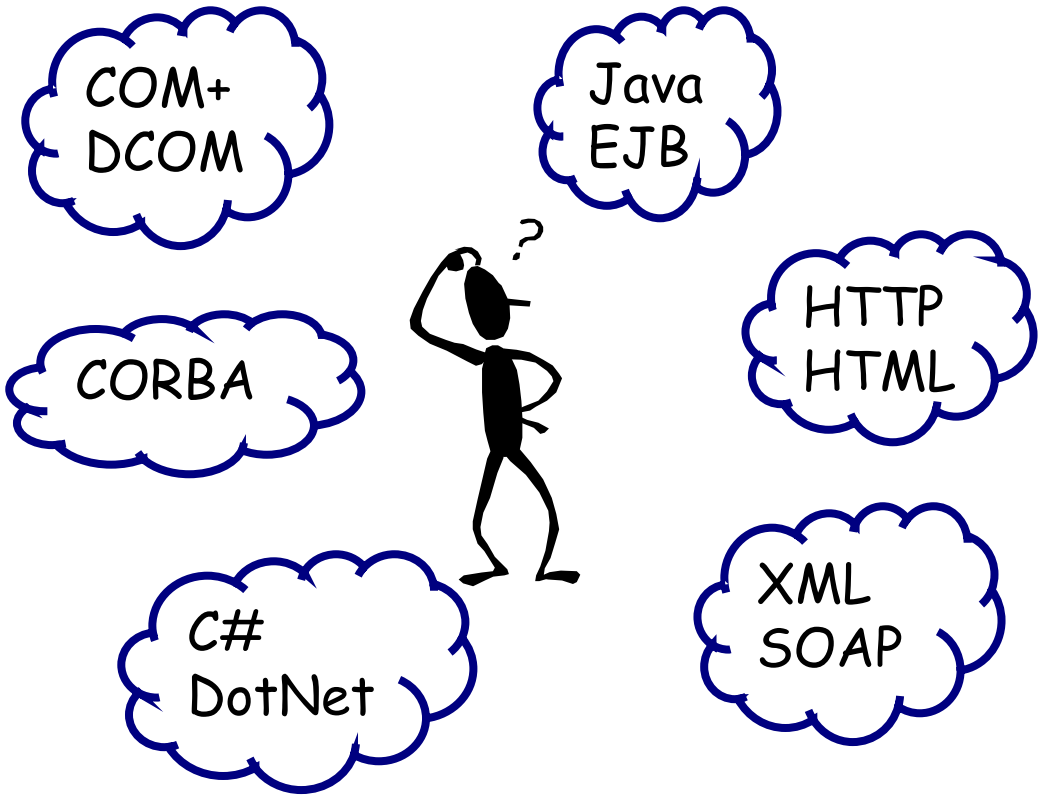
  to
  - ✓ Model technology

- How did we arrive there so quickly?
- What are the short and medium-term consequences of this move?

Outline

- The end of the middle-war
- From OMA to MDA: the new OMG vision
- Visiting the model space
- What is a model?
- What is a meta-model?
- The MOF and the four-level model stack
- Hopes and dangers of the MDA.

The middleware war is over



- There is no clear winner nor loser
- The next battlefield will be model transformation
- The OMG's Model Driven Architecture (MDA) initiative is aimed at using modelling and meta-modelling to drive the design and implementation of distributed systems.

Anger:

We don't want anymore to pay a high price for simply moving our information system to a new middleware platform (COM, CORBA, Java, HTML, XML, DotNet, etc.) when our business system stays stable.

We are prepared to pay a last price for building the abstract models of our business and services that will guarantee us against technological obsolescence.

From there, any platform provider will also have to provide the mapping solutions from standard business models before we buy.

The middleware war is over

UML and MOF compliant platform independant models

COM+ DCOM

Java EJB

HTTP HTML

CORBA

C# DotNet

XML SOAP

- MOF along with UML is a core technology for MDA.
- Technology neutral models of systems can be mapped to implementations that use a variety of middleware technologies.

Some of the OMG successes

1. CORBA
2. IDL
3. IIOP
4. UML
5. MOF
6. XMI
7. CWM
8. UPM/SPEM

yesterday
OMA

today

?

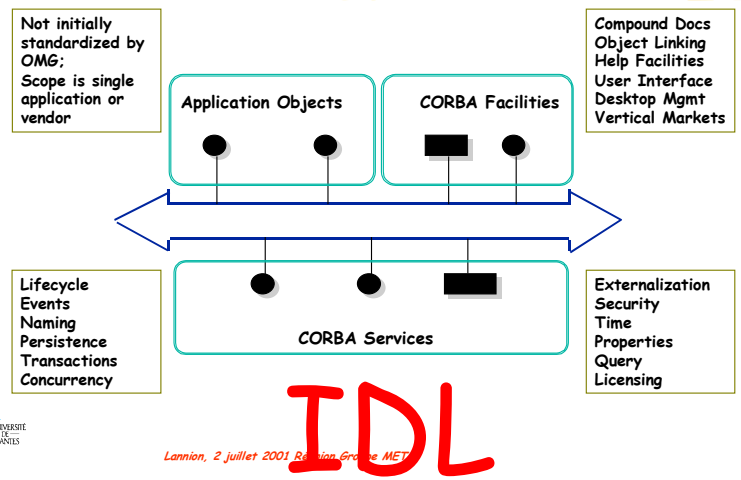
tomorrow
MDA

Yesterday : OMA

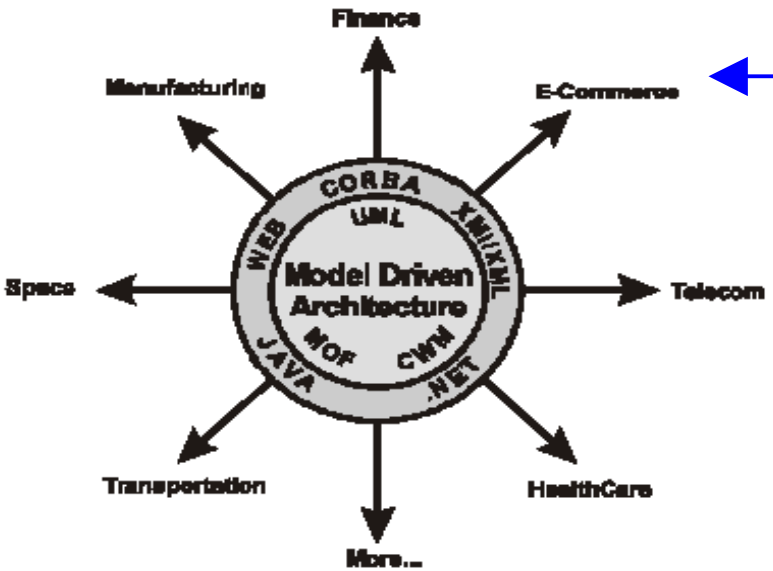
- Outlines the *Object Management Architecture*, contains foundation of standards including:
  - ✓ Overview of integration problem, with reasons for object-oriented solution.
  - ✓ Objectives of the standards group.
  - ✓ Abstract object model.
  - ✓ Reference model (architecture).
  - ✓ Glossary of terms.

In the 90's, there was a hope that a common and unique object scheme could be found.

Object Request Broker

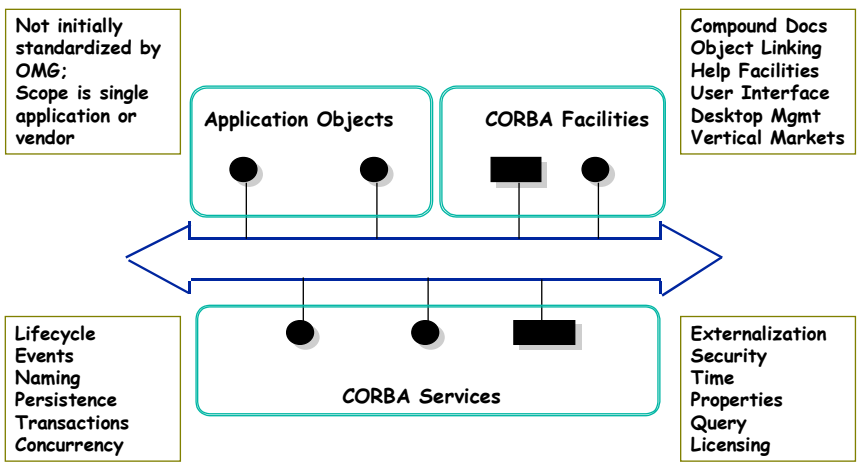


Compare code-centric and model-centric approaches



MDA
OMA

Object Request Broker



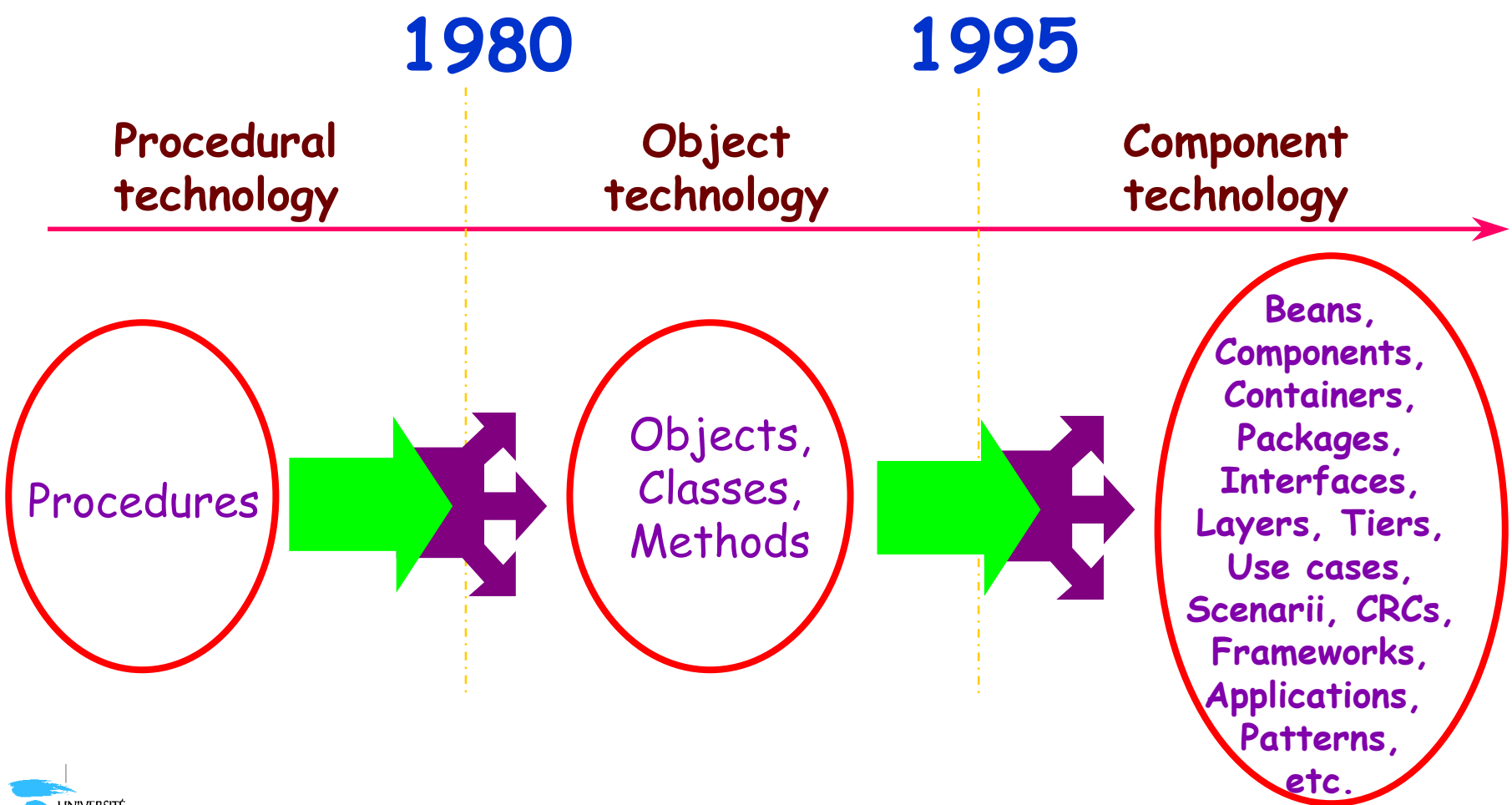
IDL --> UML

We lied about objects

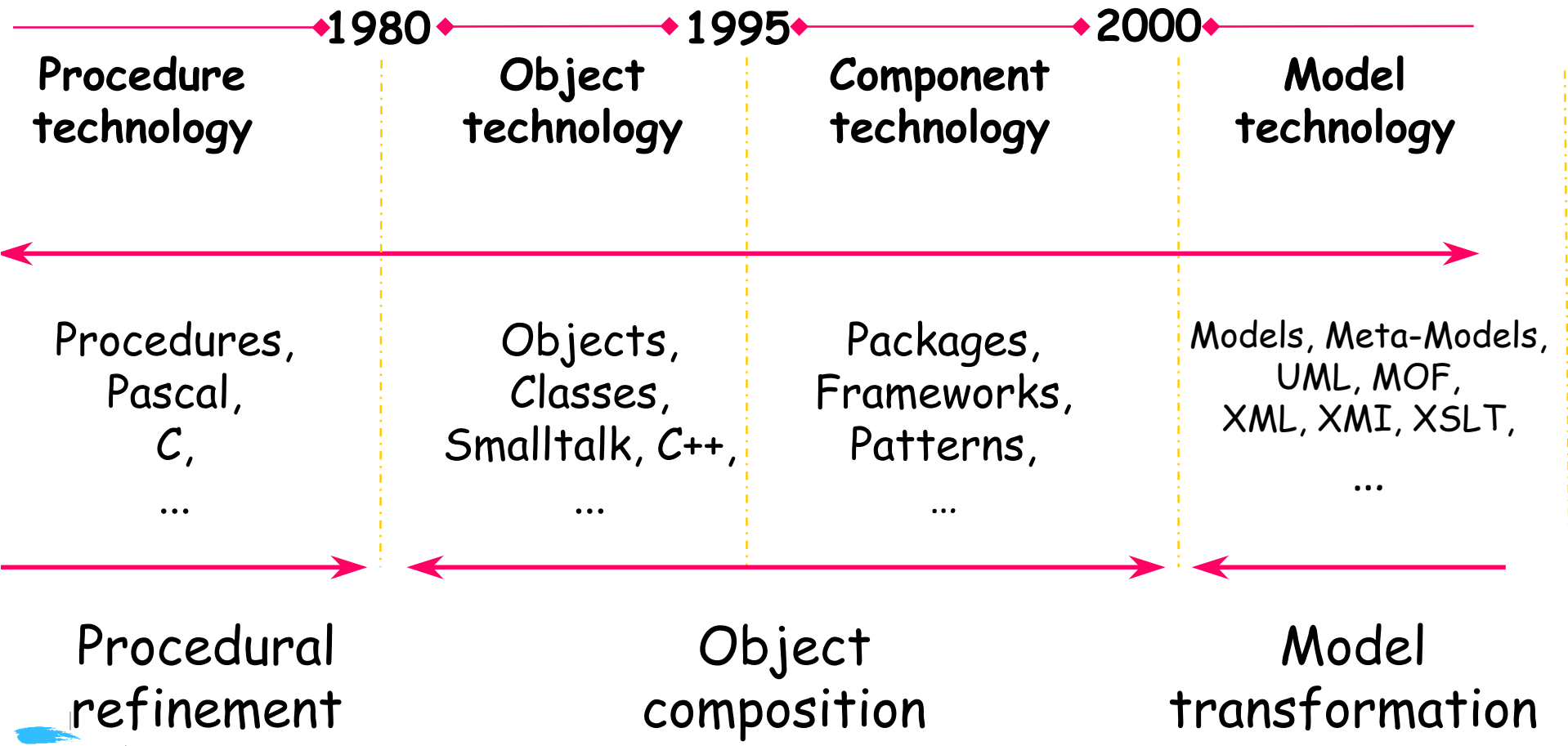
Because of the unifying capability of the object paradigm, changing from procedural to object technology will bring **huge conceptual simplification** for the software engineer. Because everything will be an object, we shall witness a **dramatic reduction** in the number of necessary concepts.

Anonymous, circa 1980

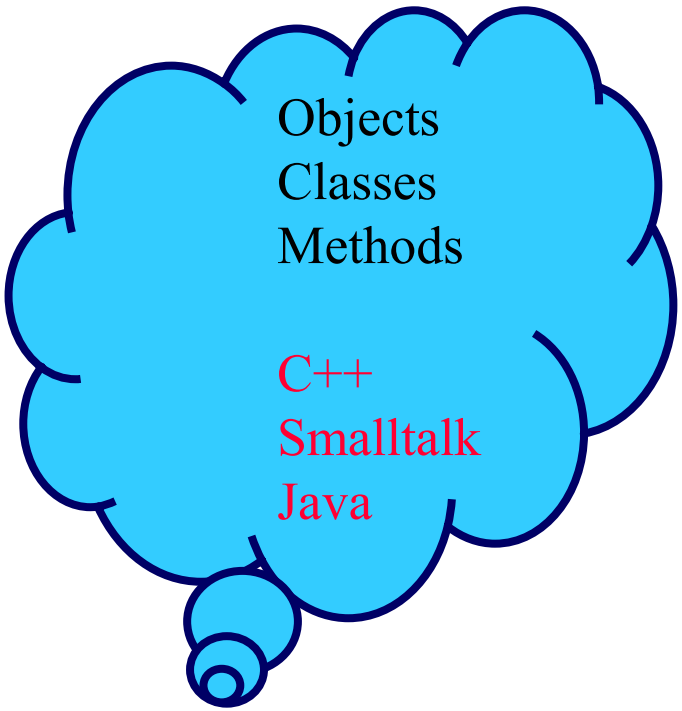
Models of increasing complexity



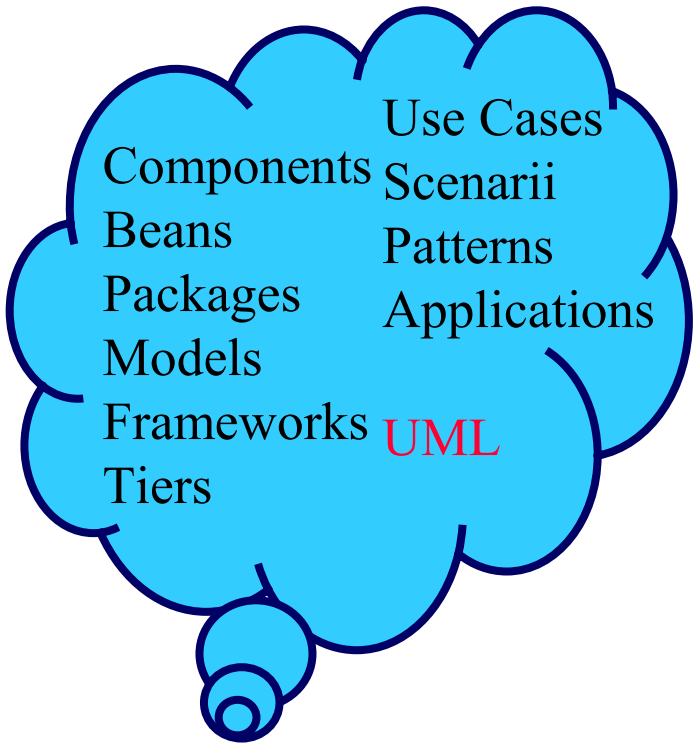
From OMA to MDA



Objets vs. Components ?

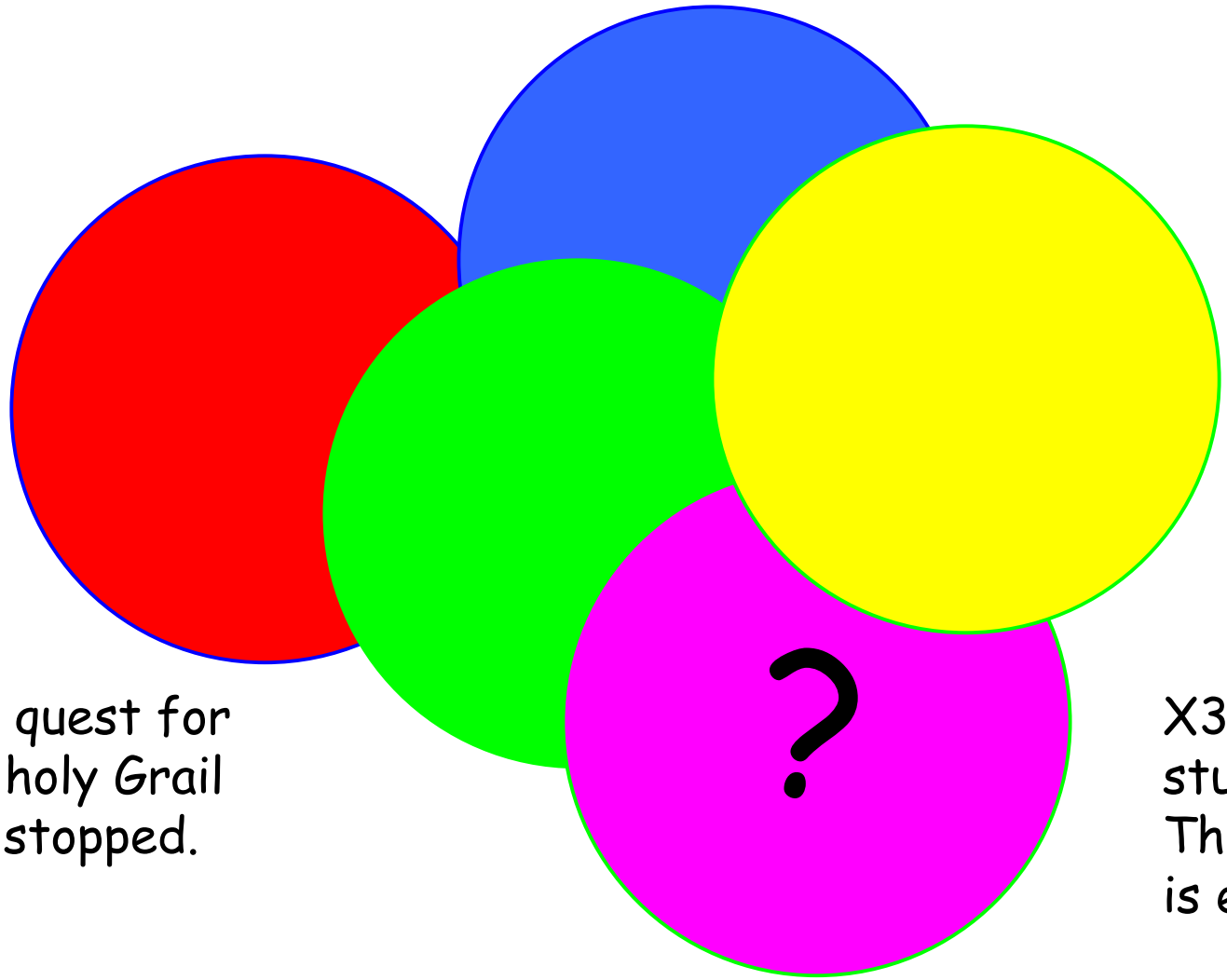


"run-time objects"



"development-time objects"

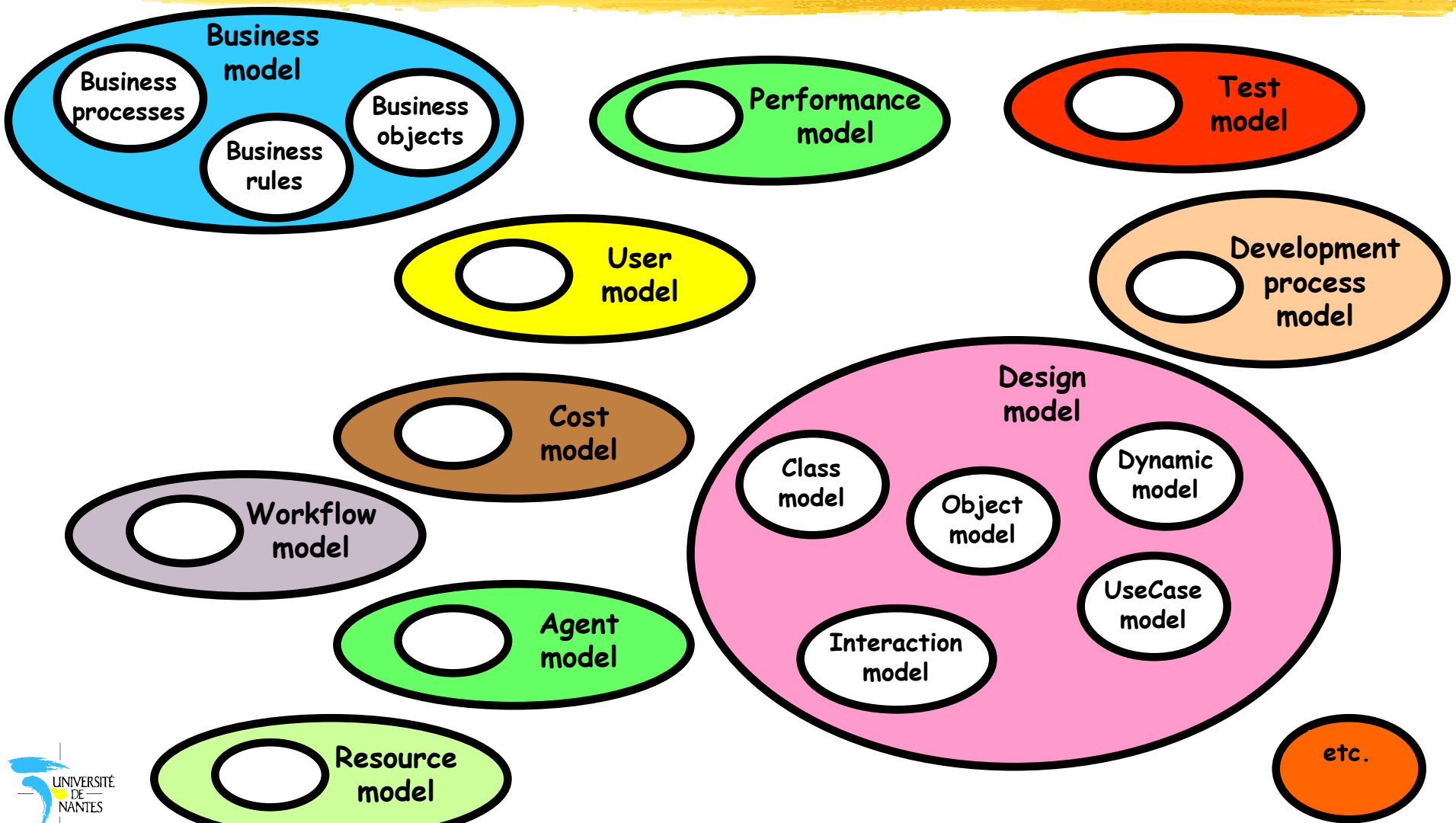
There is no unique minimal object "model"



The quest for the holy Grail has stopped.

X3H7 matrix study:
The intersection is empty

Models Everywhere



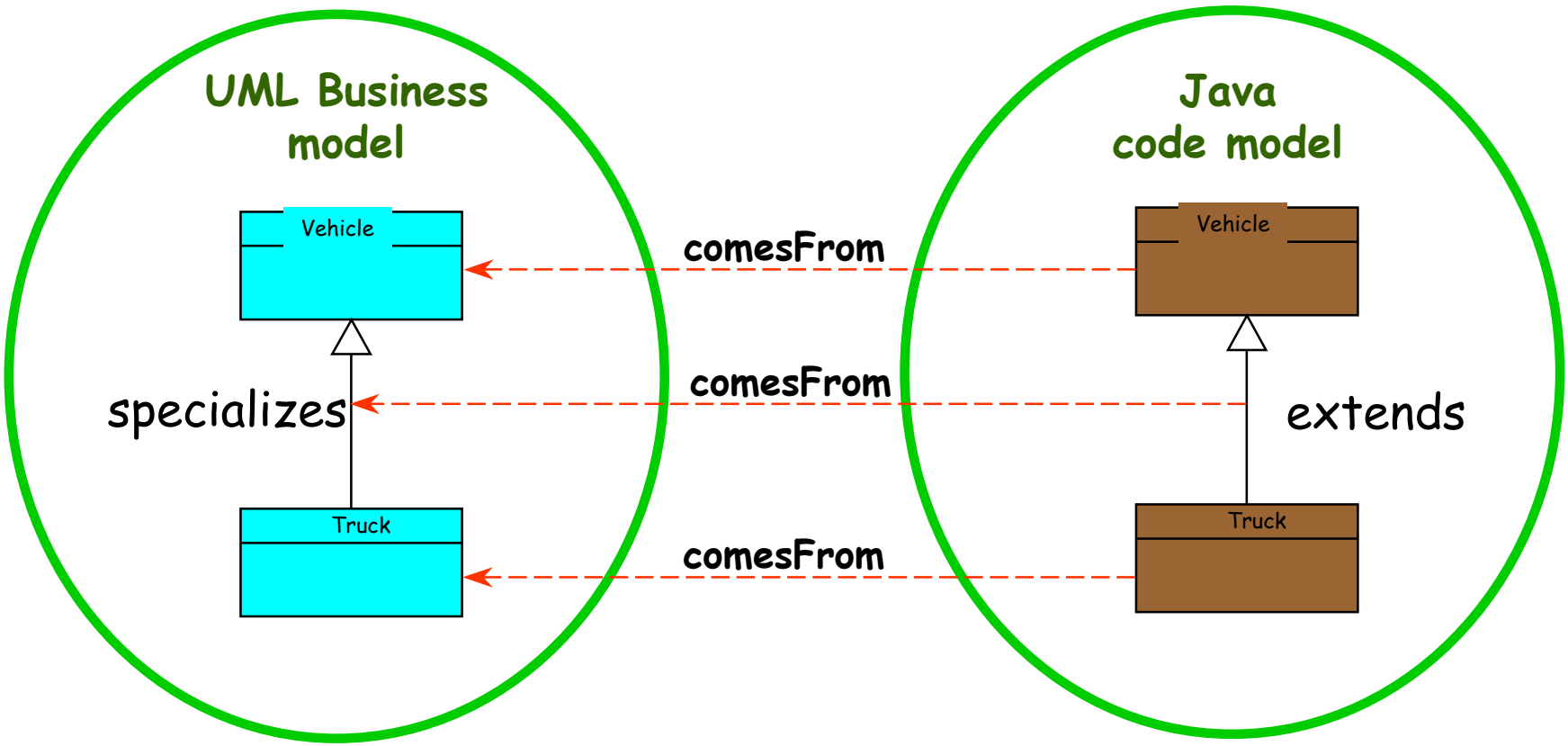
The global model space

- The development software cycle is populated with models
- Models are of unequal importance
- The model space is structured
- Models are linked in a complex organization network
- The content of each model is defined (constrained) by a corresponding meta-model (ontology)
- The model space is constantly broadening starting from the essential models (Domain, Service, Resource)

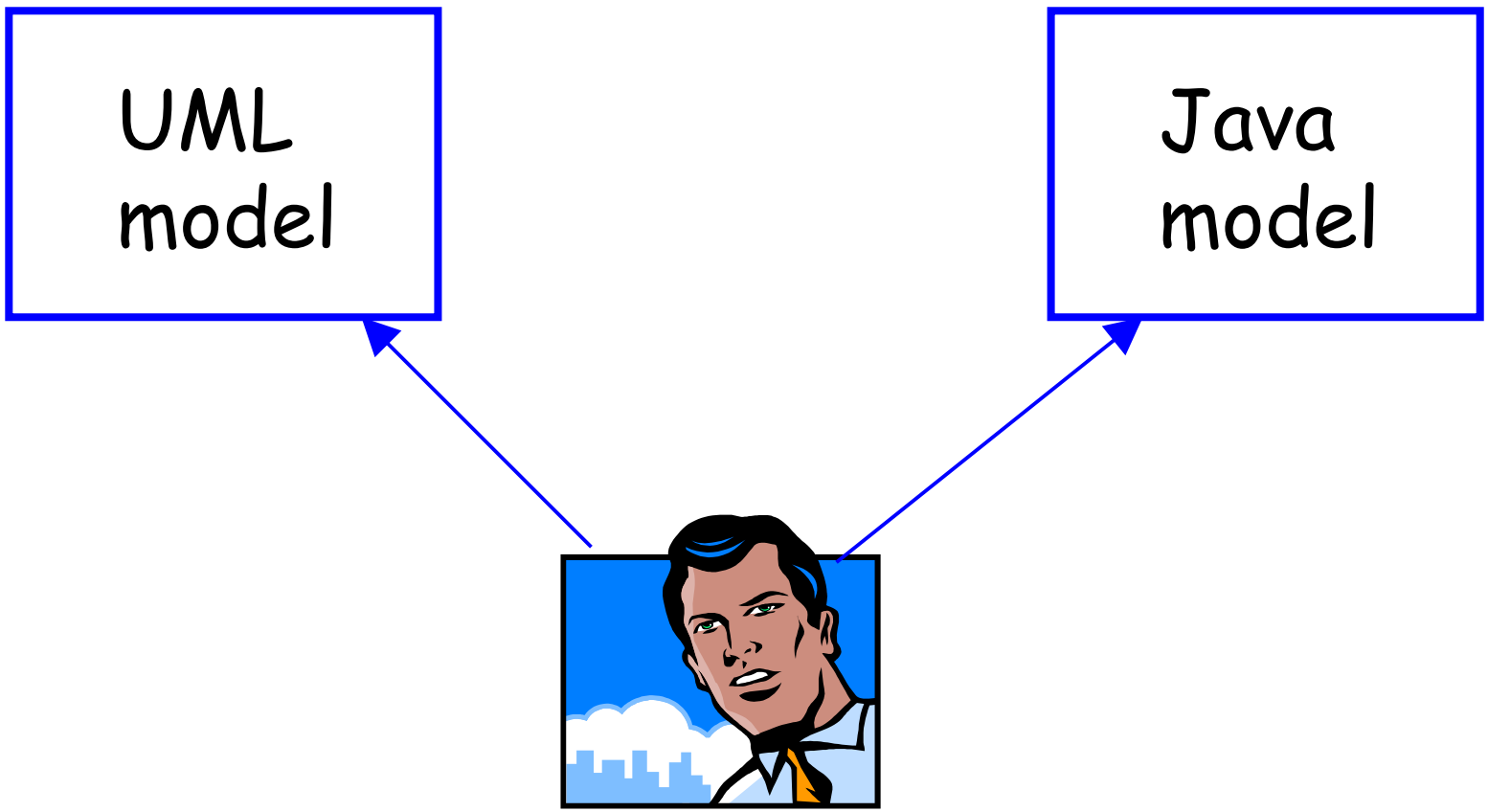
Each model has different characteristics

- A Smalltalk coding model only offers single inheritance, but also explicit anonymous meta-classes.
- An Eiffel coding model has a different form of inheritance and several extensions (contracts, etc.).
- A Java or C# coding model has two notions of inheritance, corresponding to the class and interface categories.
- A C# coding model allows cross-language inheritance
- A workflow model is built from basic tasks.
- A usage model contains the concepts of actors, use-cases and several relations like specialization of use-cases.
- etc.

Elements from different models are dependent



Consequence: having to deal simultaneously with several models of different semantics



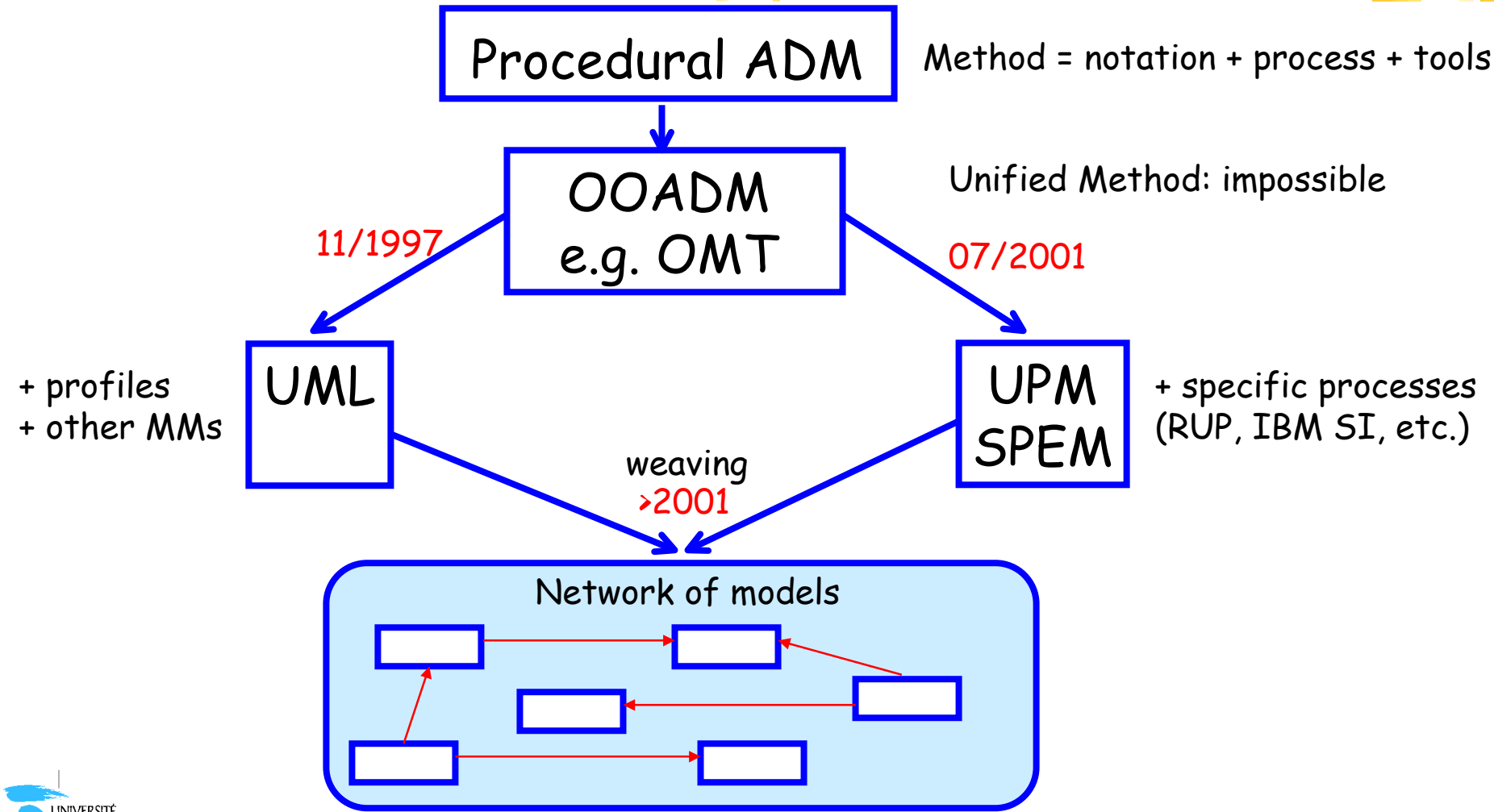
UML opened the road, several roads indeed...

From Object-Oriented Programming
to
Model-Based Software Engineering.

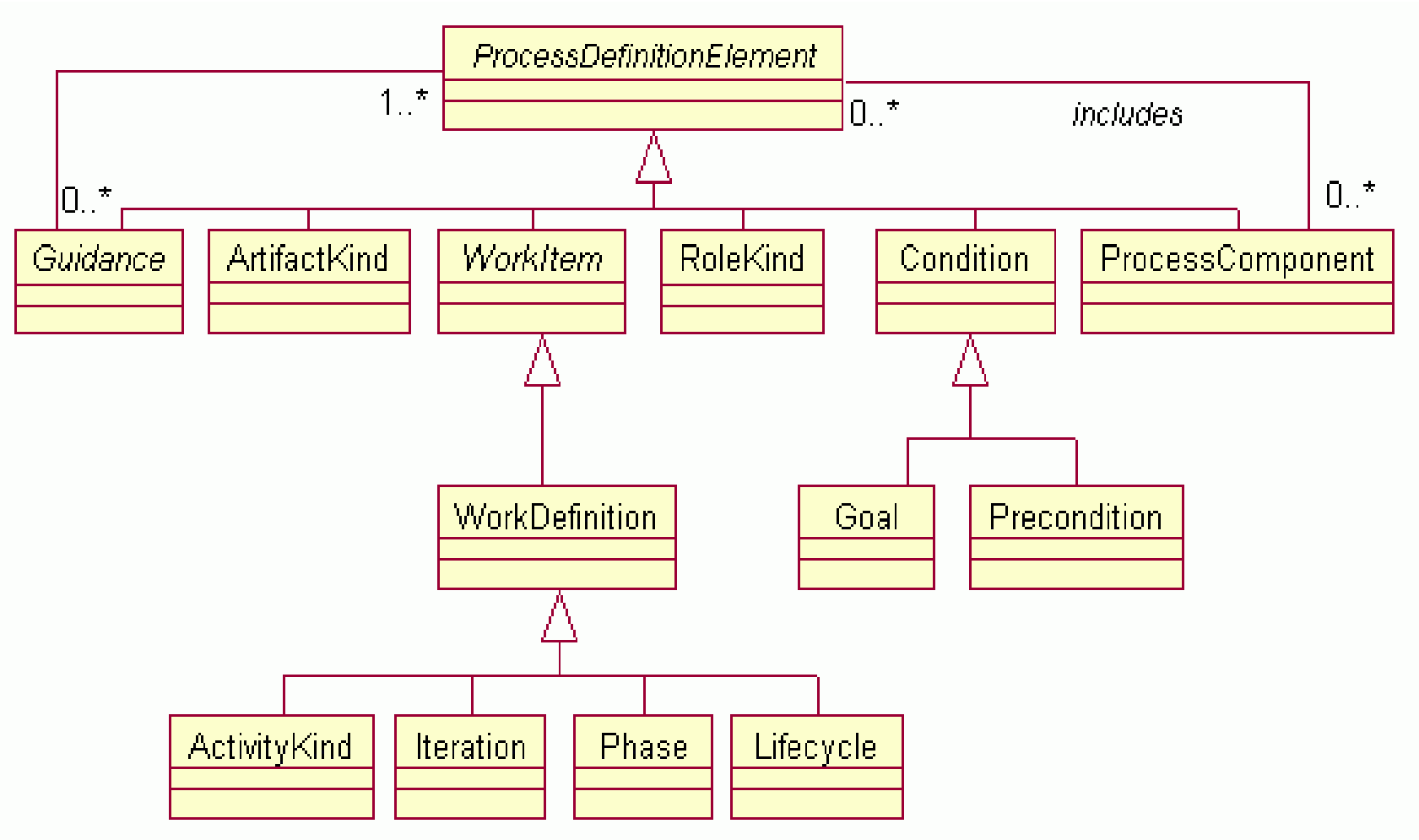
product
and
process
models
+
model
engineering

OMT
SA/RT
SADT
ERD
Merise
DFD
UFD
etc.
JSD

The roadmap


Procedural ADM
Method = notation + process + tools
OOADM
e.g. OMT
Unified Method: impossible
11/1997
07/2001
UML
+ profiles
+ other MMs
UPM
SPEM
+ specific processes
(RUP, IBM SI, etc.)
weaving
>2001
Network of models

The SPEM/UPM meta-model

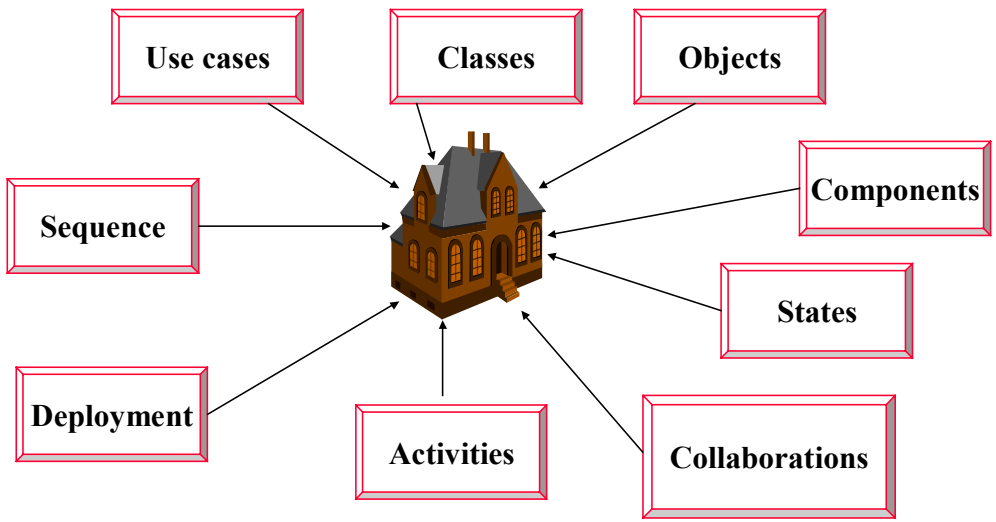


Visualization elements

Diagrams :

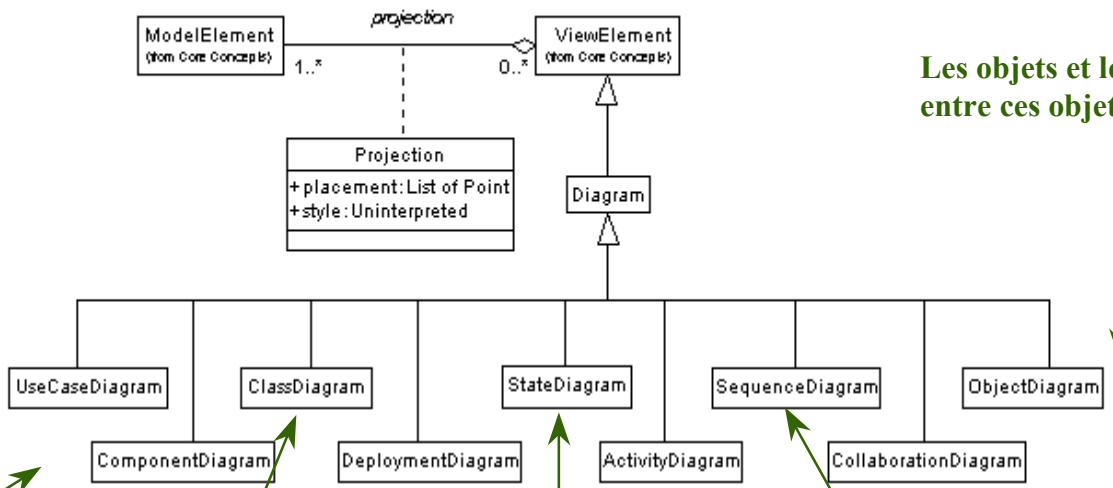
- UseCaseDiagram
- ComponentDiagram
- CollaborationDiagram
- ClassDiagram
- DeployementDiagram
- StateDiagram
- ActivityDiagram
- SequenceDiagram
- ObjectDiagram

The nine UML diagrams



Projections

A view element is a textual and graphical projection of a collection of model elements. The UML predefines a number of such graphical projections as common diagrams.



Fonctions du système du point de vue de l'utilisateur.

Les objets et les relations de base entre ces objets.

Composants physiques d'une application.

Schémas de l'installation des composants sur les dispositifs matériels.

Représentation du comportement en termes d'états.

Représentation du comportement des opérations en termes d'actions.

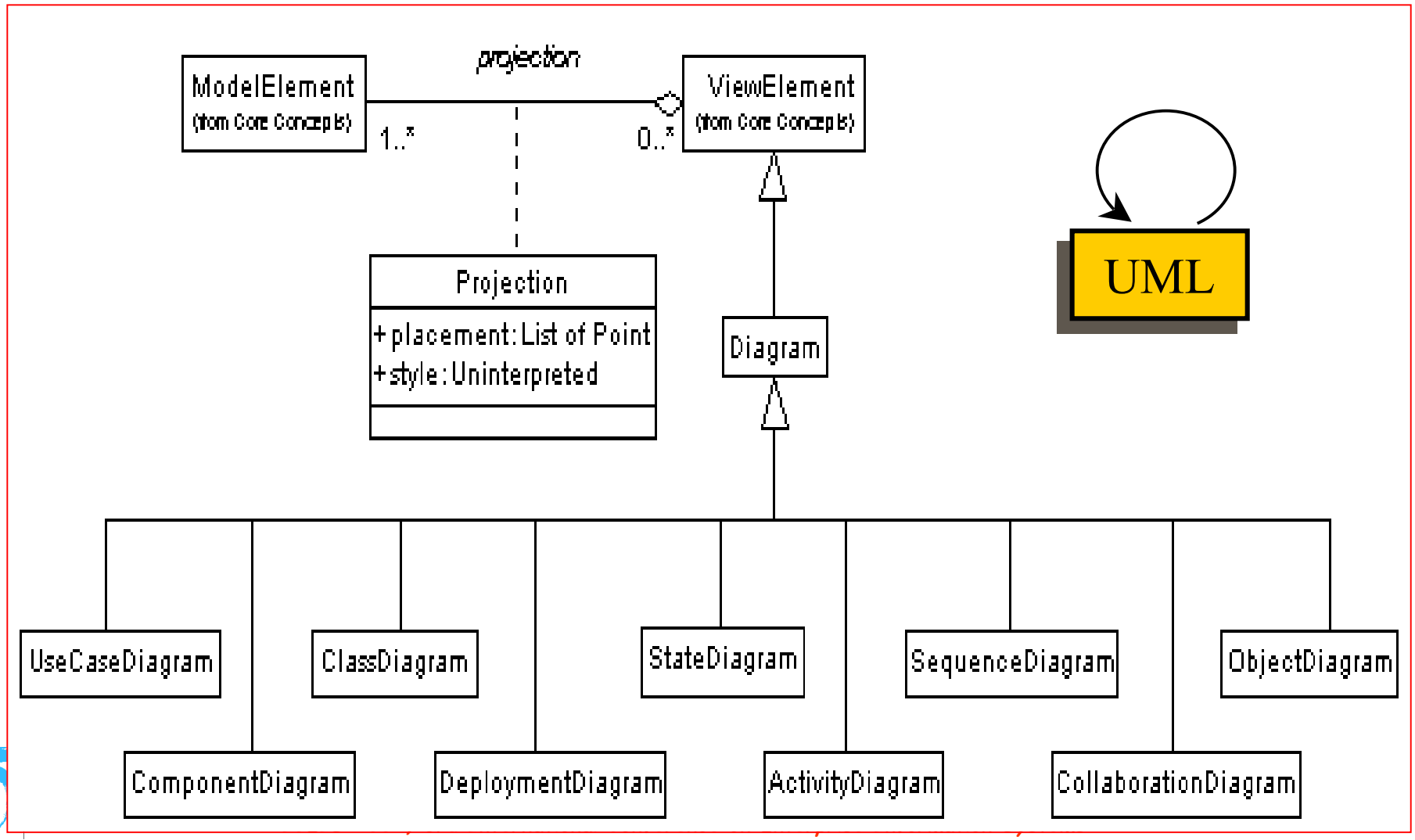
Représentation des objets, des liens mutuels et des interactions potentielles.

Représentation des objets et de leurs interactions temporelles.

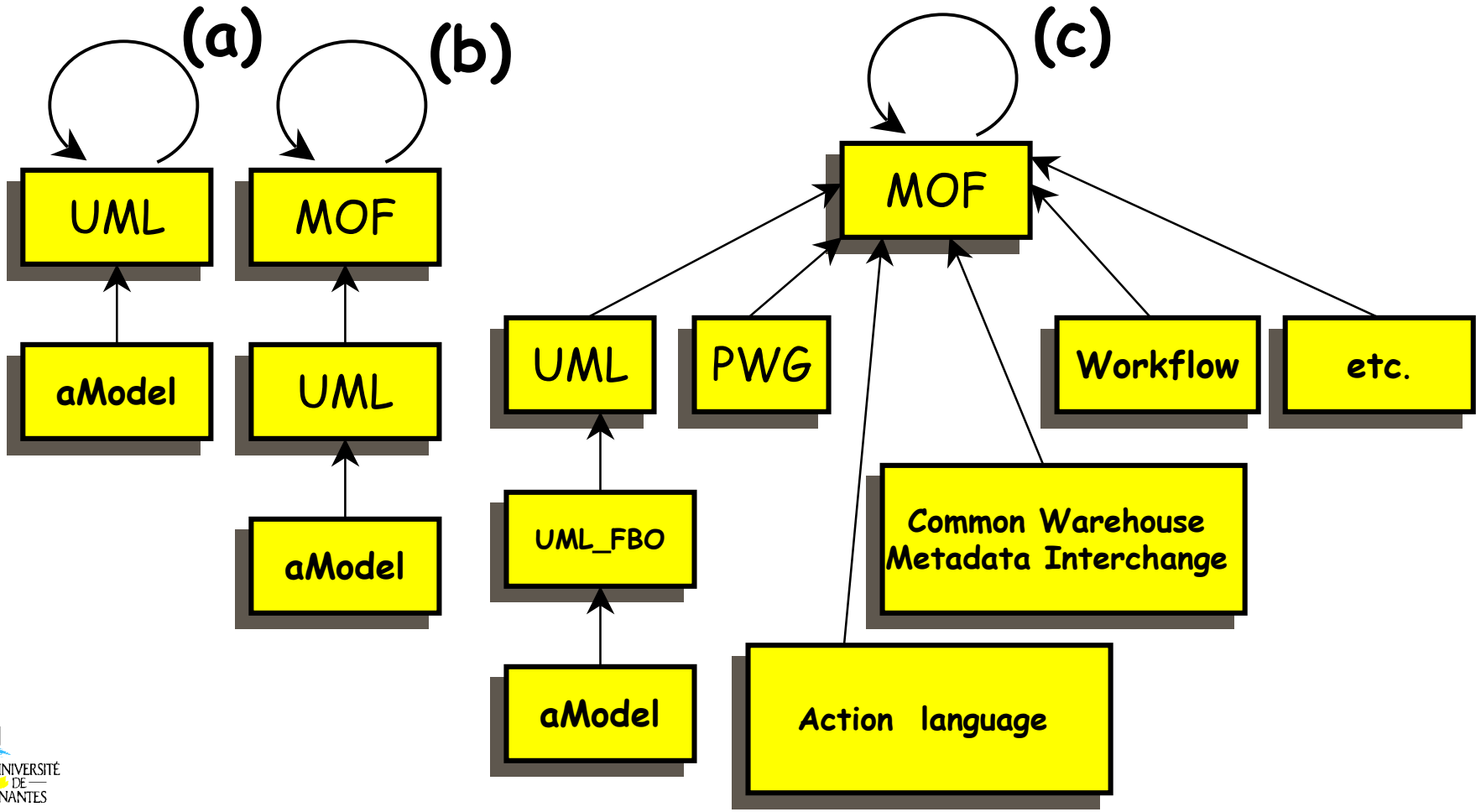
Structure statique des classes et des relations entre ces classes.

Fragments of a UML meta-model

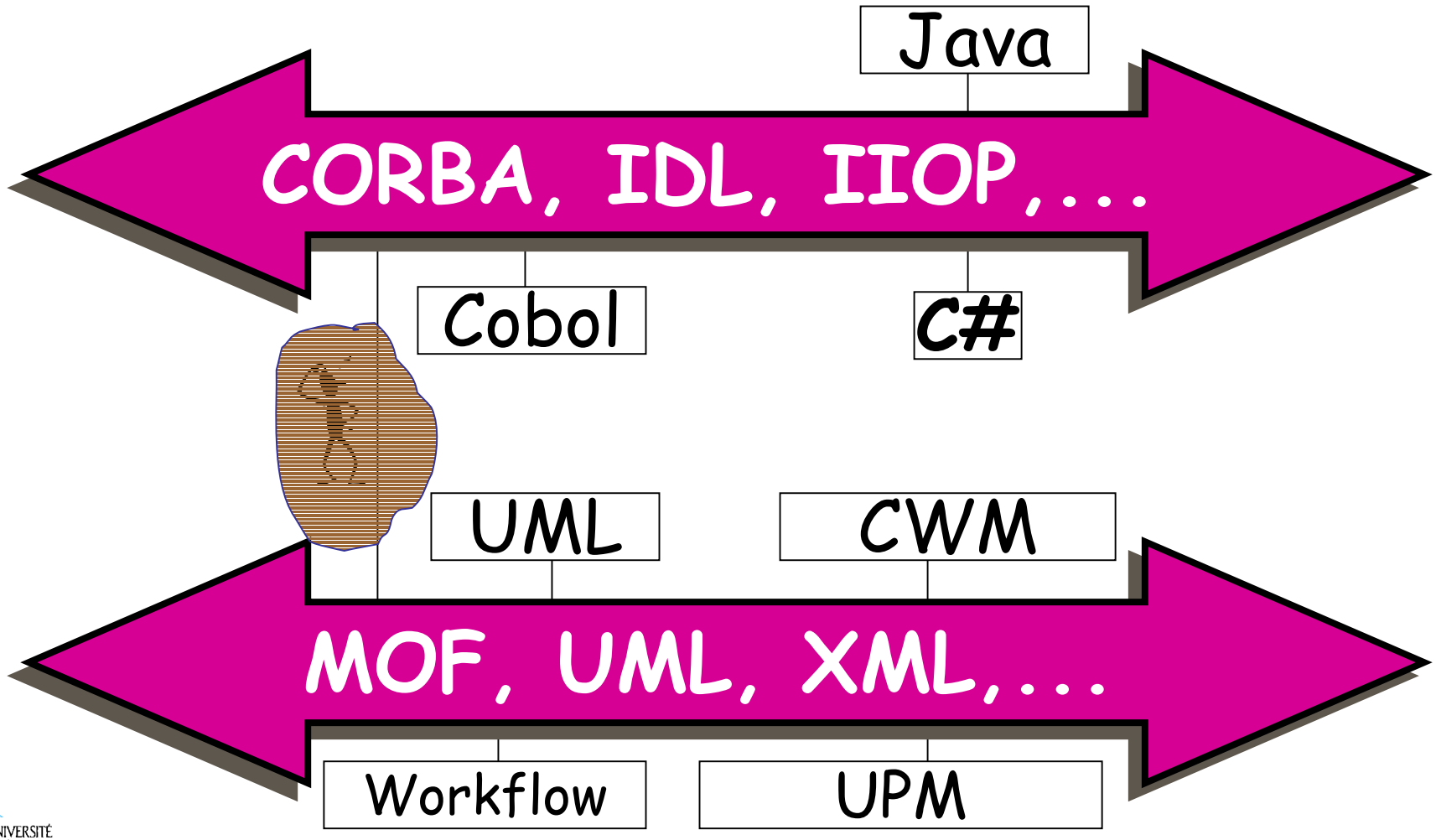
not 1.3



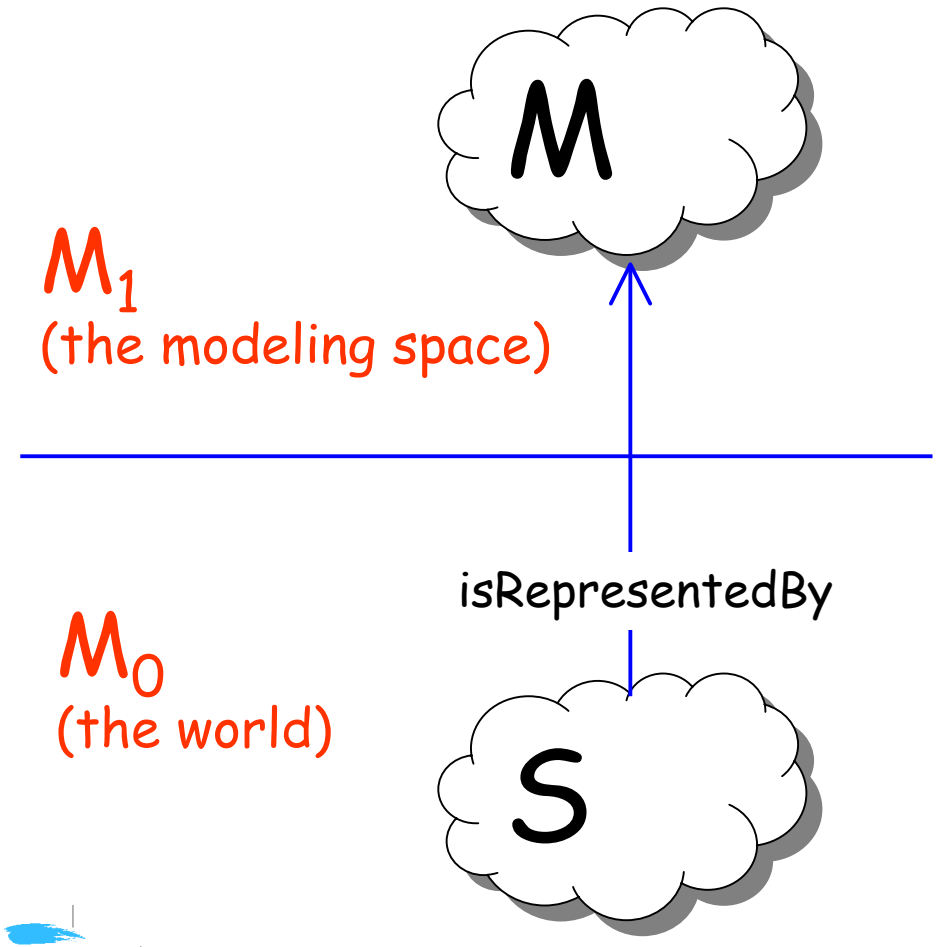
Three stages in the evolution of modeling techniques at OMG.



OMG : the software bus and the knowledge bus.

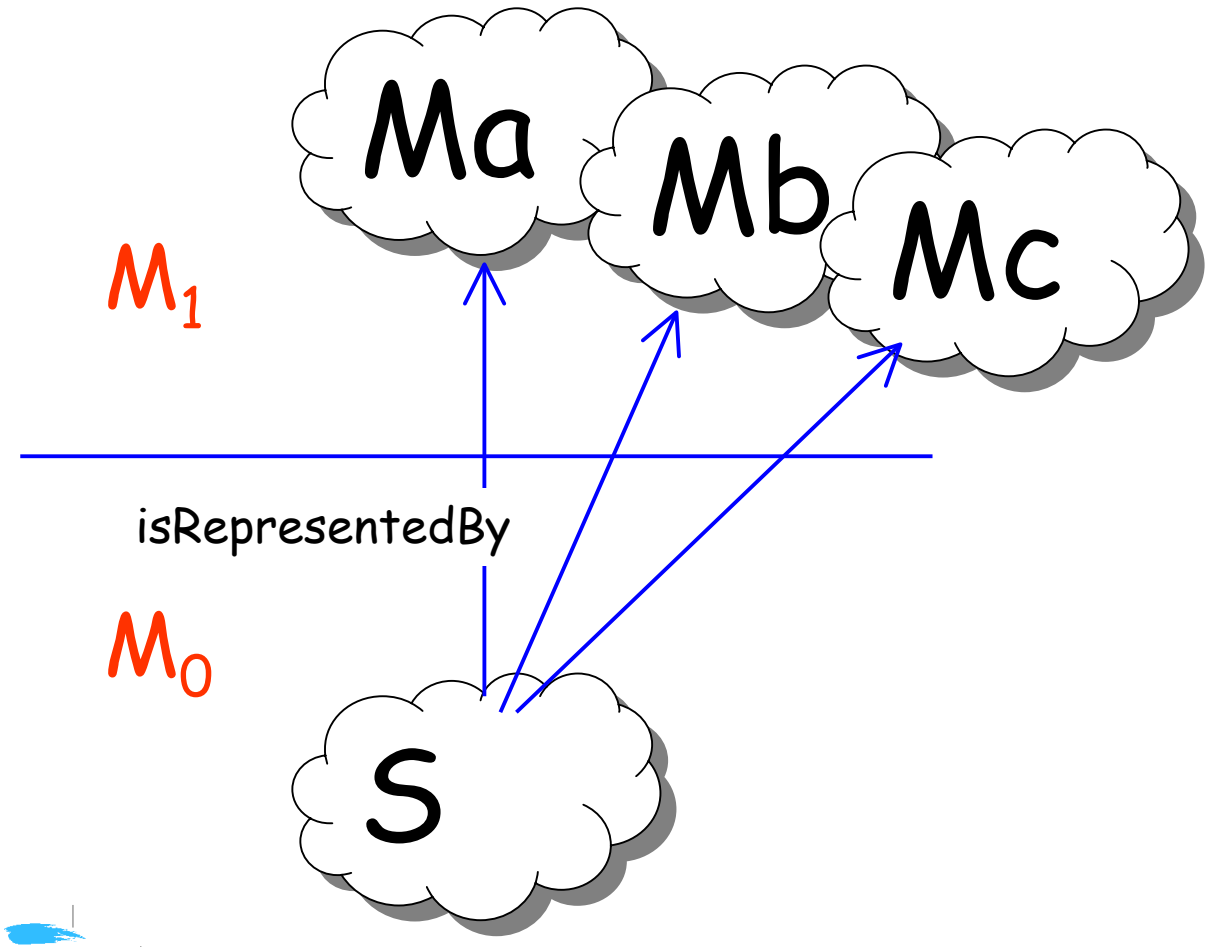


Systems and models



A model M is a simplified representation of the world, as a matter of fact of only a part S of the world called the system.

Aspect-Oriented Modeling

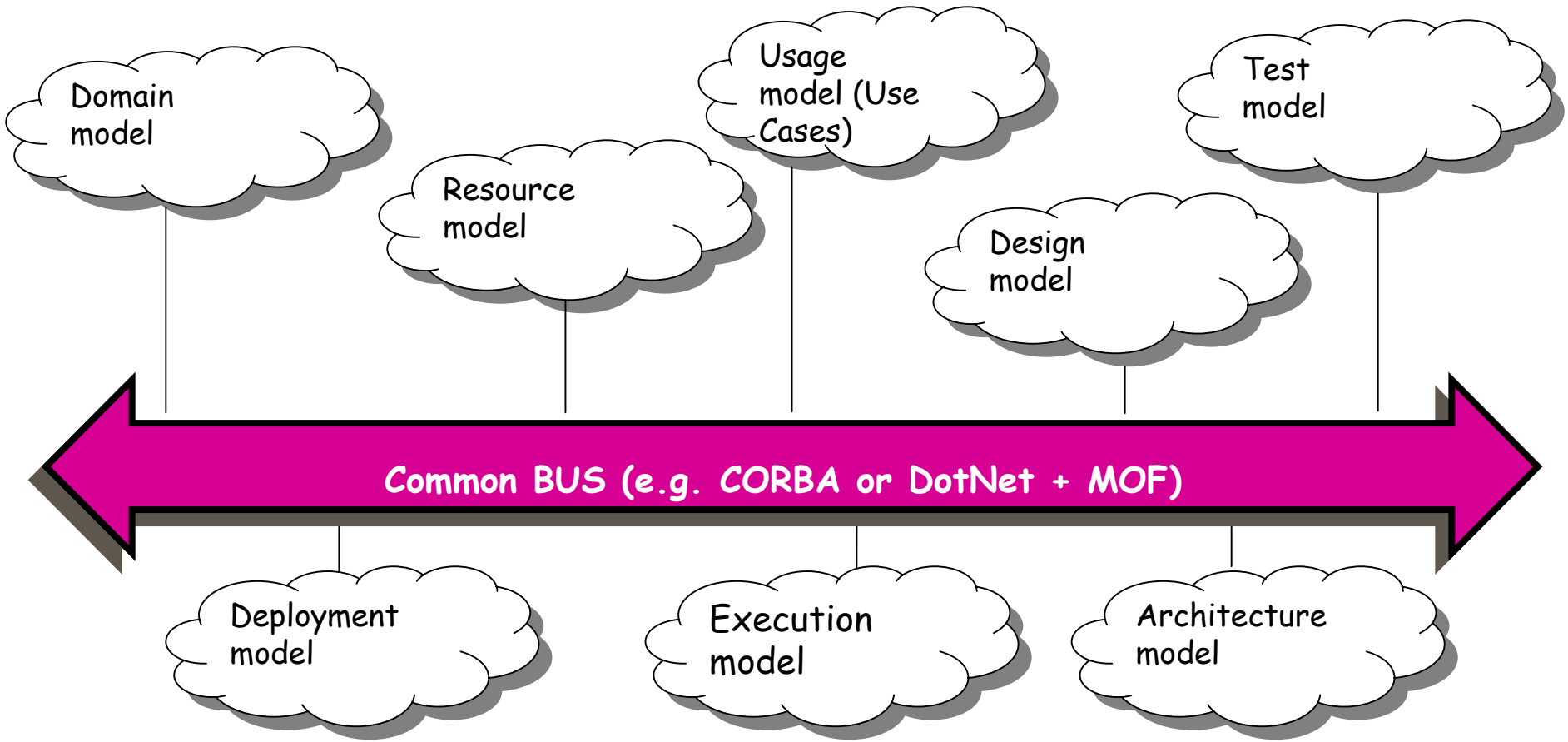


Obviously a given system may have plenty of different models.

Each model represents a given aspect of the system.

AOM (Aspect-Oriented Modeling) is a pleonasm.

The on-line separate models organization (aspect oriented software engineering)



First-class independent models and meta-models

Modeling

Modeling, in its broadest sense, is **the cost-effective use of something in place of something else for some cognitive purpose**. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality.

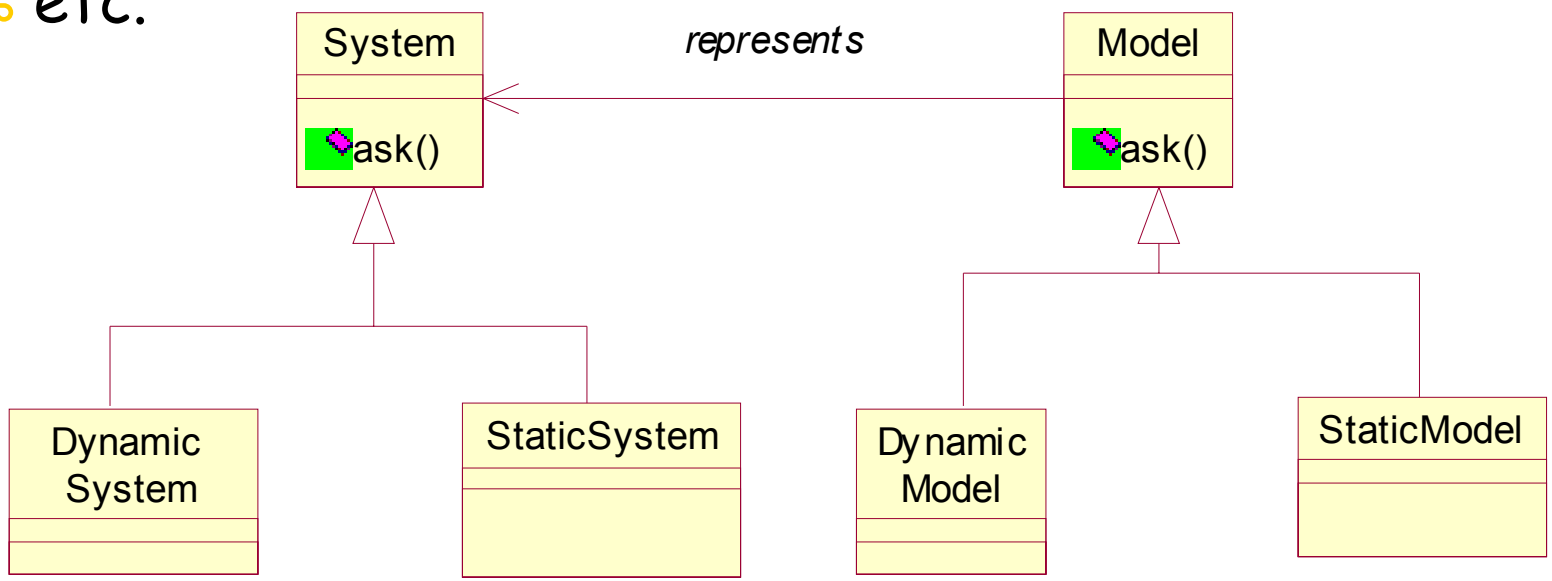
Jeff Rothenberg

Limited Substituability Principle

- The purpose of a model is always to be able to answer some questions in place of the system, exactly in the same way the system itself would have answered similar questions

Various kinds of models

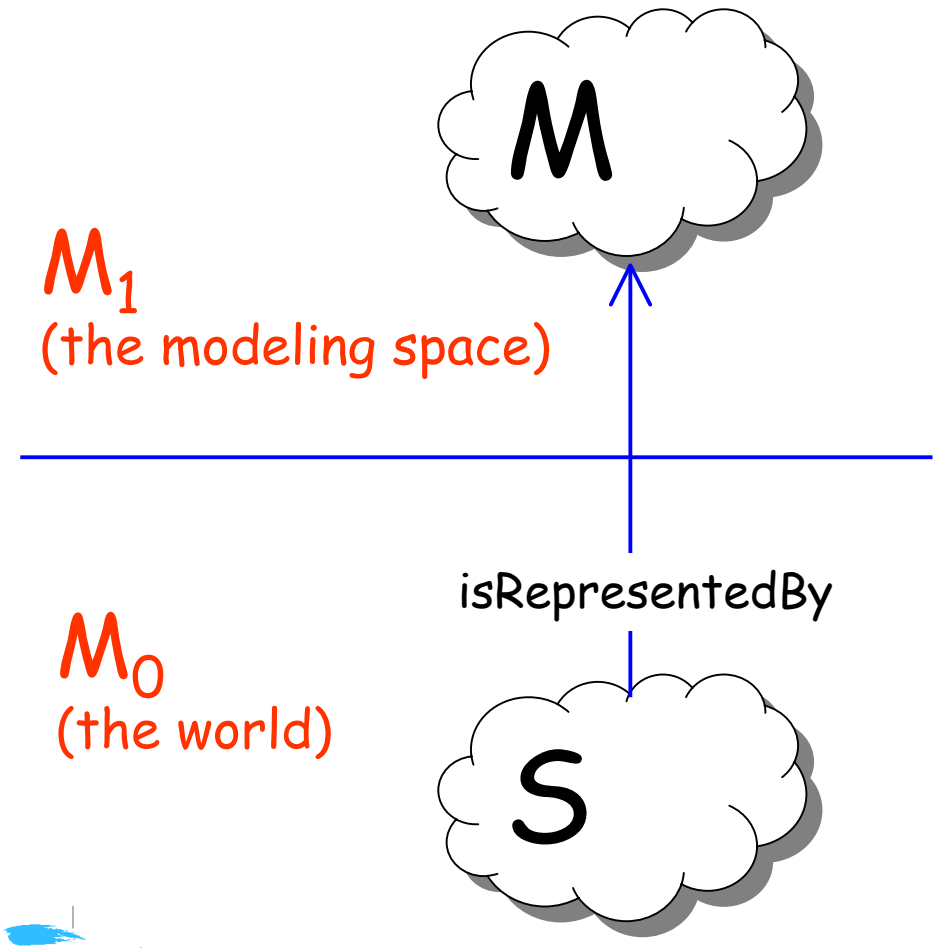
- Products and processes
- Legacy and components
- Static and dynamic
- etc.



Theory?

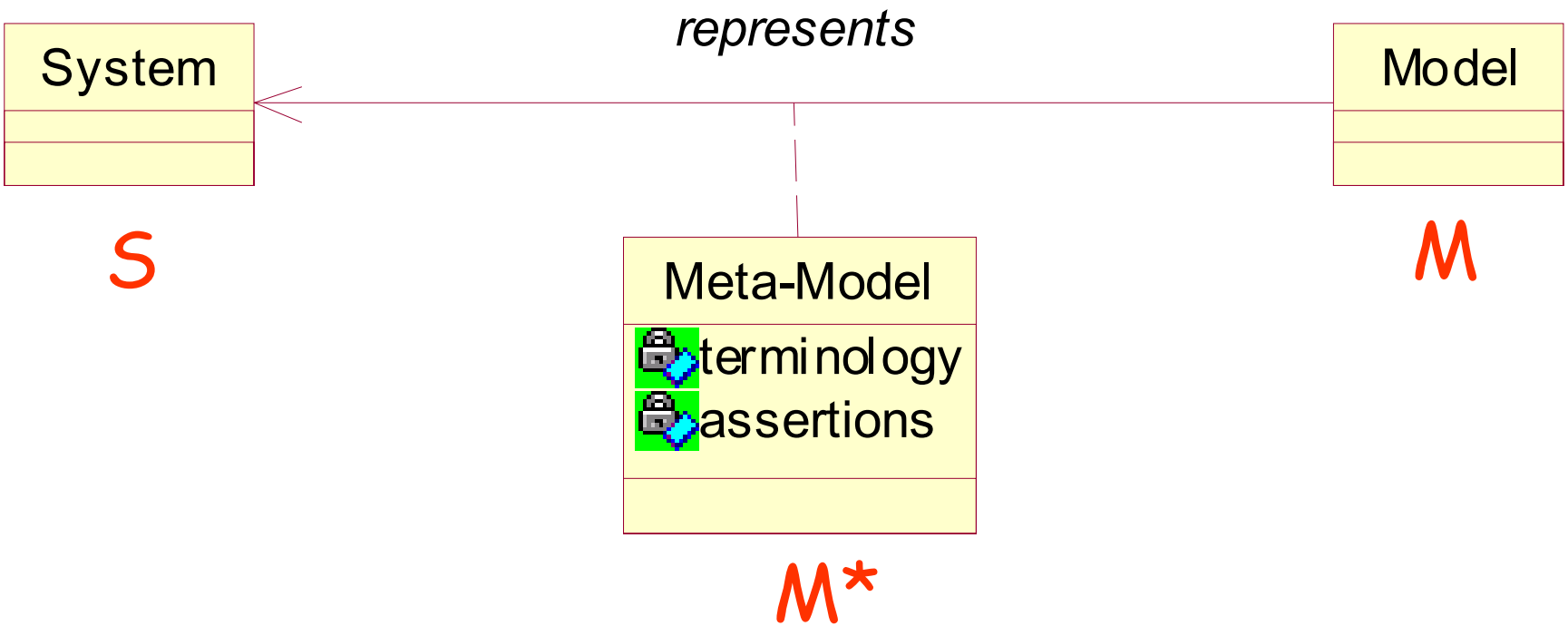
- What are the theoretical tools that could be useful in model engineering?
- What help can they provide with the MDA effort?
- Main answer: Ontologies (Gruber, Guarino, etc.)
- Concrete translation: the four-levels OMG modeling stack

Systems and models



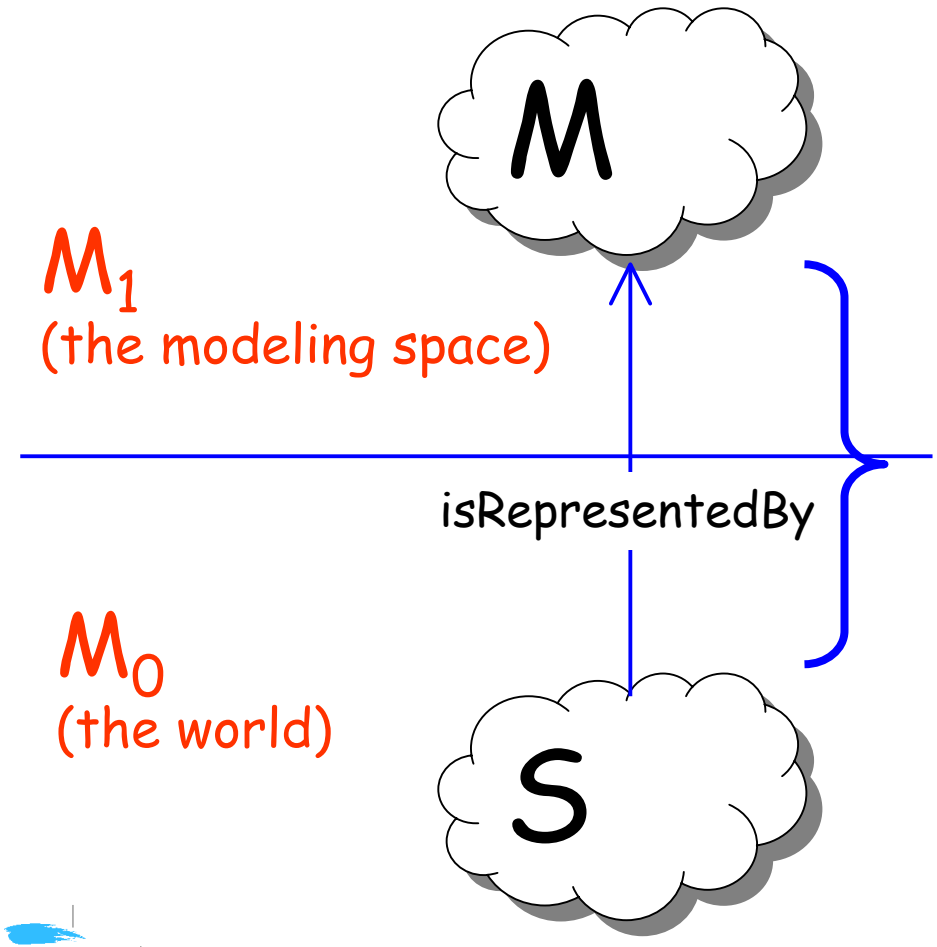
- What is exactly this relation?
- Can we specify this "aspect selection" with precision?
- How to combine several such relations? How to characterize them?

Meta-Model



The correspondance between a model and a system is defined by a meta-model.

Ontologies and meta-models



- For the theorician, this is an ontology
- For the IS practitioner, this is a meta-model
- A meta-model is a simple form of an ontology

Ontology: definition

"A body of formally represented knowledge is based on a **conceptualization**: the **objects**, **concepts**, and other **entities** that are presumed to exist in some **area of interest** and the **relationships** that holds them.
A conceptualization is an **abstract**, **simplified view** of the world that we wish to represent **for some purpose**.
An ontology is an **explicit specification of a conceptualization**. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. For knowledge-based systems, what "exists" is exactly that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, we can define the ontology of a program by defining a set of representational terms. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g. classes, relations, functions or other objects) with human-readable text describing what the names are meant to denote ..."

Gruber, T.G.
A Translation Approach to Portable Ontology Specifications
Knowledge Acquisition, V.5, N.2, (1993)

System, Model and Ontology

a System S

an Ontology O

a Model M

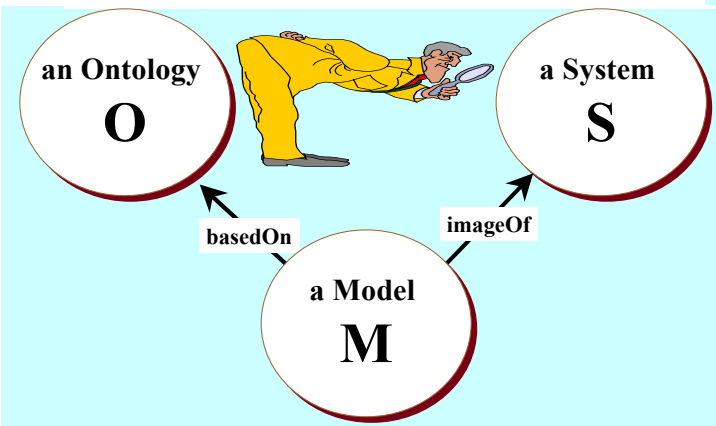
imageOf

sem

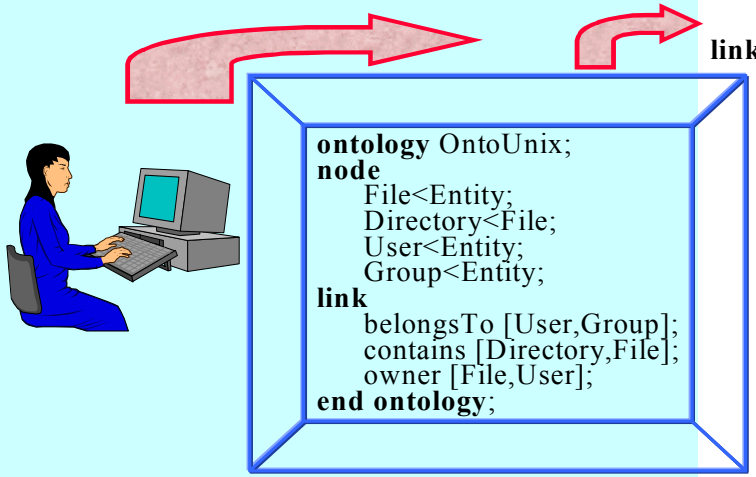
A model is the result of the observation of a system with respect to an ontology.

Meta-models and ontologies

- ✓ Ontologies bring:
  - ☒ Abstraction
  - ☒ Consensus and sharing



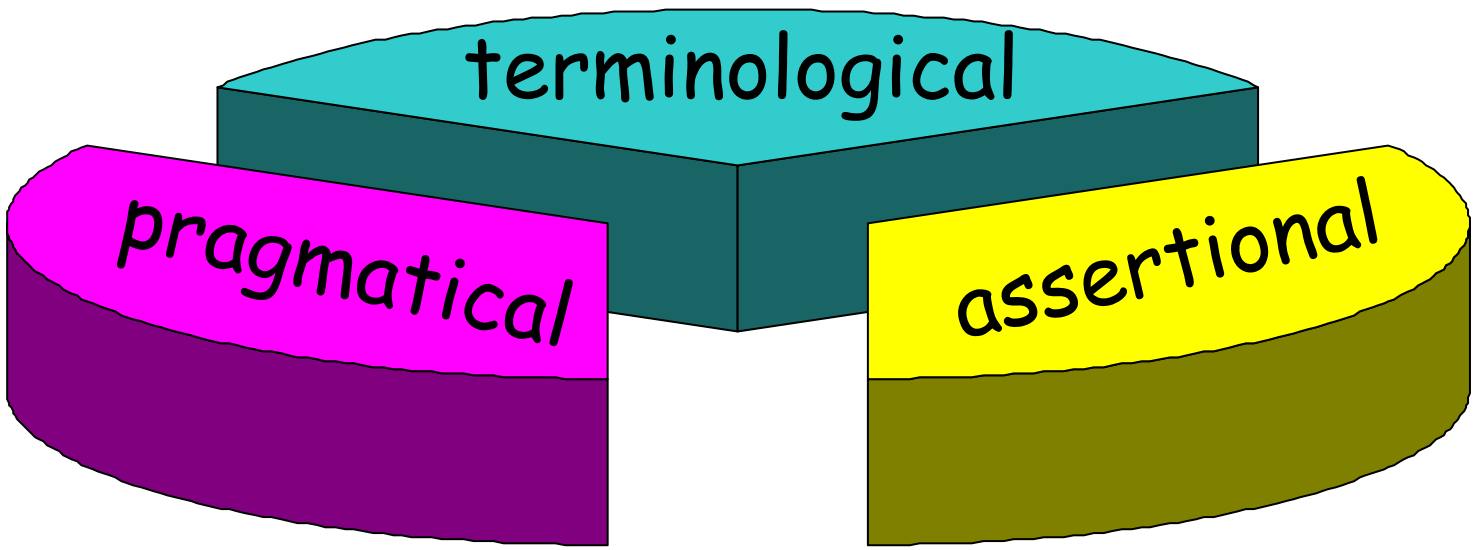
System, ontology, model:




```
model UnixModel;
ontology OntoUnix;
node
    teachers, students: Group;
    pierre,paul,jacqques: User;
    f1,f2,f3,f4: File;
    d1,d2,d3: Directory;
link
    belongsTo (pierre, teachers);
    belongsTo (paul, students);
    belongsTo (jacques, teachers);
    owner (f1, pierre);
    owner (f2,paul);
    owner (f3, paul);
    owner (f4, jacques);
    owner (d1, pierre);
    owner (d2, pierre);
    owner (d3, paul);
    contains (d1,f1);
    contains (d1,f2);
    contains (d1,d2);
    contains (d2,f3);
end model;
```

Layered ontologies

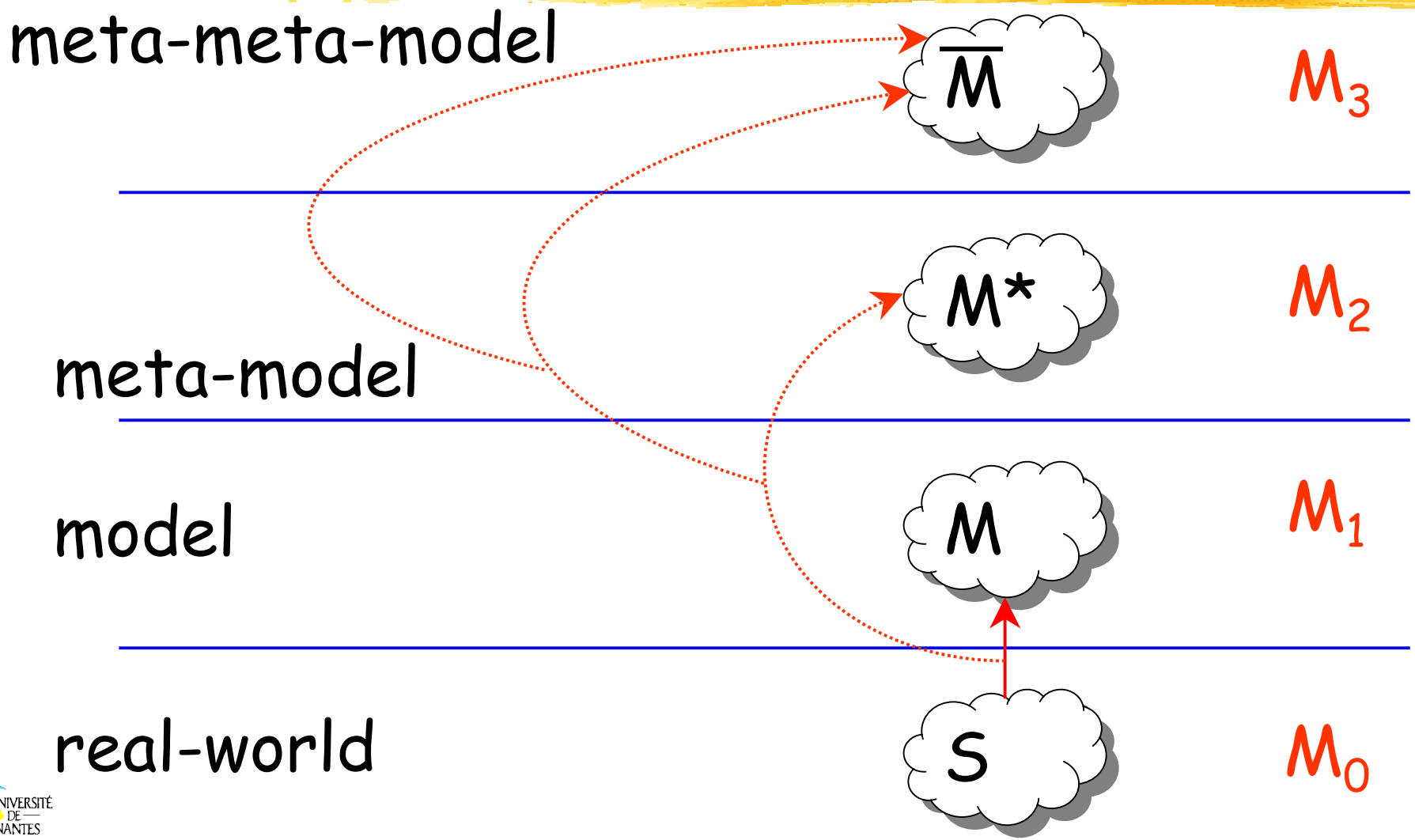
Concepts and Relations
e.g. UML diagrams



e.g. How to draw a class?
presentation issues, etc.

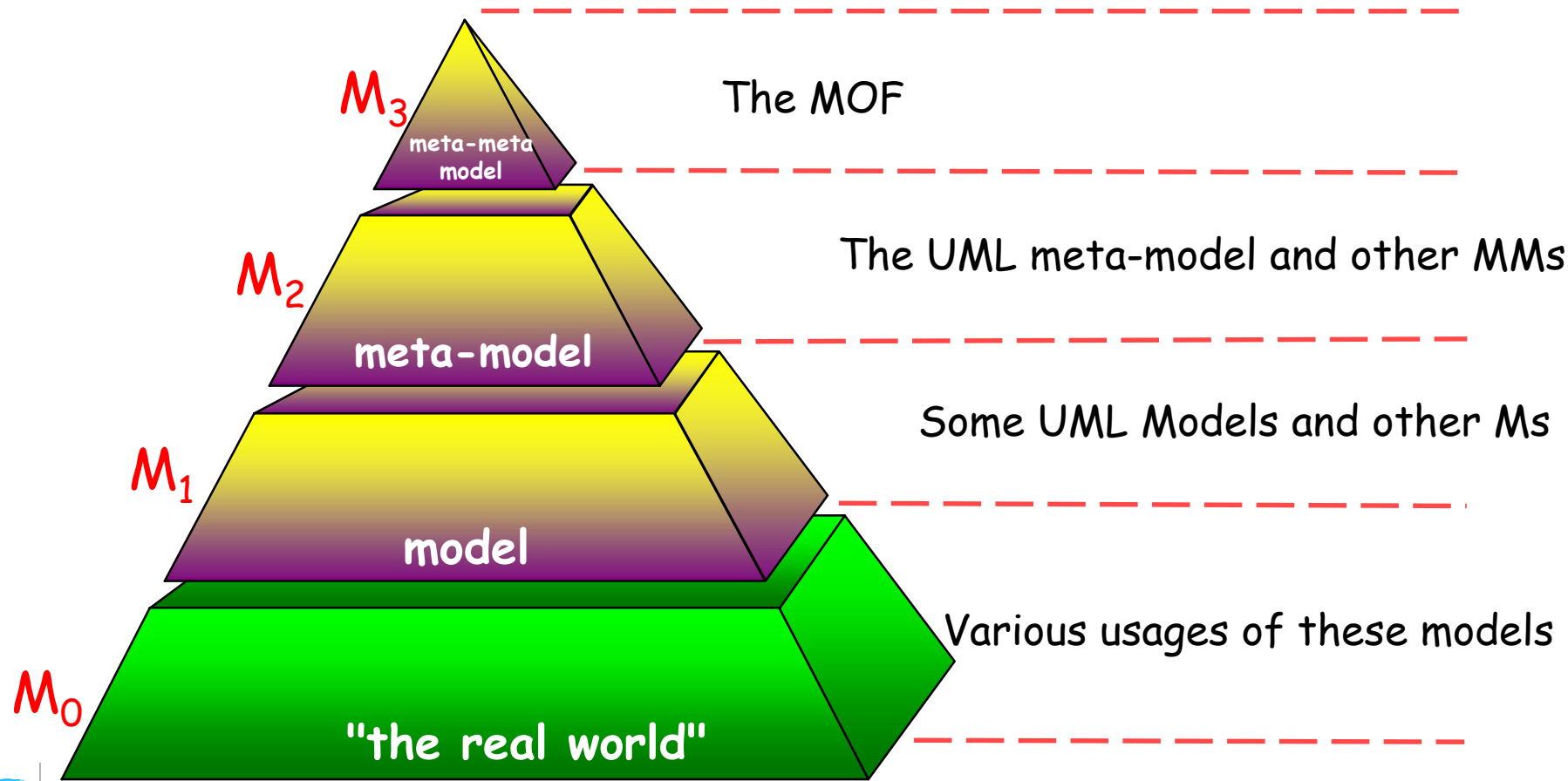
e.g. OCL statements

The global view

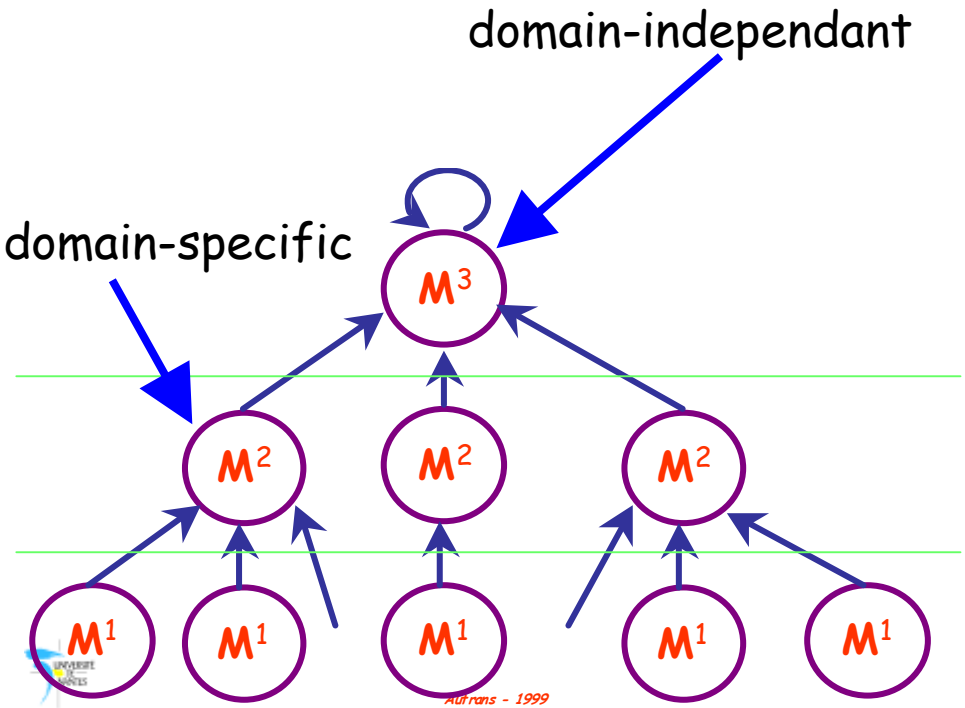


Egyptian architecture

[Inspired by IRDS, CDIF, etc.]

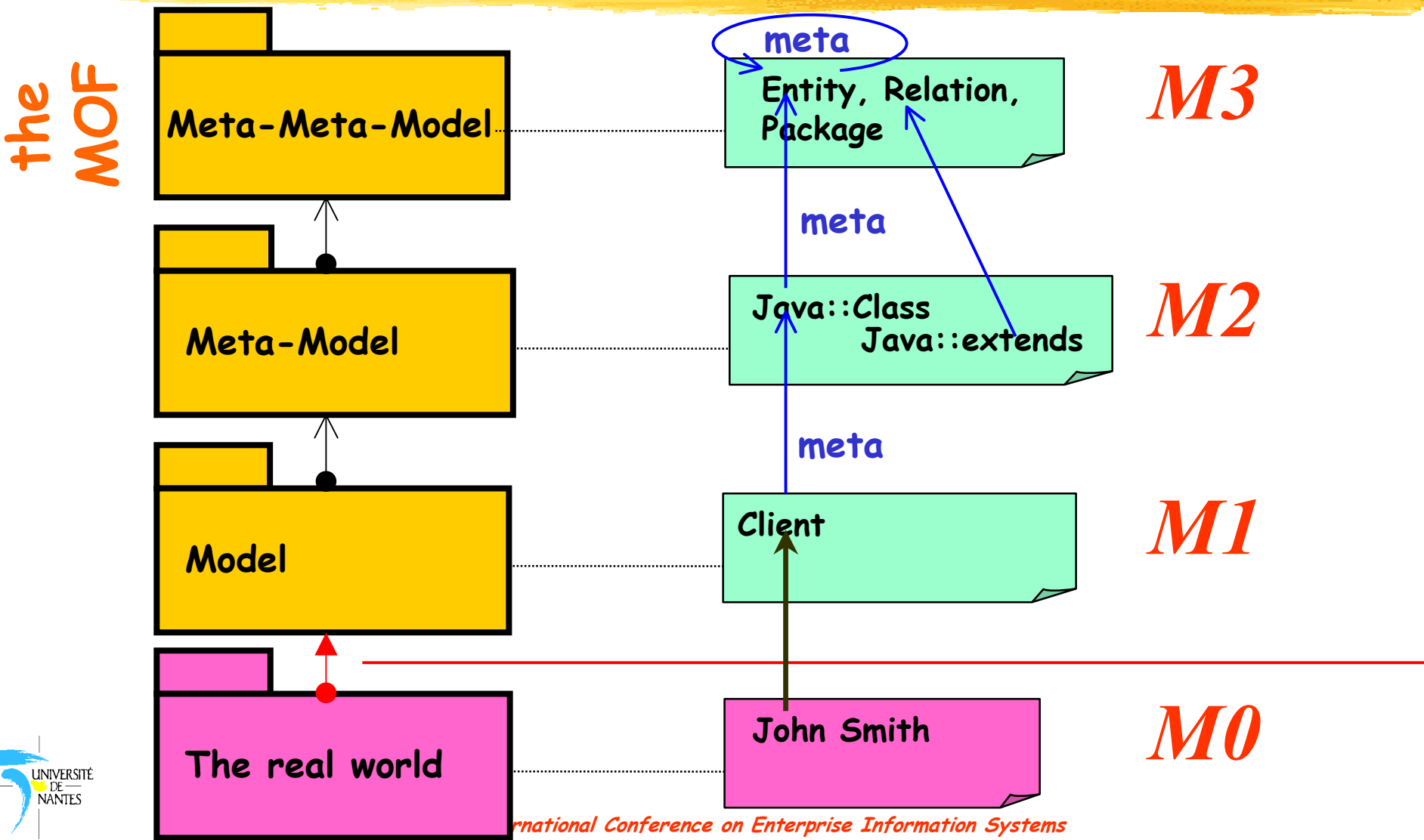


MOF : some definitions

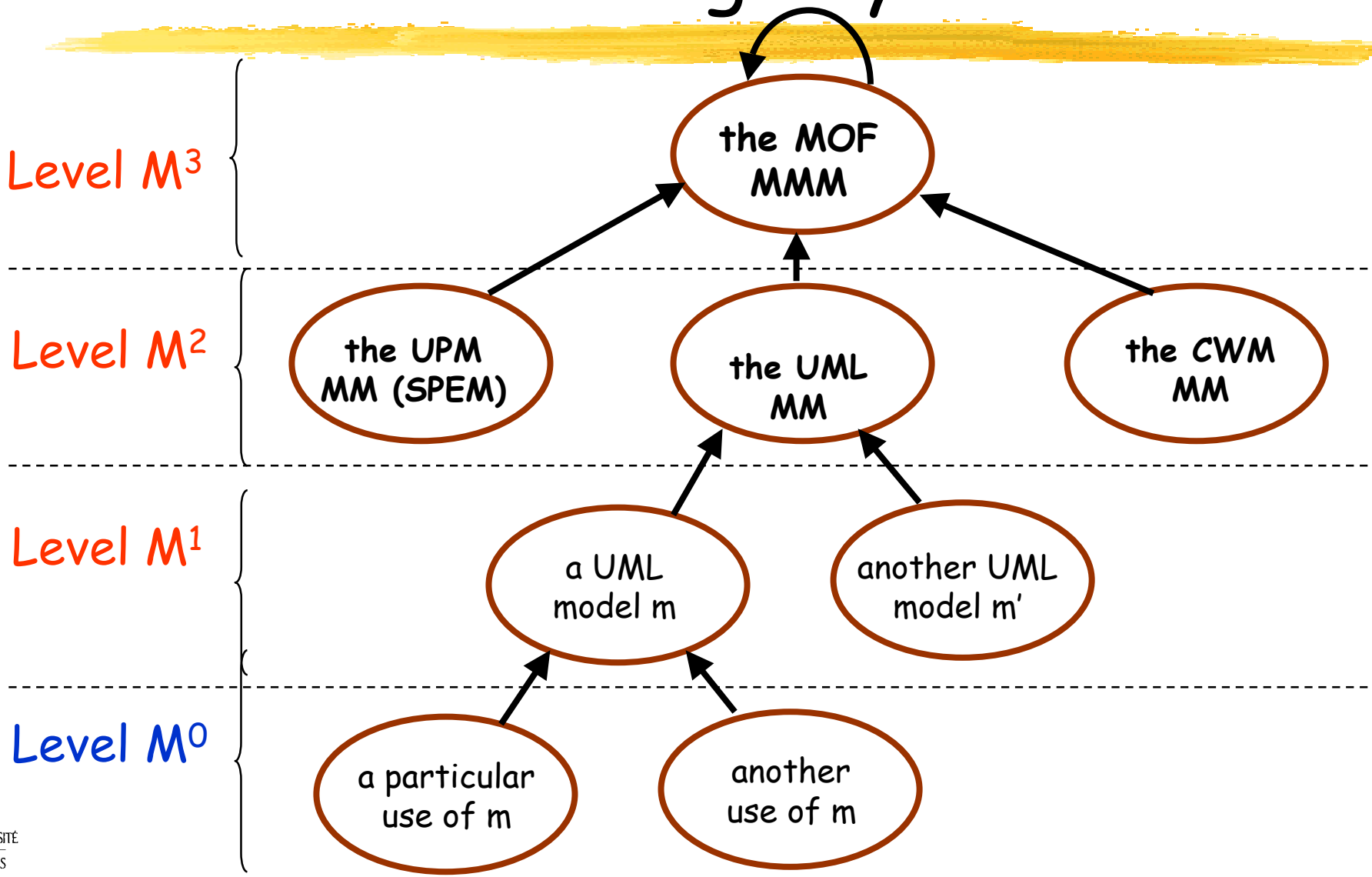


- The MOF is unique and self-defined.
- In principle, the MOF should be minimal.
- The MOF factorize all that is common to all meta-models, e.g.:
  - ✓ Generating towards a given middleware (.Net, Corba, Java, etc)
  - ✓ Transporting models and meta-models
  - ✓ Persisten support for models and meta-models (repository)

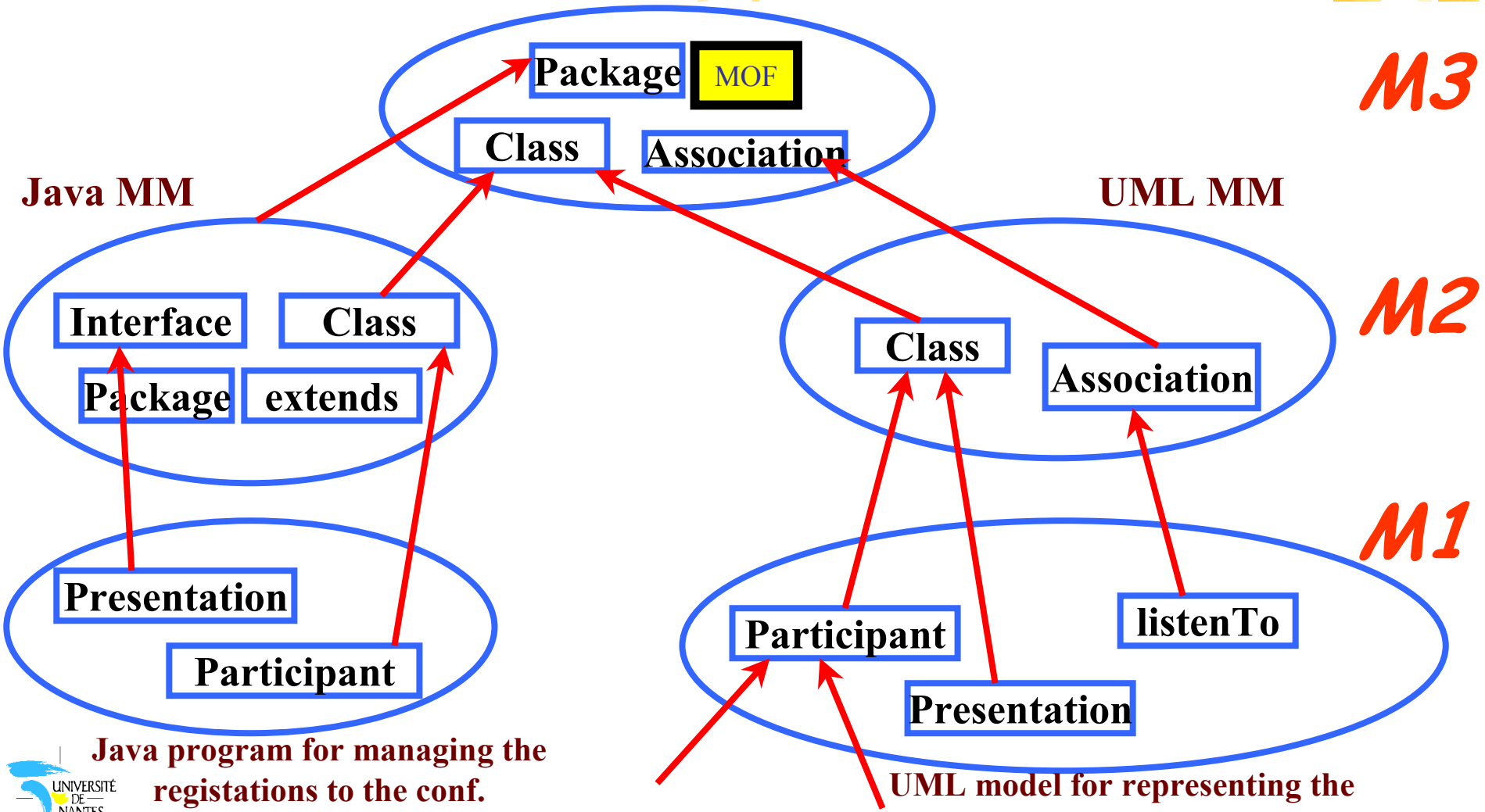
The four-level architecture.



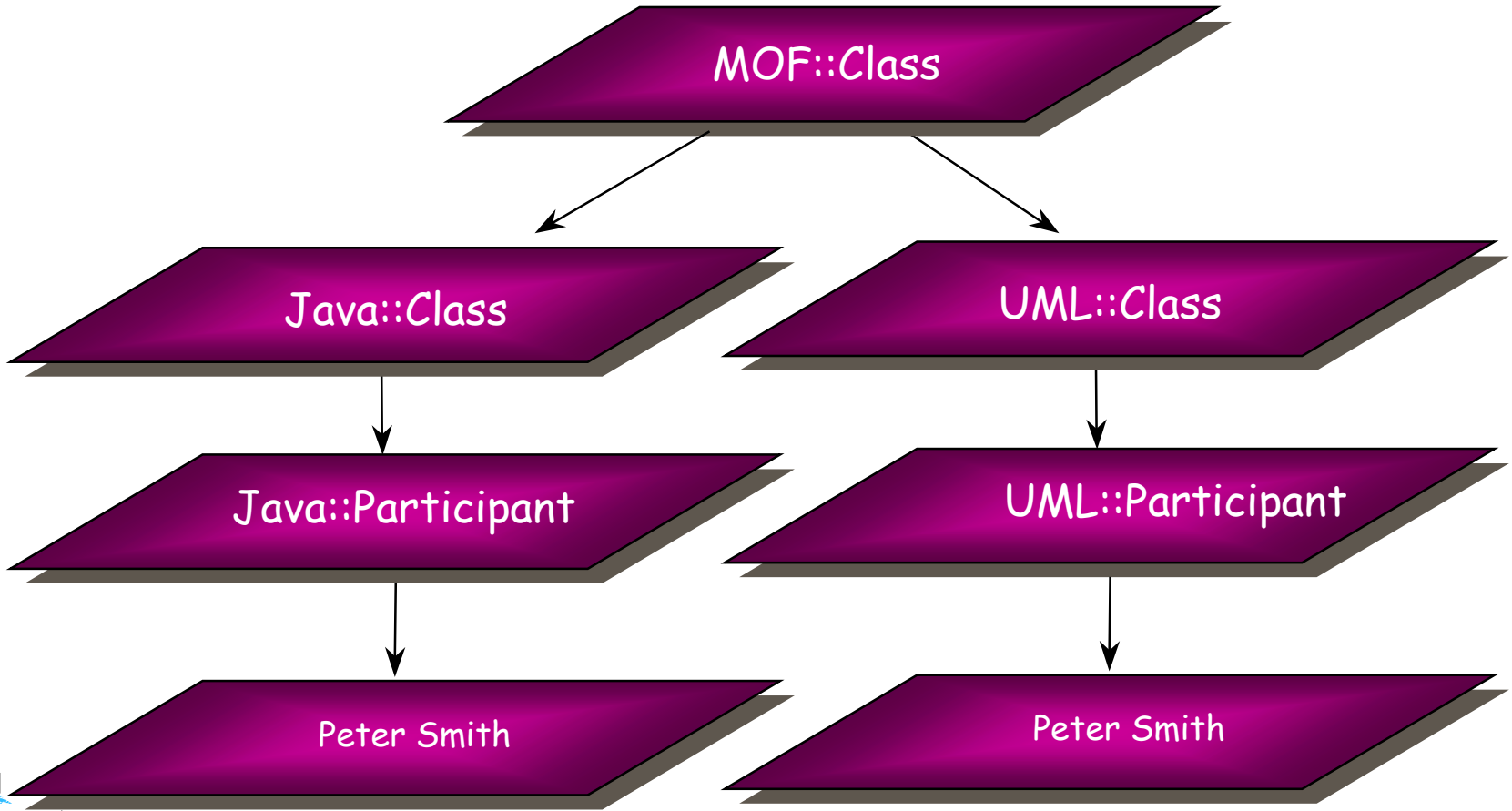
The Three Modeling Layers



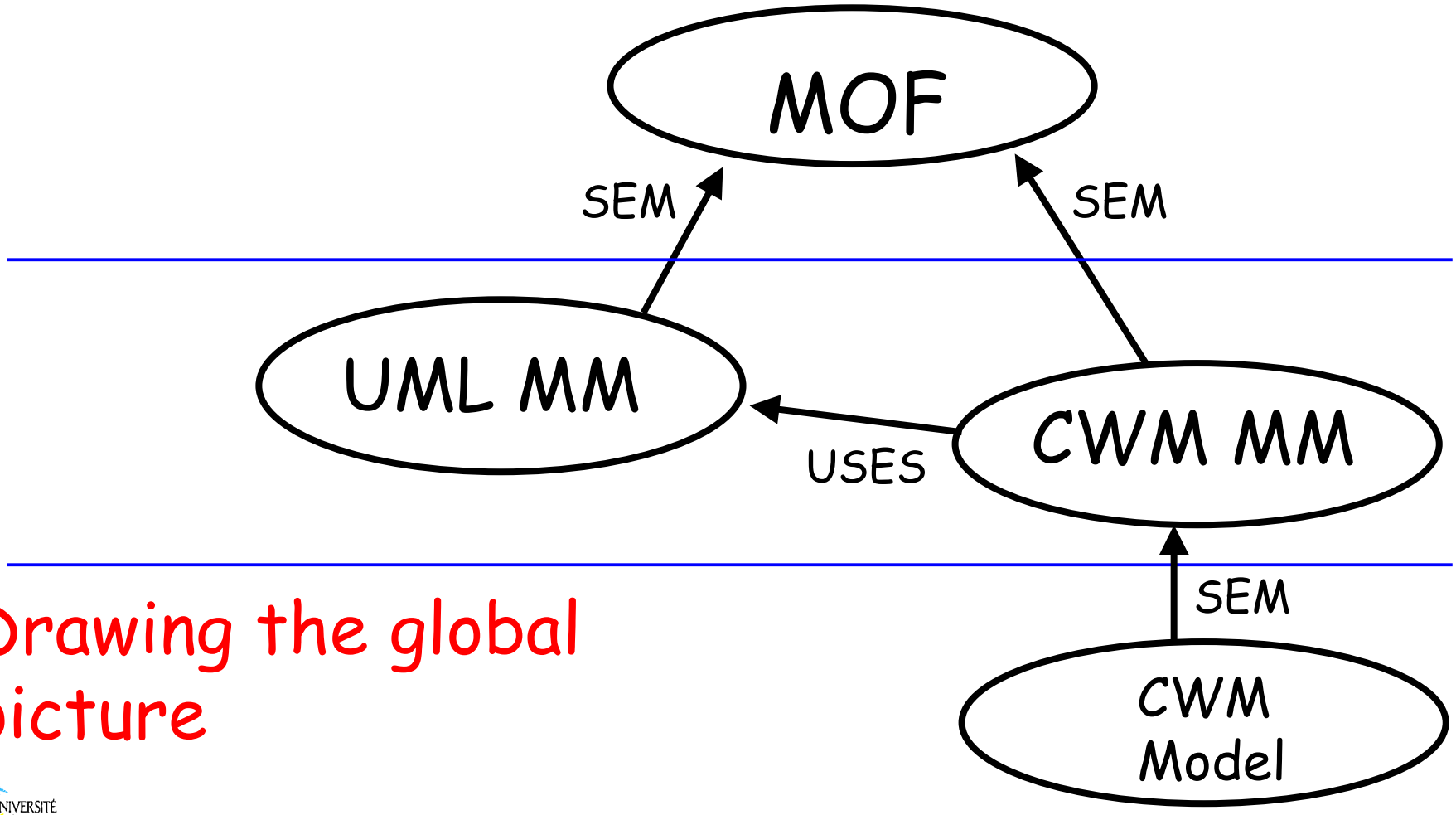
Illustrating the OMG modeling architecture.



Multiple meta-models



Direct relations between MMs



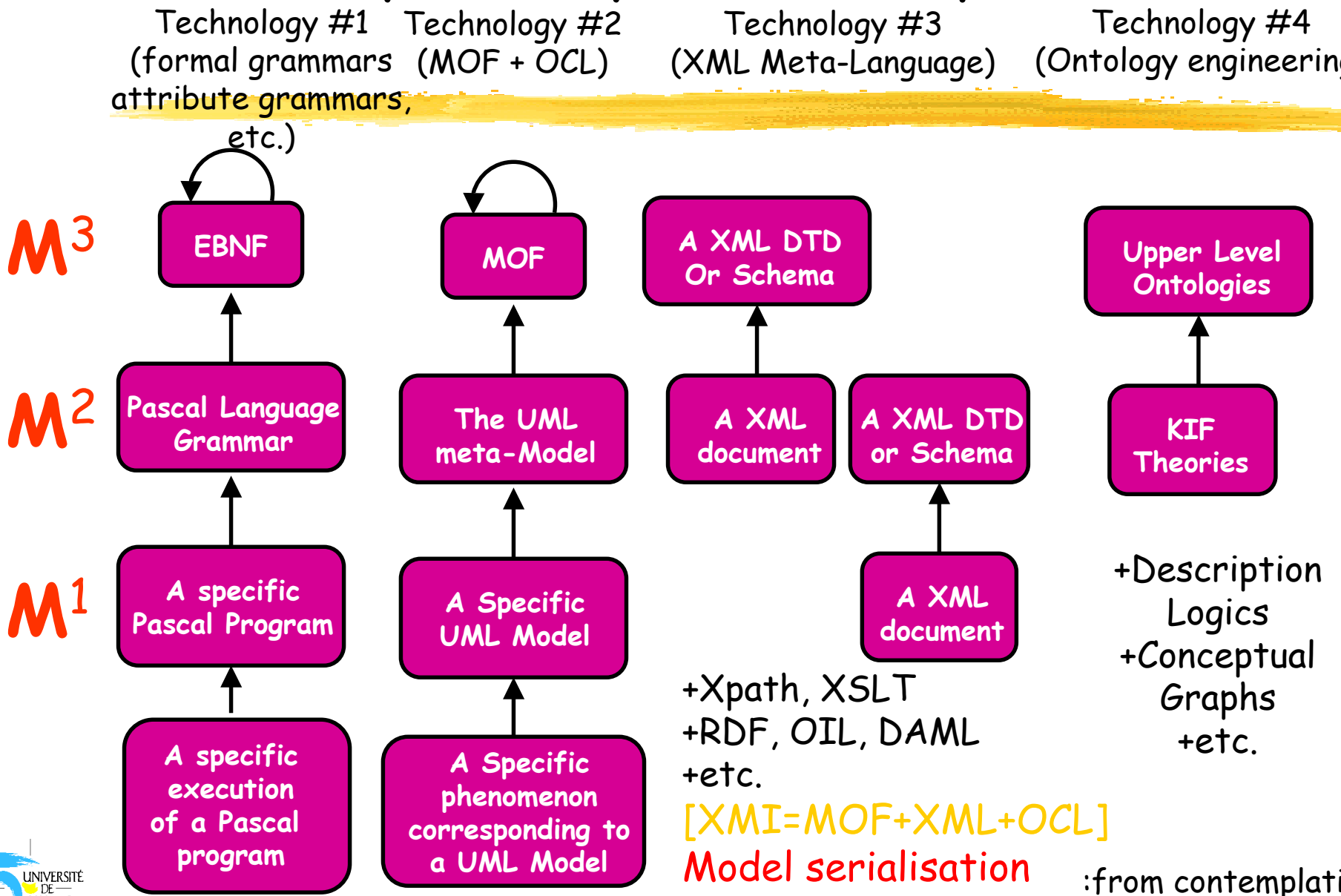
Drawing the global picture

UML profiles

- A UML profile is a grouping construct for UML model elements that have been customized for a specific domain or purpose using extension mechanisms such as stereotypes, tagged values and constraints. For example, the UML Profile for CORBA RFP customizes UML for specifying CORBA IDL.

- A meta-model defines a domain-specific language.A profile is a variant of a meta-model. It allows to define a dialect of a given language. There are a dozen of UML profiles that are currently being defined.

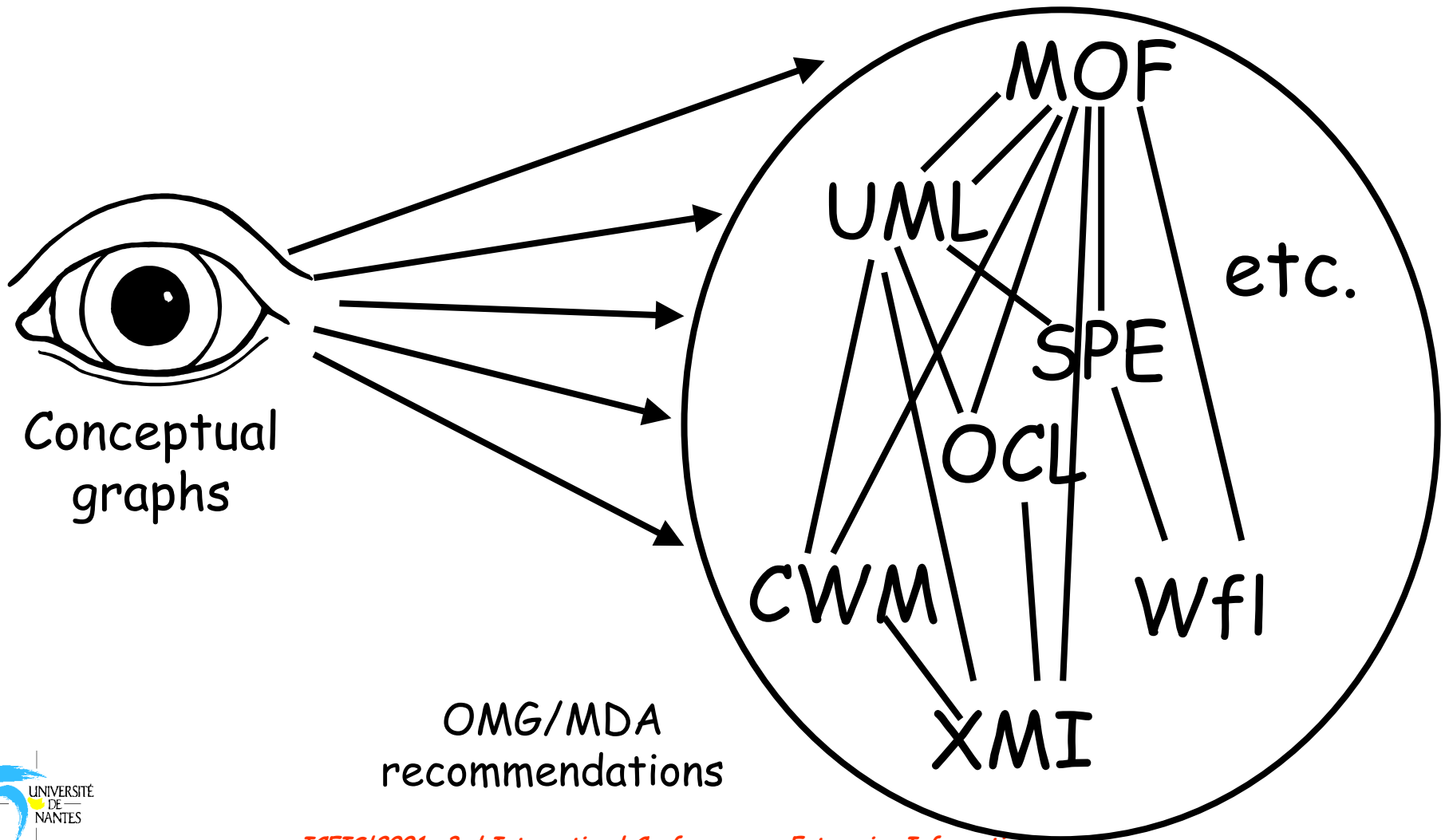
Abstract Syntax Systems Compared



Precision

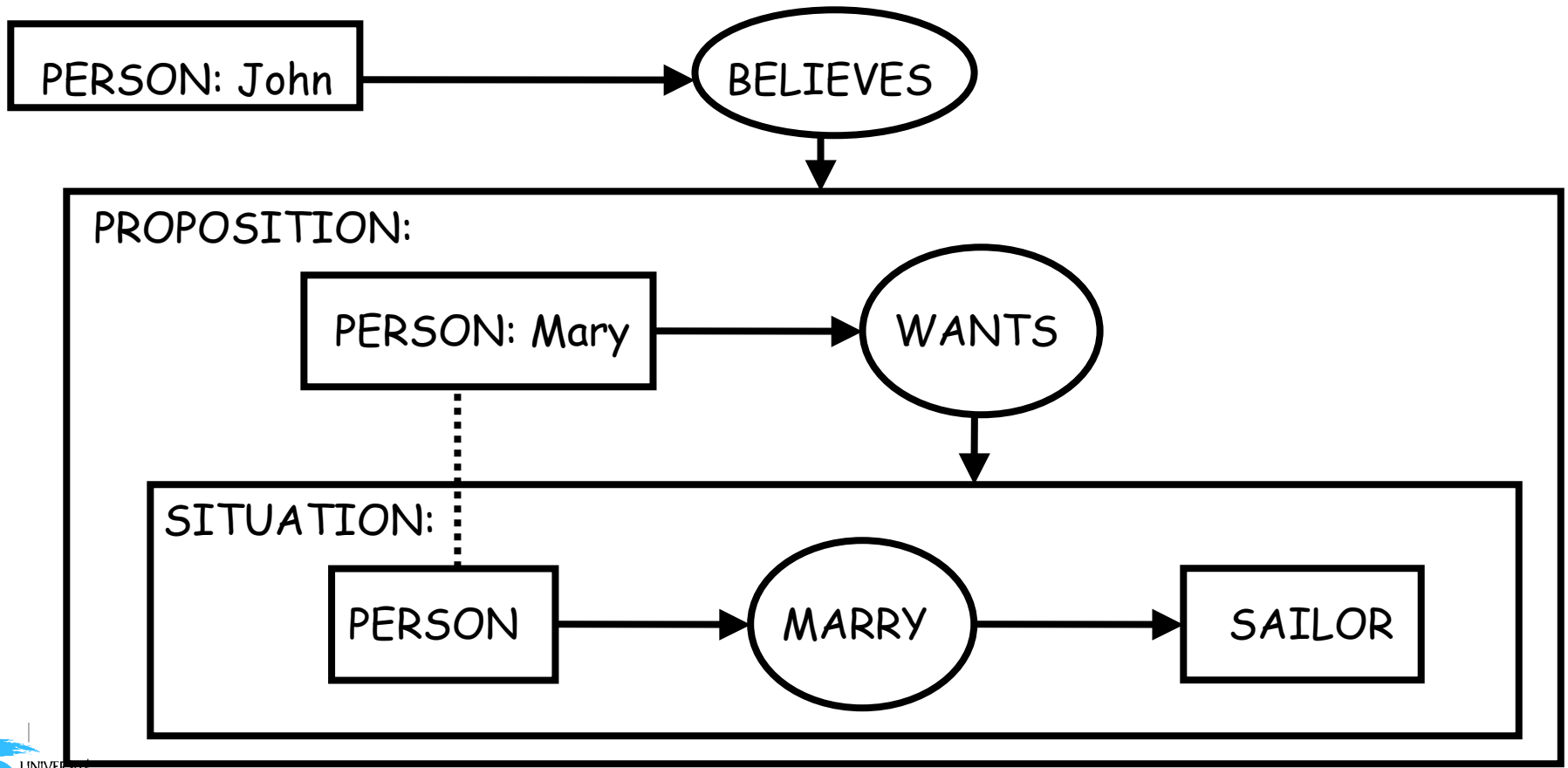
- In order to start building the industrial tools that will populate the MDA, we need a very precise definition of what a model or a meta-model is.
- Explicit, precise and operational definitions are prerequisite for a successfull deployment of the technology.

The global view

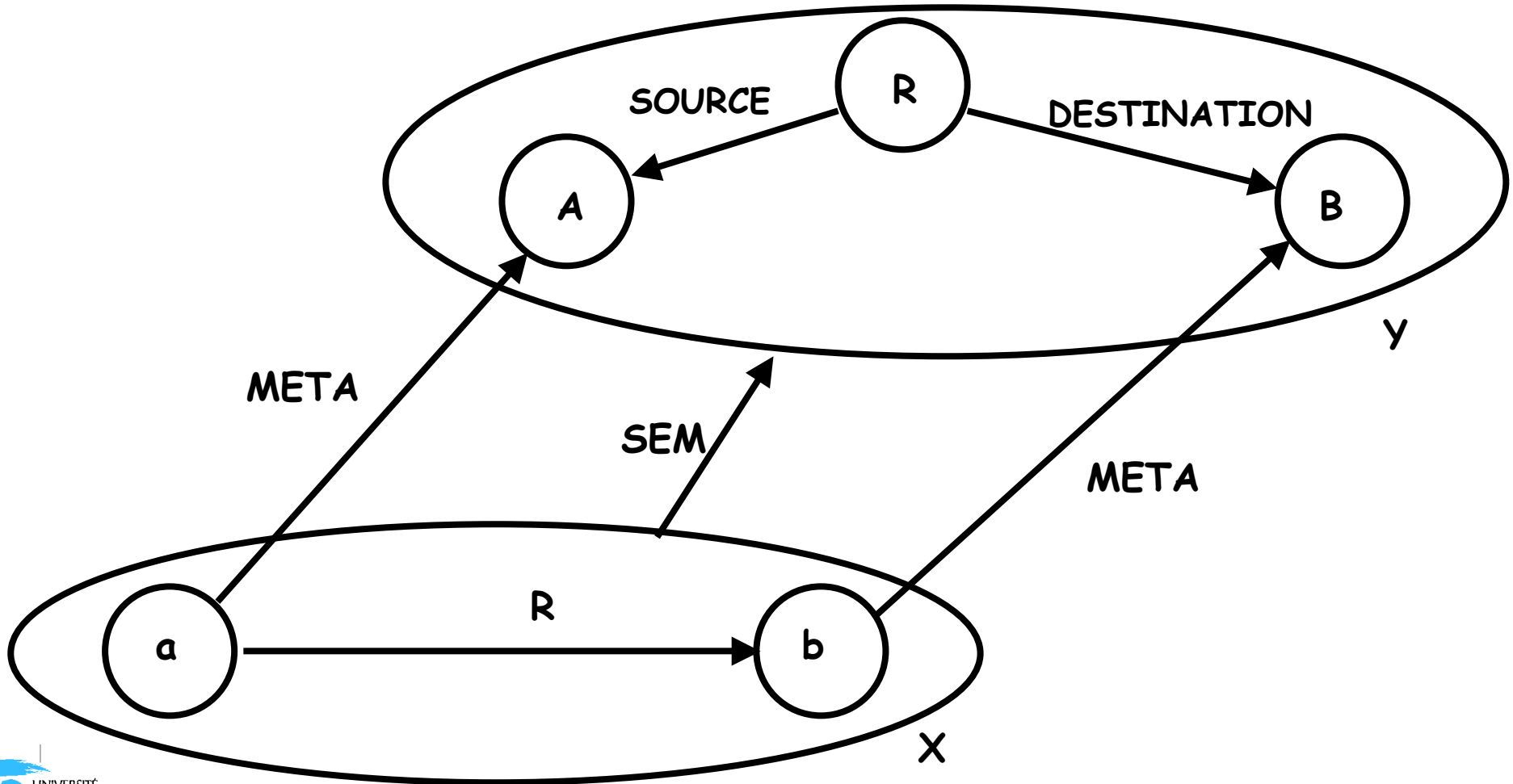


Contexts and Coreferences

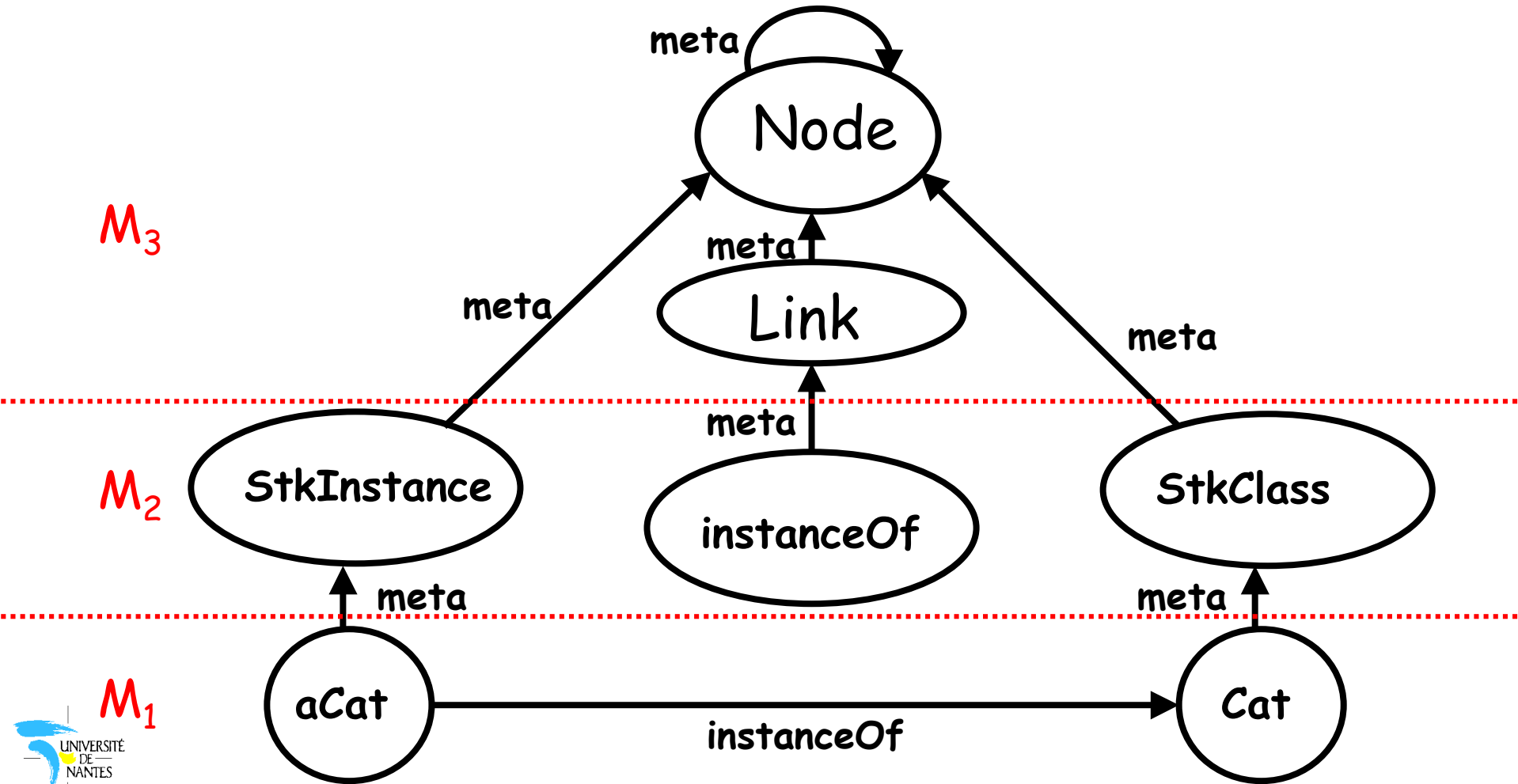
"John believes that Mary wants to marry a sailor"



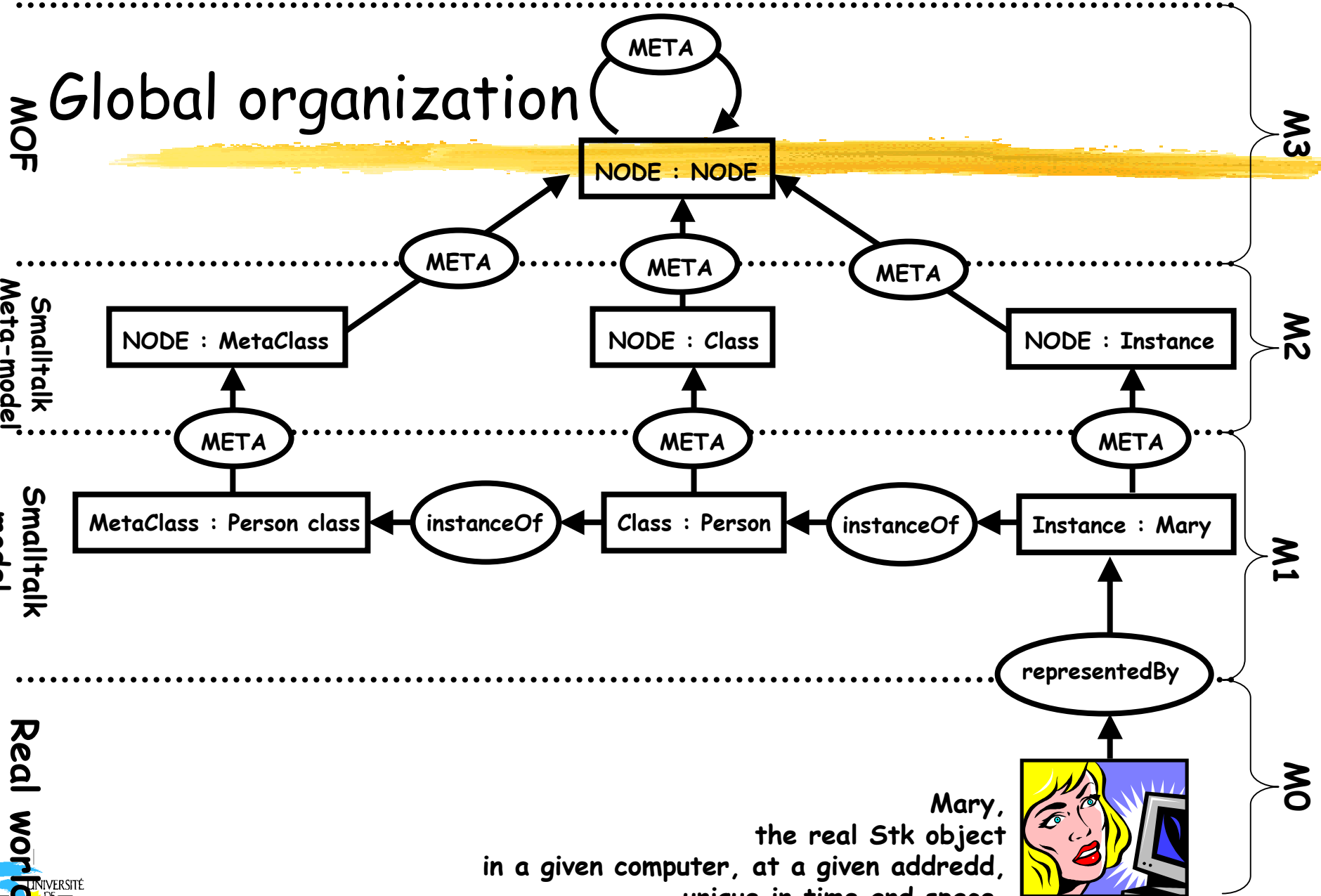
The SEM and META relations



Local and global definitions



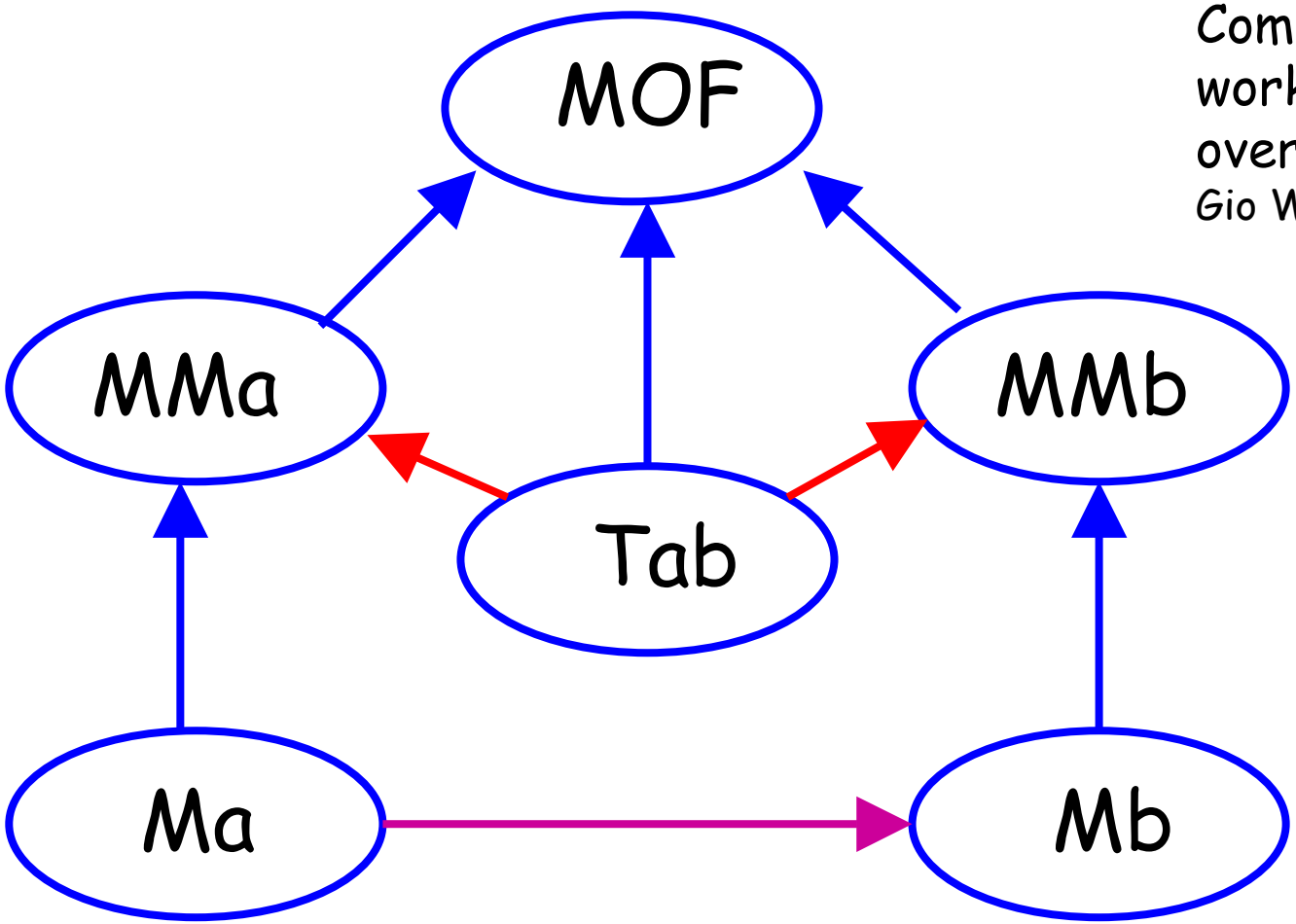
Global organization



Some Hopes and Dangers of the MDA (examples)

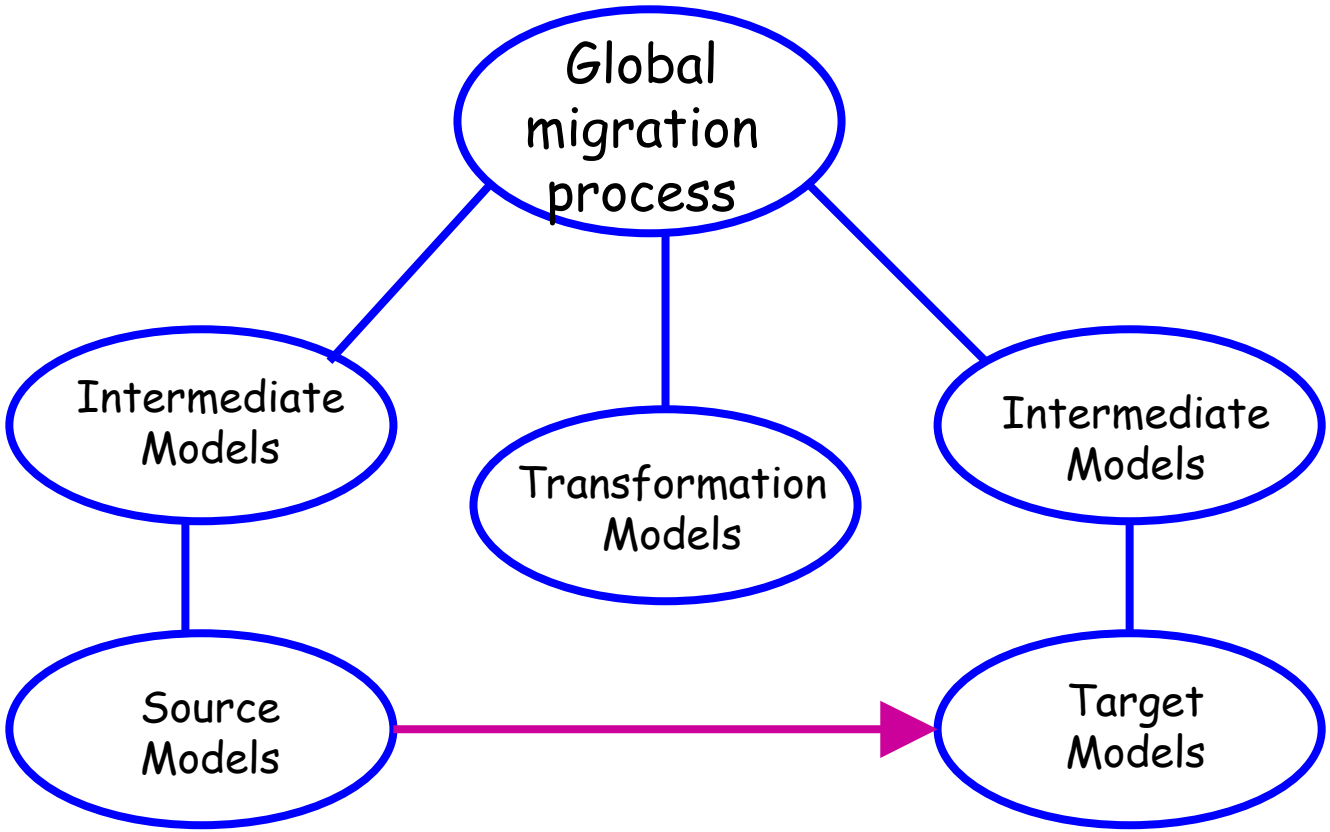
- Some hopes
  - ✓ Ontology-Driven transformations
  - ✓ Combining MP and MM
- Some dangers
  - ✓ Confusing model of the problem and model of the solution
  - ✓ UML executability

Meta-model driven model transformation



Compare with other work on algebras over ontologies.
Gio Wiederhold, …

Trams Project: Meta-model based migration framework

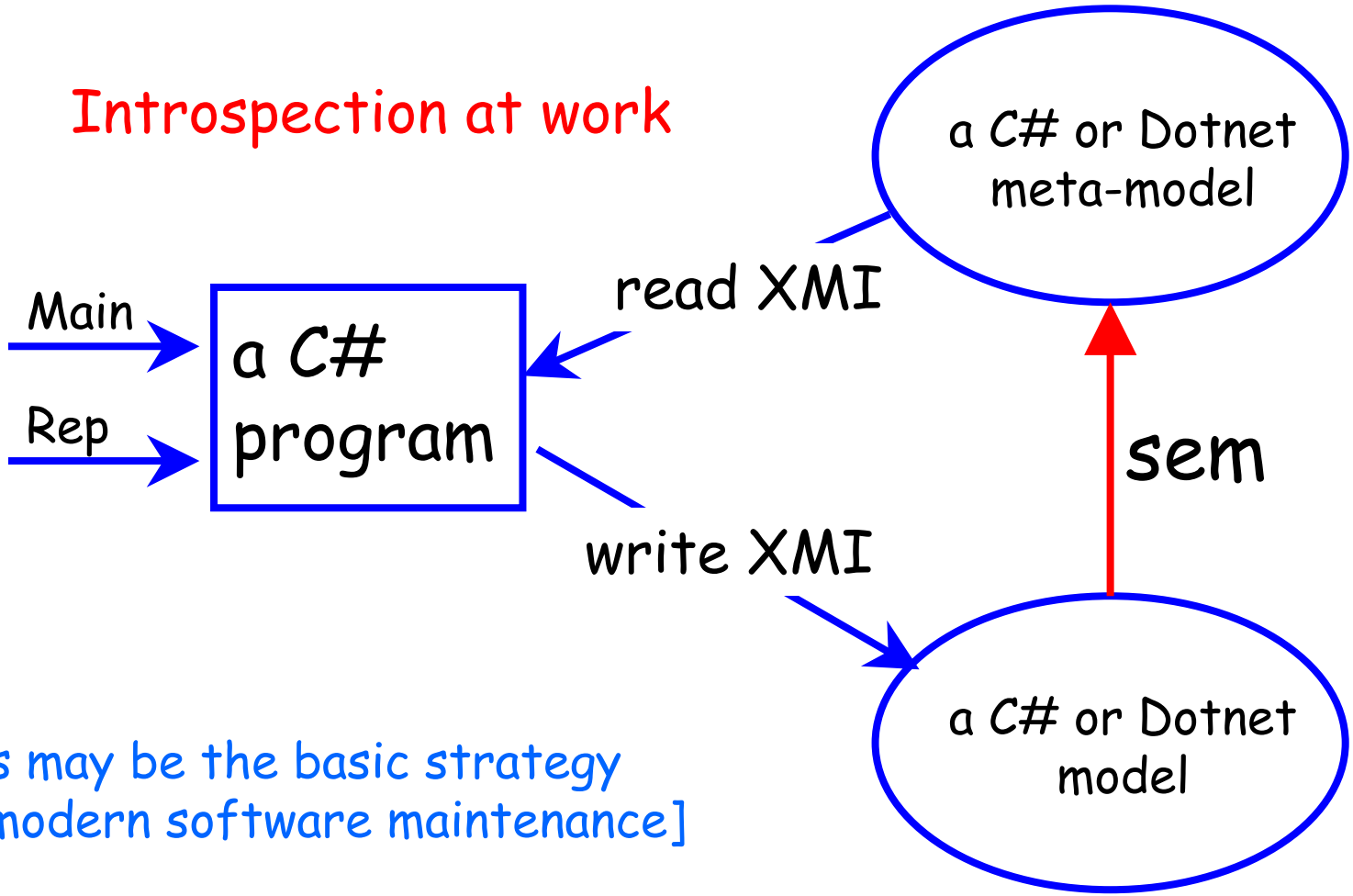


Legacy (Cobol, Relational Data Bases, etc)

Component technology EJB DotNet/C#, Etc.

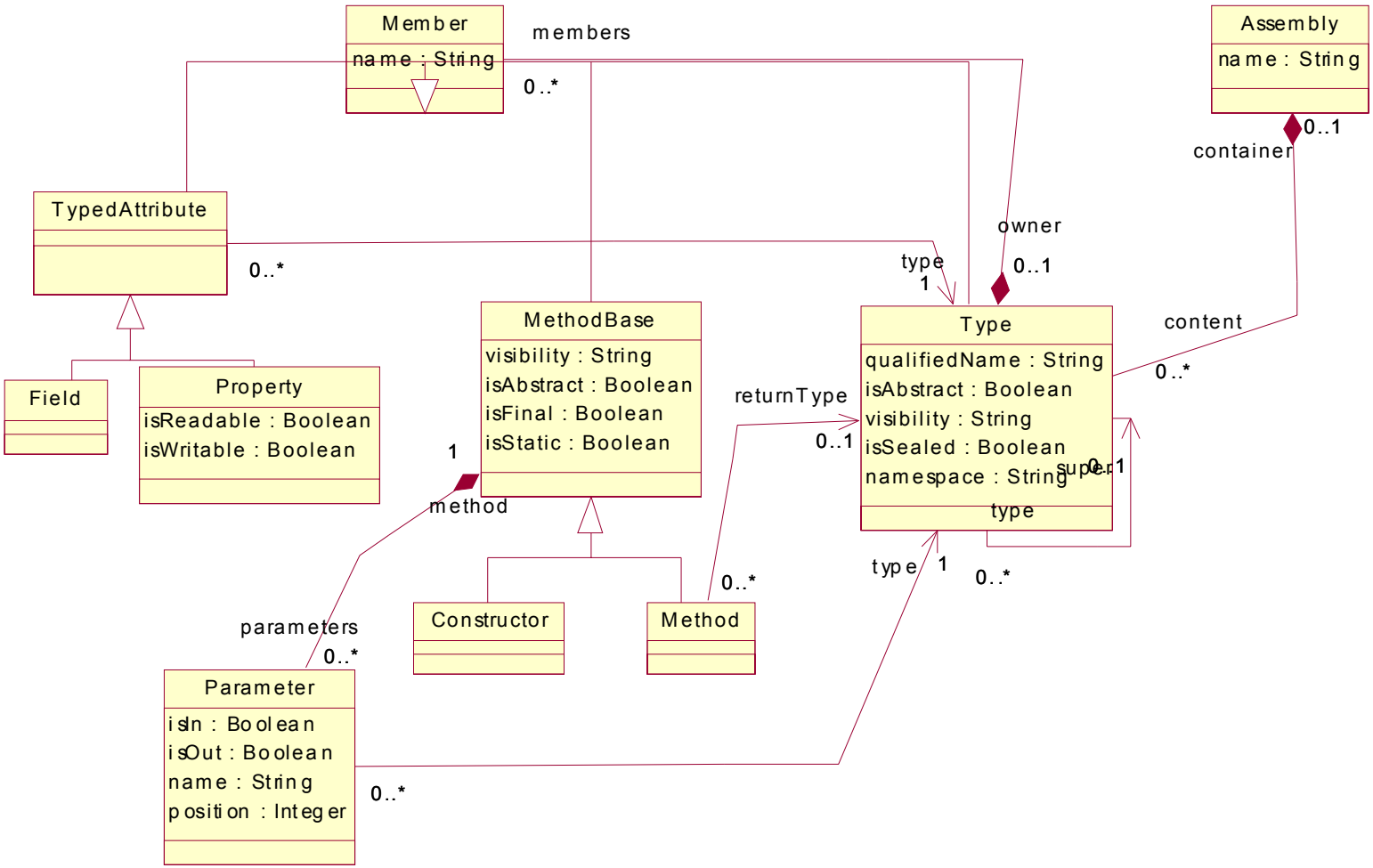
Combining the power of meta-modeling and meta-programming

Introspection at work

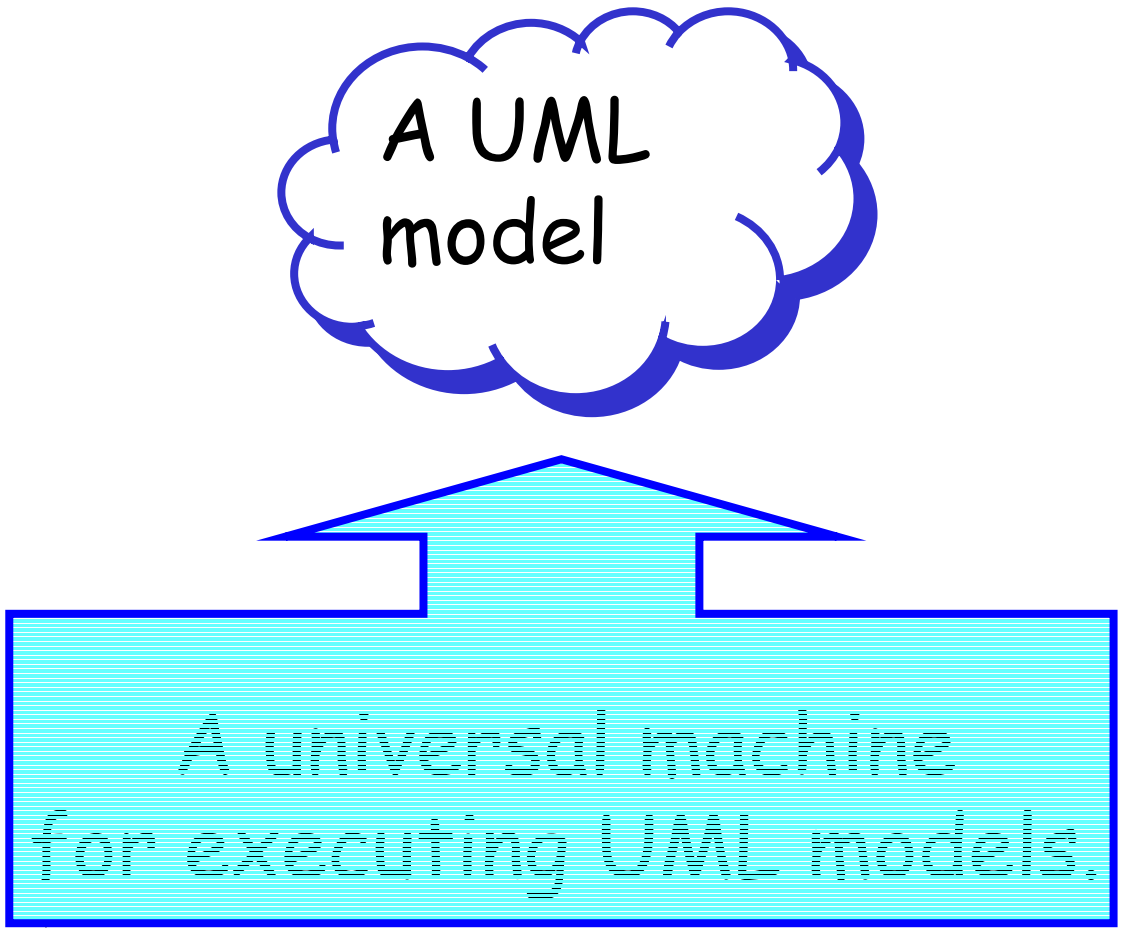


[This may be the basic strategy for modern software maintenance]

Part of the C#/DotNet MetaModel



UML executability



There is no canonical execution for a UML model.

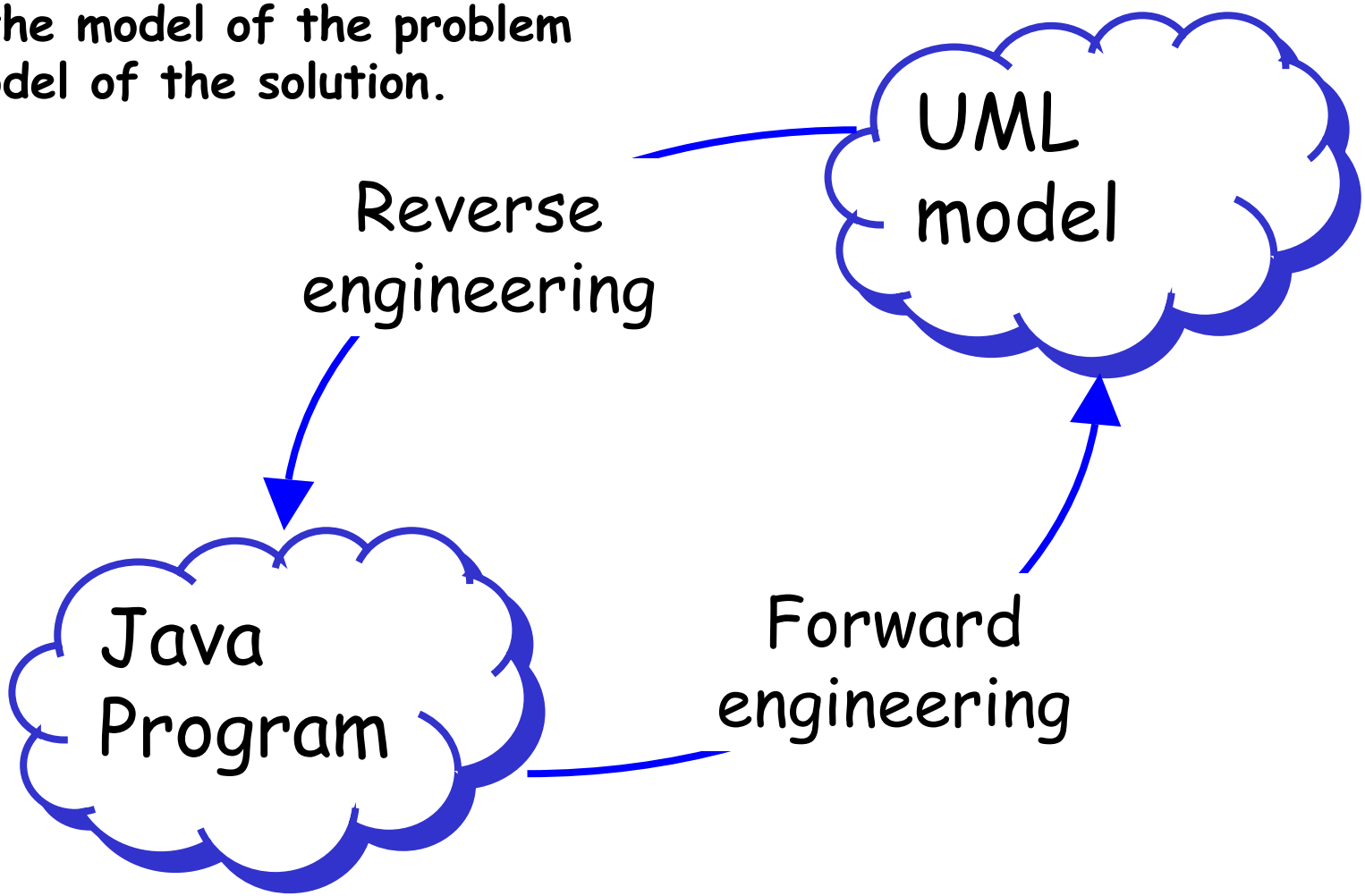
However the "Action Semantics for UML" will provide a meta-model to define execution schemes.

Fairy Tales or Horror Stories?

- Once upon a time there was a group of three young programmers that had to build a small system in a given object-oriented language. The person in charge of the project had to leave for some weeks and insisted before quitting that a UML model should be built before any code was produced. As soon as the guy left, one of the young programmers told the others that he had a good reverse engineering tool able to automatically produce UML models from program code. So they immediatly jumped on direct coding in their favorite programming language and had the program running some days before their boss return. The UML was produced and everybody was happy.

Round-Trip Engineering

Confusing the model of the problem and the model of the solution.



MDA: beyond the buzzword

- Modern model engineering techniques are ready for prime time in software engineering.
- They are based on:
  - ✓ A four level architecture (3+1)
  - ✓ A unique meta-meta-model (MOF),
    - ☒ with transfer and exchange mechanisms
    - ☒ with transformation mechanisms
    - ☒ with standard projection mechanisms on a variety of middlewares (CORBA first, Java and DotNet next, …)
  - ✓ A grawing collection of specialized meta-models (evolutive)
    - ☒ Object meta-models (Java, CLR, etc.)
    - ☒ Legacy meta-models (Relational, CWM)
    - ☒ Enterprise meta-models : Business objects, Healthcare, Transportation, Process & Rules, and much more
    - ☒ Product an process meta-models (e.g. workflow, RUP)
- Automatic and semi-automatic generation tools, from high abstraction standardized models to various middleware platforms will progressively appear in the coming years.

UNIVERSITÉ DE NANTES

Conclusion

Model engineering is the future of object technology

- As object and classes were seen in the 80's as "first class entities", with libraries of several hundred of classes hierarchically organized, models and meta-models are beginning to be considered alike in the 2000's.
- Libraries (lattices) of hundreds of meta-models (ontologies) of high abstraction and low granularity are beginning to appear. Each such mata-model may contains several hundreds of concepts and relations.
- Tools will be needed to work with these vast libraries of models and meta-models.
- This will have a rapid impact on the daily work of the information engineer.
- More research is urgently needed to bring together the people involved in the theory and practice of model engineering (ontologists, methodologists, software practitioners, information system builders, database specialists, etc.).

UNIVERSITÉ DE NANTES