# Model Engineering for Software Modernization

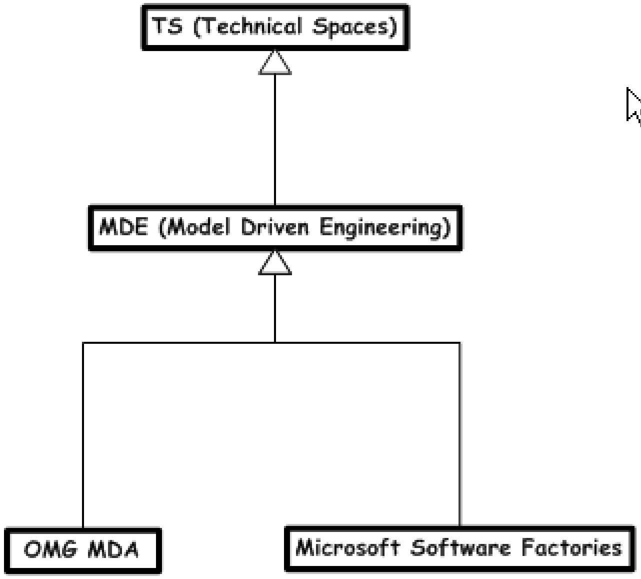
**Jean Bézivin**

Jean.Bezivin{noSpamAt)lina.univ-nantes.fr

ATLAS Group (INRIA & LINA),
University of Nantes, France
http://www.sciences.univ-nantes.fr/lina/atl/

# Agenda

- The industrial evolution (OMG MDA™, IBM EMF, Microsoft Software factories) and the MDE trend
- The need for sound principles (models as first class entities)
- Technology spaces or why MDE is not sufficient
- Why OT is not sufficient and what did we learn?
- Towards an open MDE platform
- Applications of MDE to Software modernization
- Conclusions & Perspectives



Agenda

# The industrial evolution (OMG, IBM, Microsoft)

# MDA™: OMG's new vision

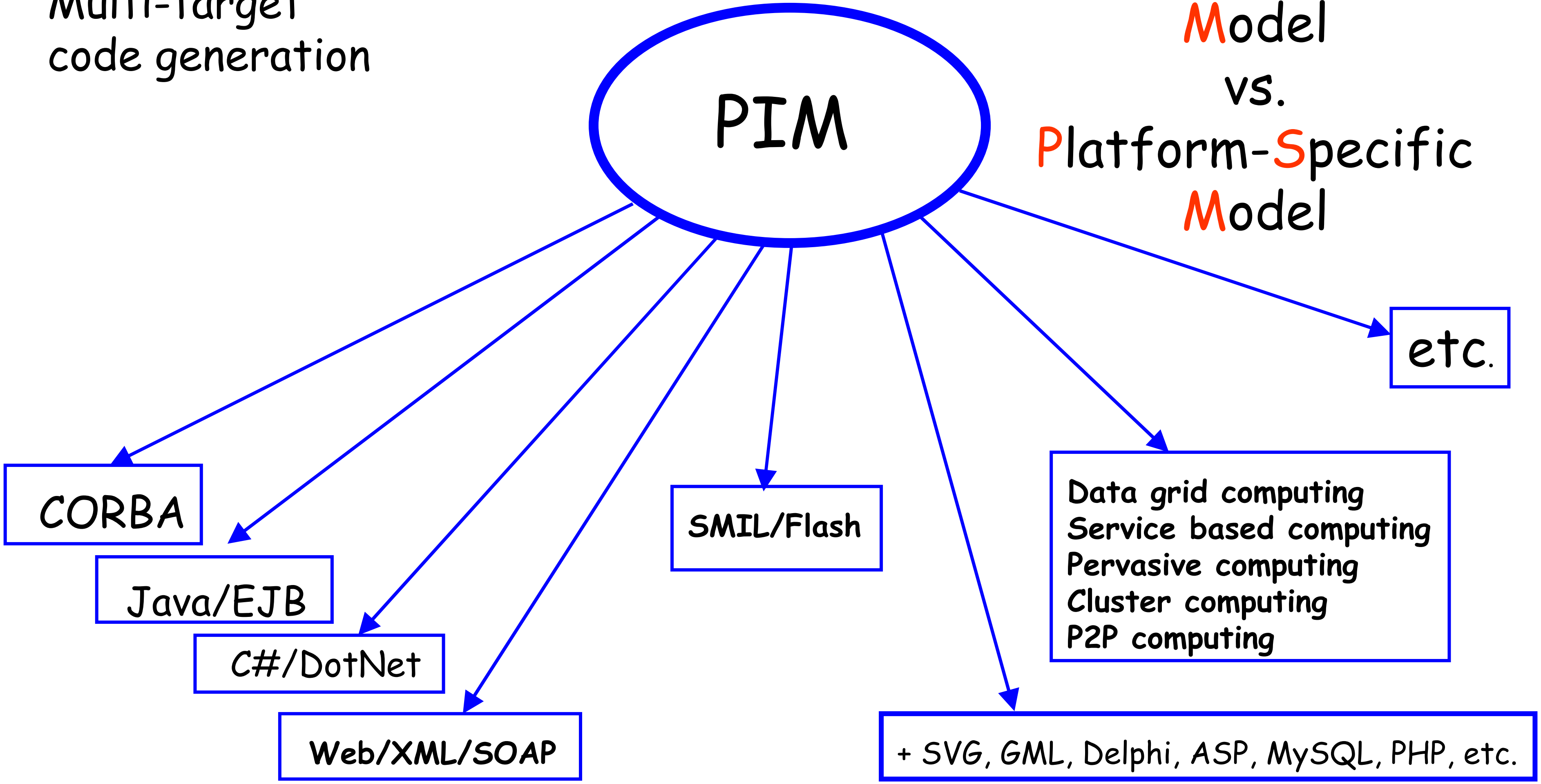
"OMG is in the ideal position to provide the model-based standards that are necessary to extend integration beyond the middleware approach… Now is the time to put this plan into effect. Now is the time for the Model Driven Architecture."

*Richard Soley and the OMG staff,*
*MDA Whitepaper Draft 3.2*
*November 27, 2000*

# ~~Write Once, Run Anywhere~~
# Model Once, Generate Anywhere

Multi-target
code generation

Platform-Independent
Model
vs.
Platform-Specific
Model



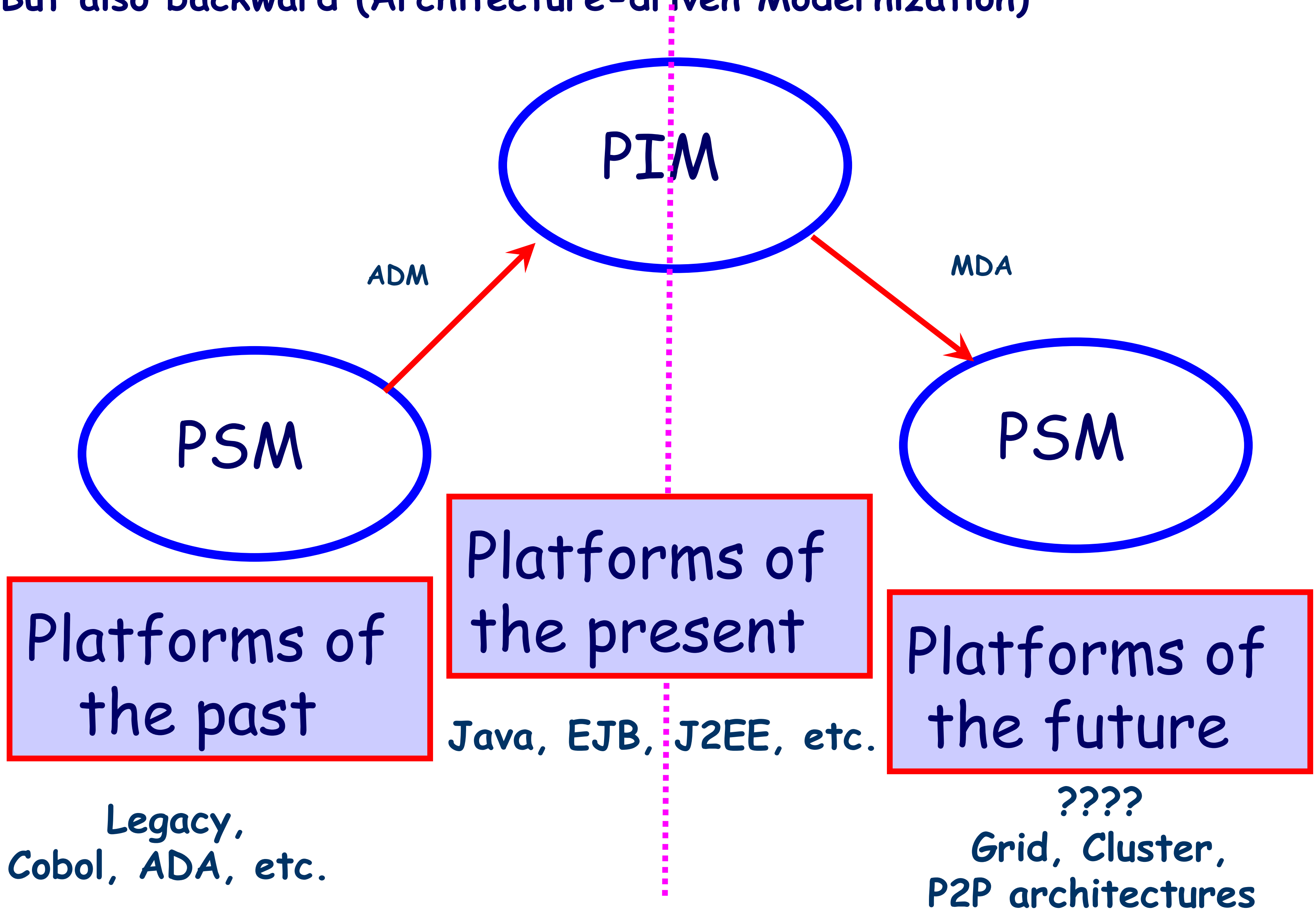
# Steve Cook (OOPSLA 2004 panel)

Suggests that MDA proponents fall into the following three camps:

1. The UML PIM camp: MDA involves the use of UML to build Platform Independent Models (PIMs) which are transformed into Platform Specific Models (PSMs) from which code is generated.
2. The MOF camp: MDA does not involve the use of UML, but instead the crucial technology is MOF, and the definition of modelling languages and language transformations using MOF.
3. The Executable UML camp: MDA involves building a UML compiler, making it a first class programming language.

Ref: Steve Cook Blog @: http://blogs.msdn.com/stevecook

# But also backward (Architecture-driven Modernization)



PIM

ADM

MDA

PSM

PSM

Platforms of the past

Platforms of the present

Platforms of the future

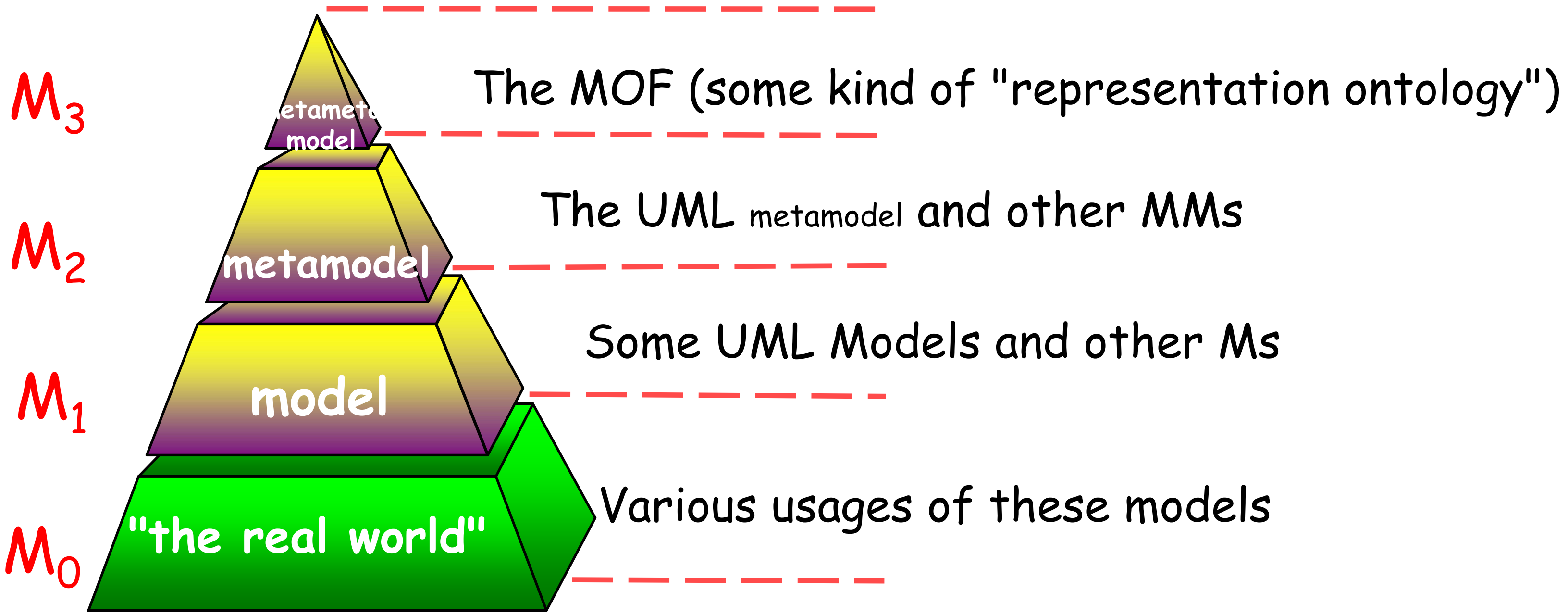
Java, EJB, J2EE, etc.

Legacy,
Cobol, ADA, etc.

????
Grid, Cluster,
P2P architectures

# The metamodelling stack

- UML
- CWM
- SPEM
- EDOC
- QVT
- MDA is based on a coordinated set of metamodels



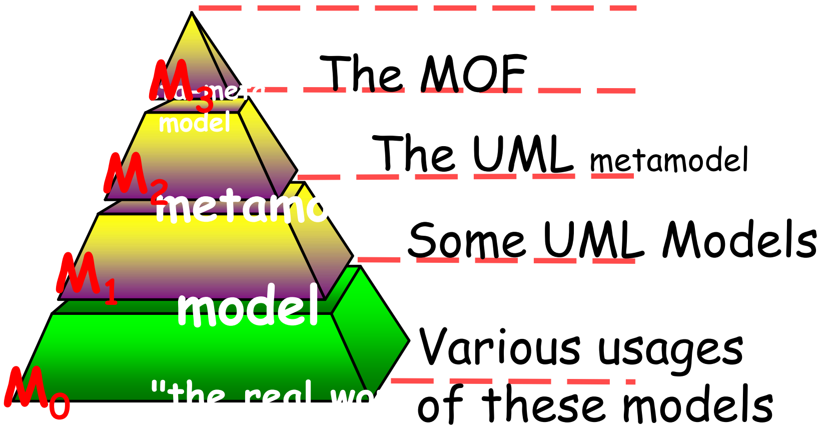
The MOF (some kind of "representation ontology")

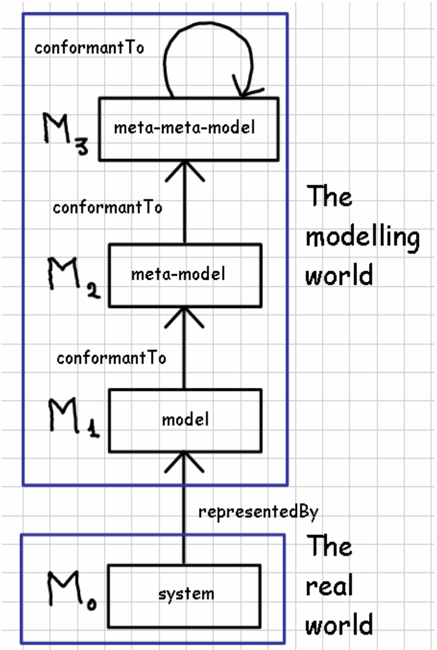
The UML metamodel and other MMs

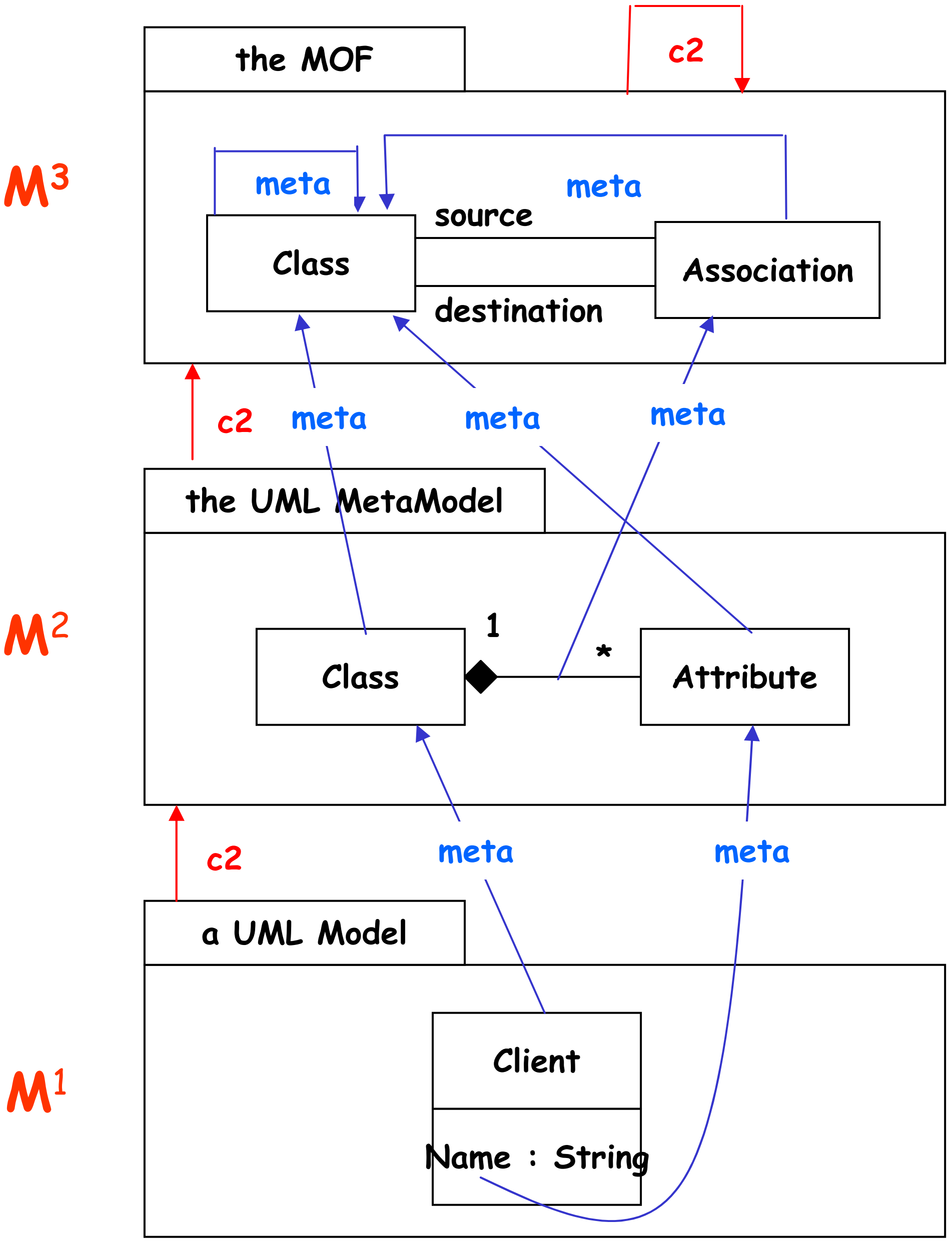
Some UML Models and other Ms

Various usages of these models

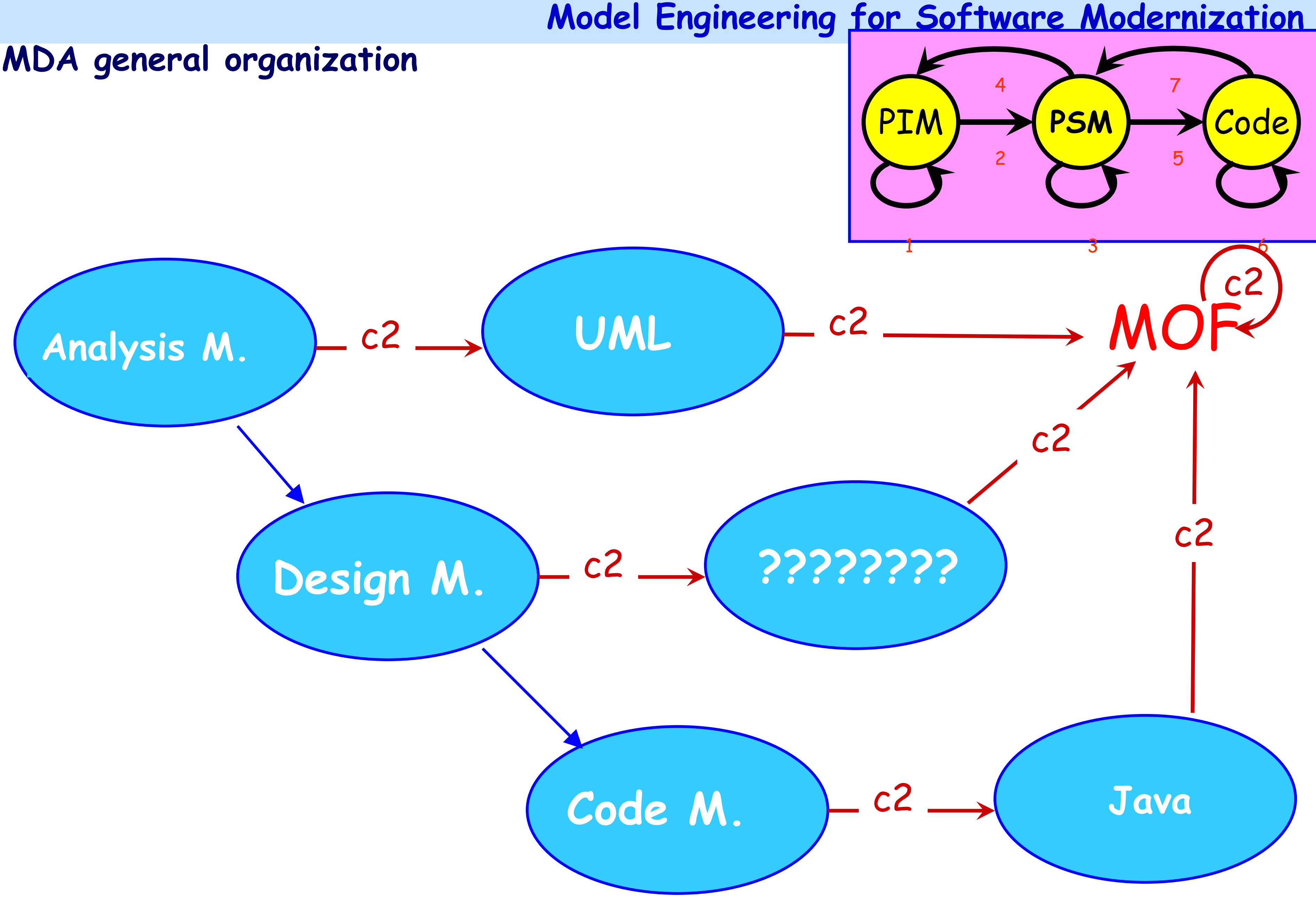
# Coordinated metamodels (conformance)



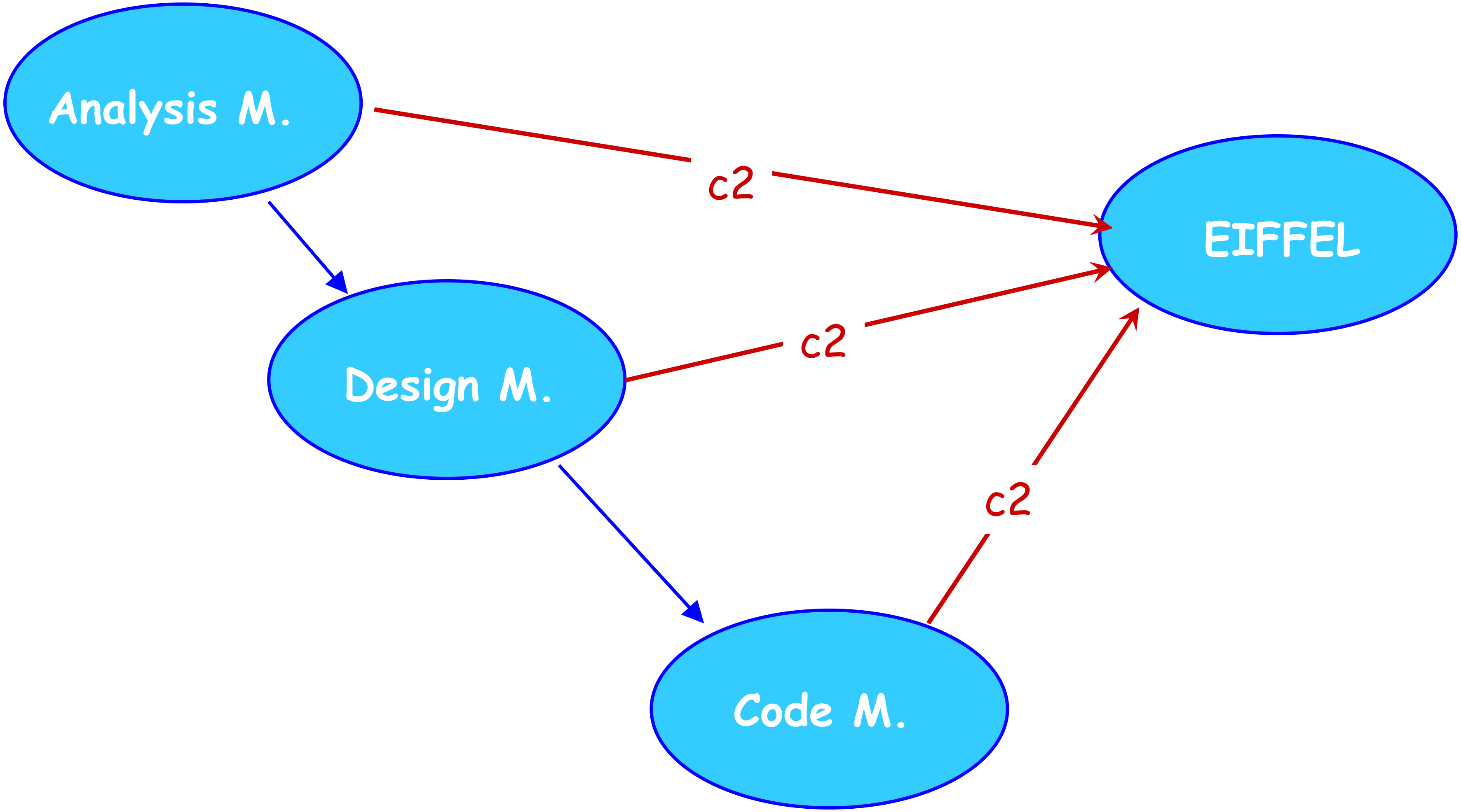




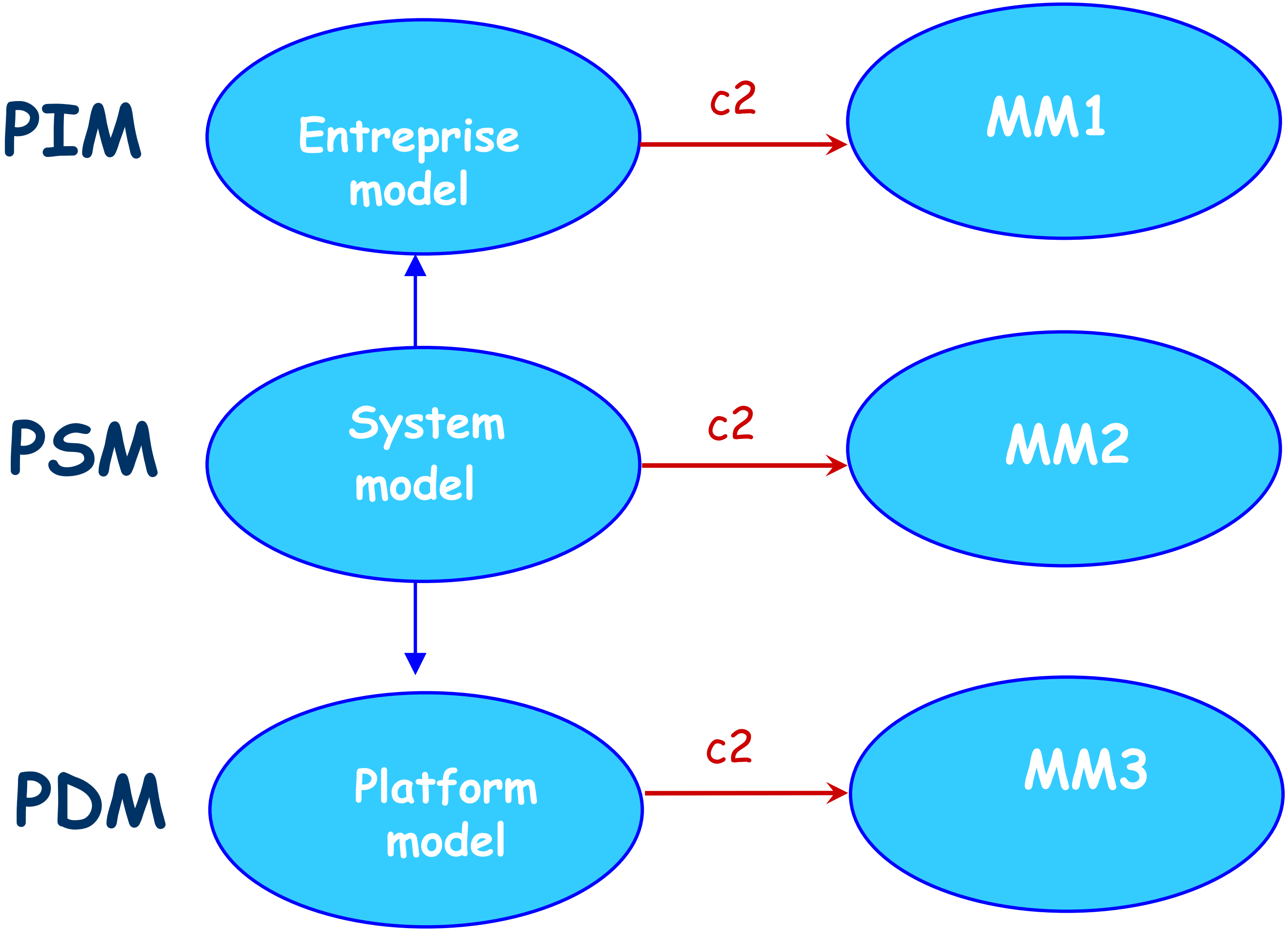
# MDA general organization



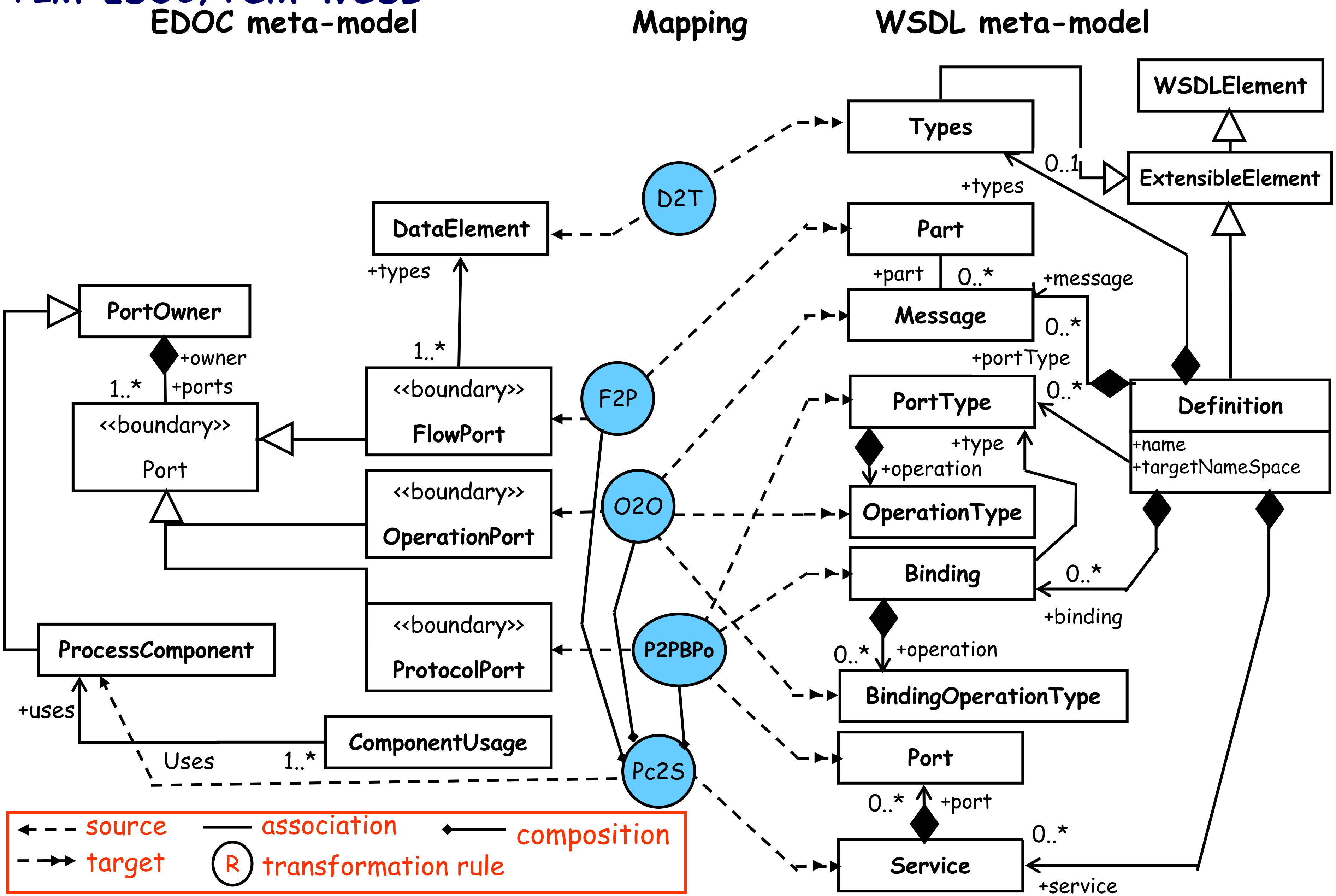
## Simplified vision of the world



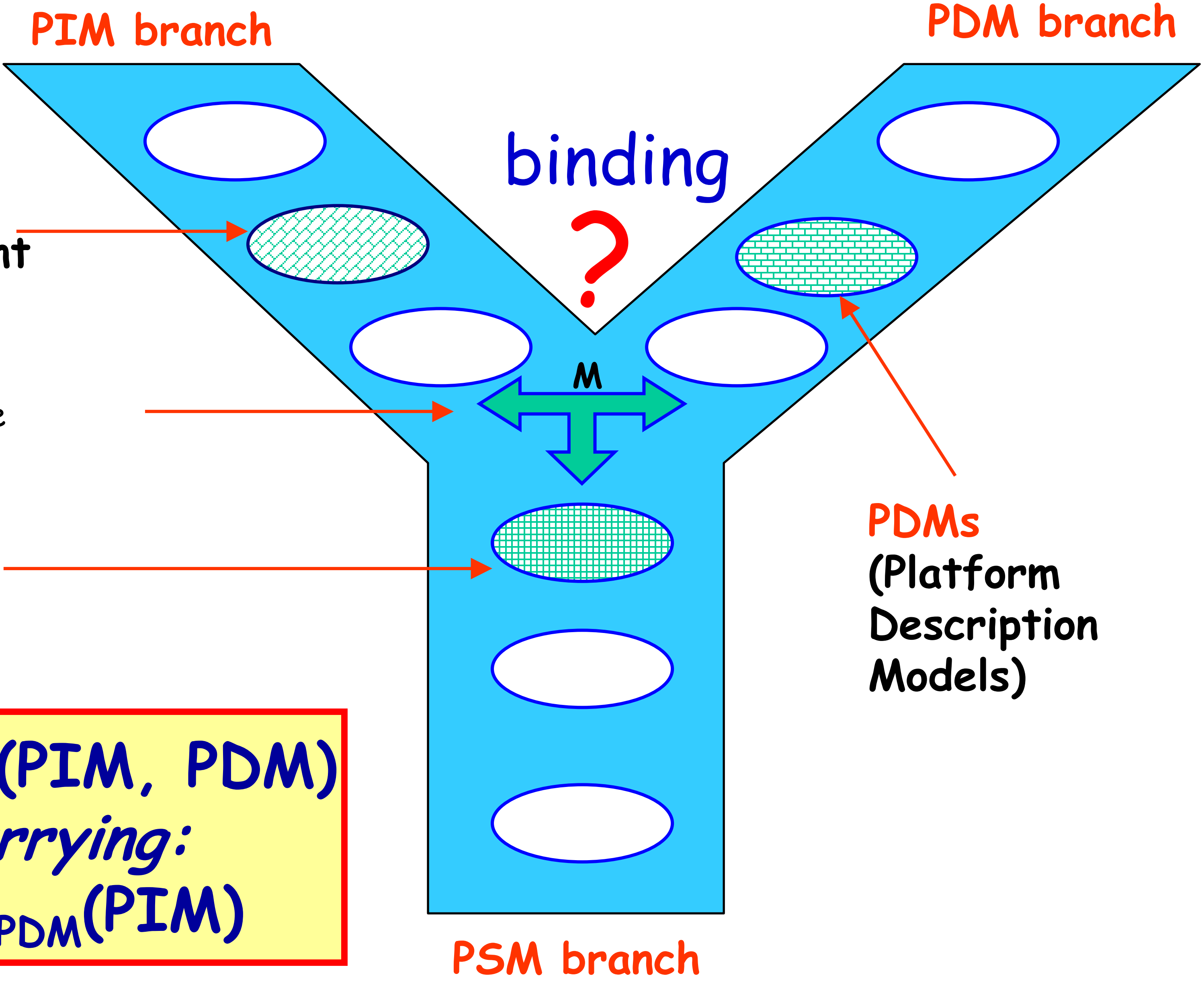
# General case



# PIM=EDOC; PSM=WSDL



# The initial Y conjecture : PSM = f(PIM, PDM)



PIMs (Platform Independent Models)

Merging phase

PSMs (Platform Specific Models)

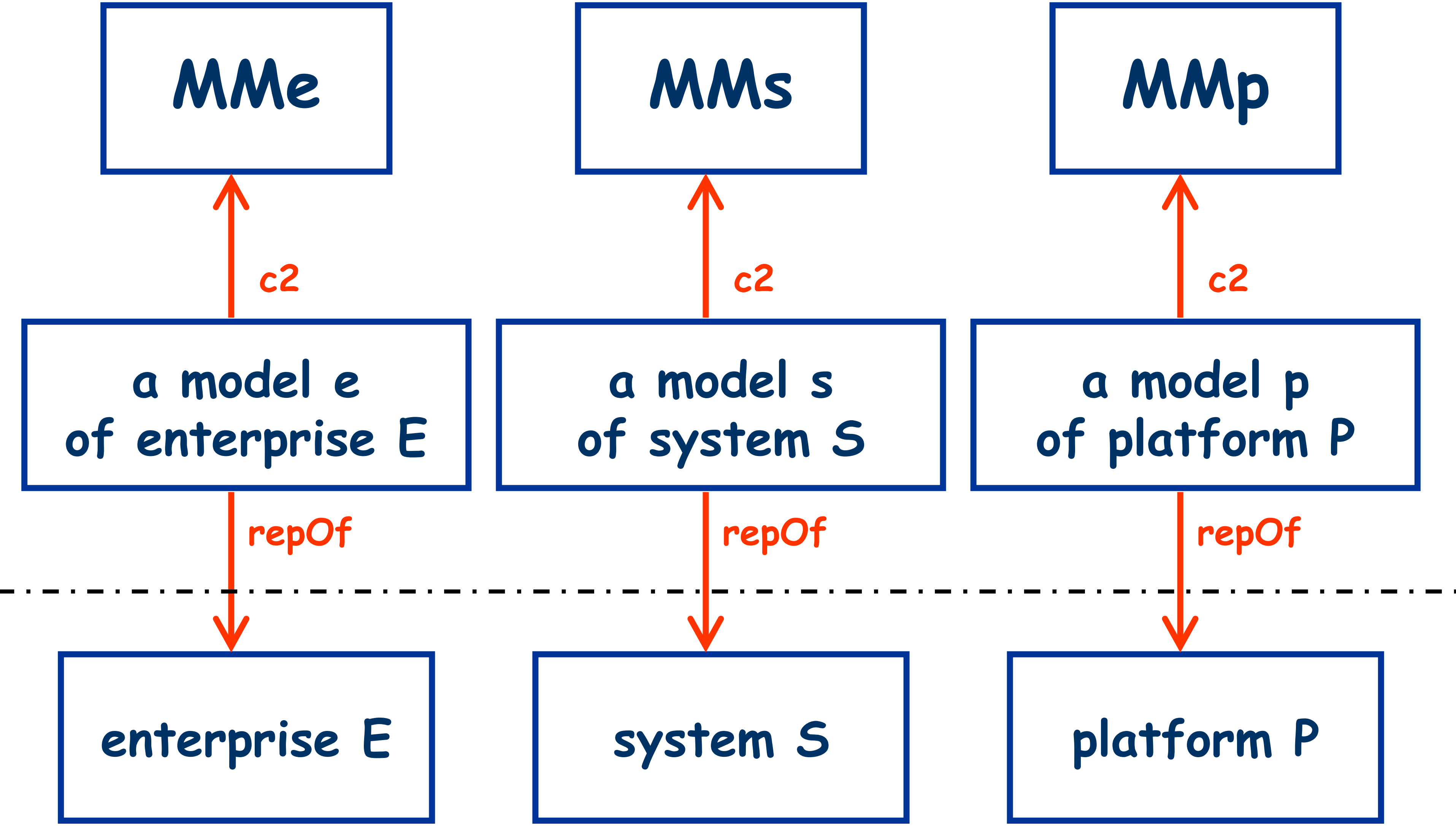
PDMs (Platform Description Models)

**PSM = f(PIM, PDM)**

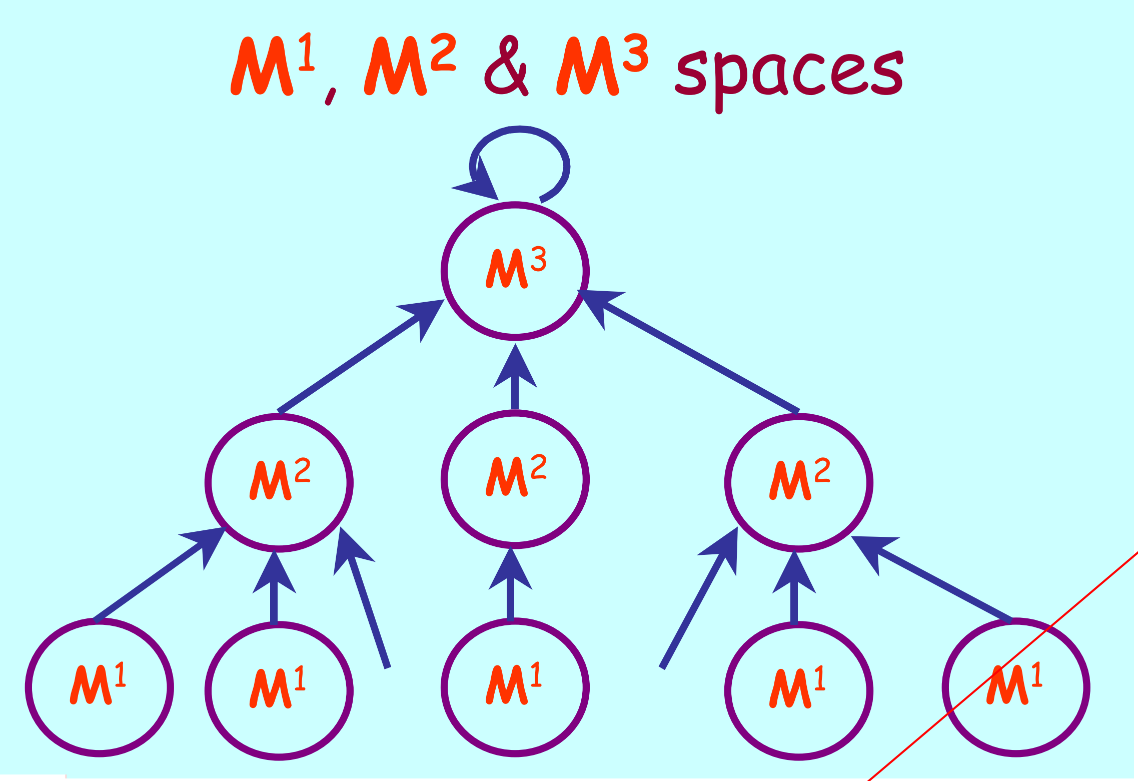
***or by currying:***

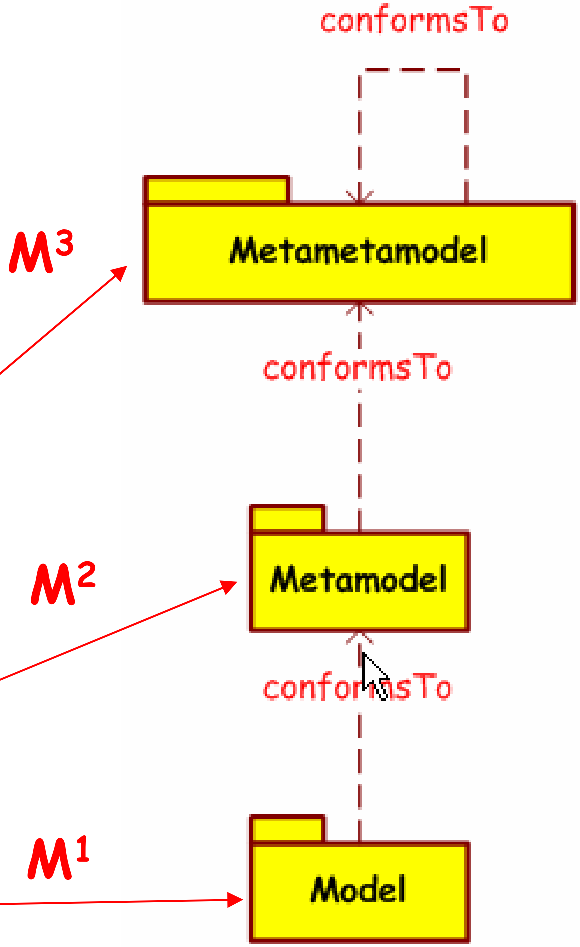
**PSM = $f_{PDM}$(PIM)**

## Abstract models of Platforms and Enterprises



# MDA in a nutshell





- One unique Metametamodel (the MOF)
- An important library of compatible Metamodels, each defining a DSL
- Each of the models is defined in the language of its unique metamodel

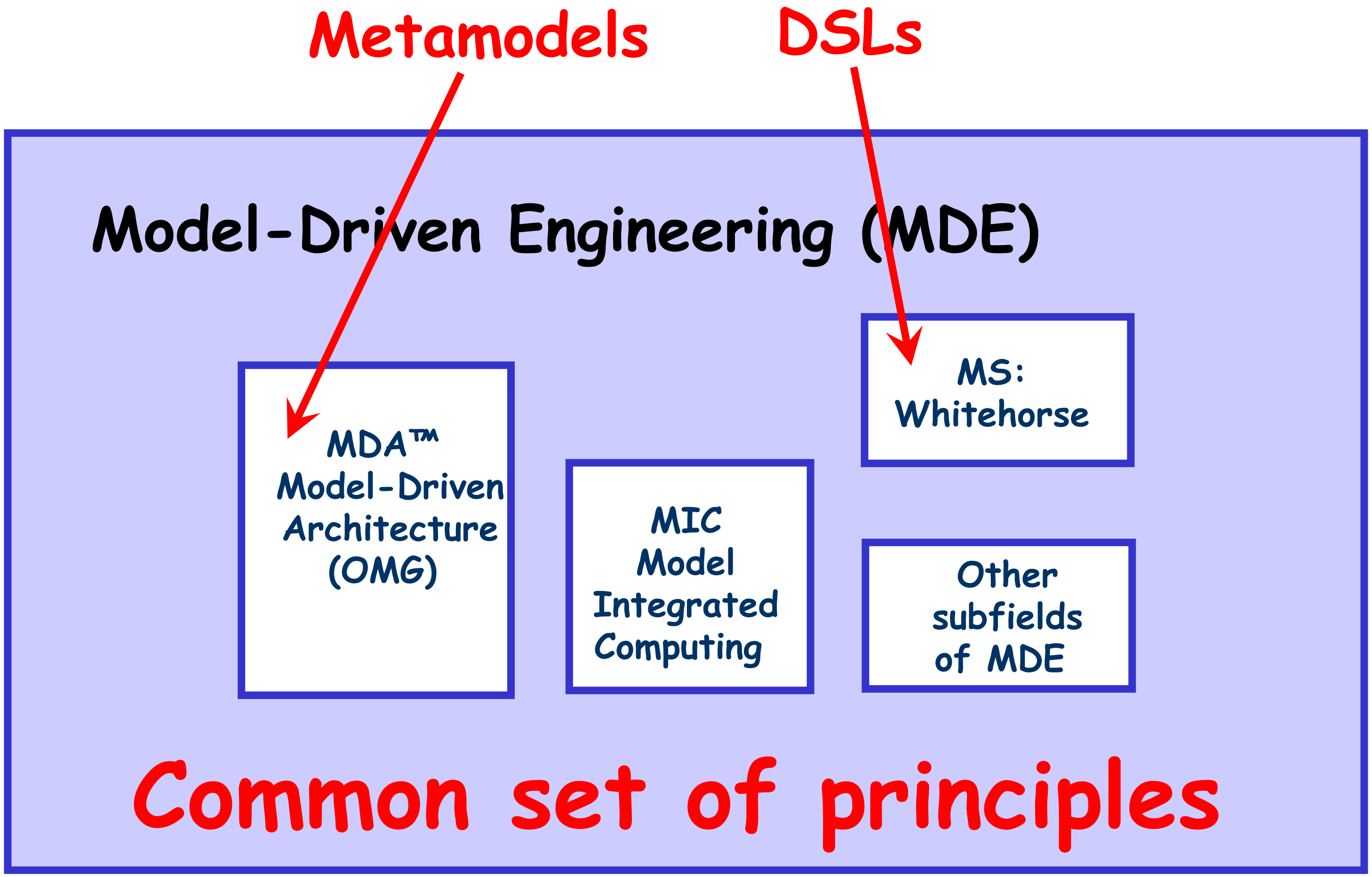
# A definition of MDA

OMG/ORMSC/2004-06-01 (The OMG MDA Guide): A Definition of MDA (The following was approved unanimously by 17 participants at the ORMSC plenary session, meeting in Montreal on 23 August 26, 2004. The stated purpose of these two paragraphs was to provide principles to be followed in the revision of the MDA Guide.)

- MDA is an OMG initiative that proposes to define a set of non-proprietary standards that will specify interoperable technologies with which to realize model-driven development with automated transformations. Not all of these technologies will directly concern the transformations involved in MDA.
- MDA does not necessarily rely on the UML, but, as a specialized kind of MDD (Model Driven Development), MDA necessarily involves the use of model(s) in development, which entails that at least one modeling language must be used.
- Any modeling language used in MDA must be described in terms of the MOF language, to enable the metadata to be understood in a standard manner, which is a precondition for any ability to perform automated transformations.

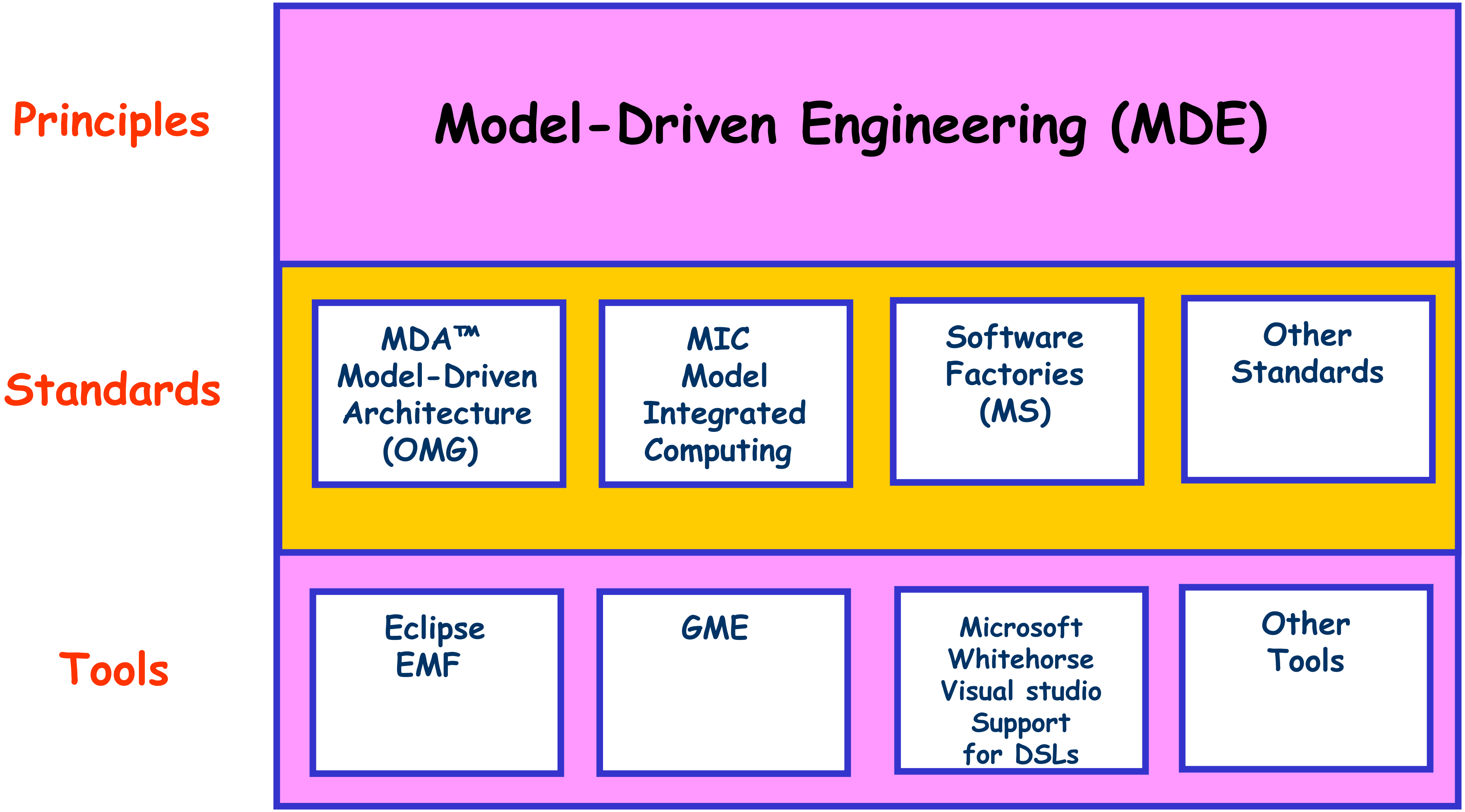
# MDE and the MDA™

"MDA™ is a specific MDD™ deployment effort around such industrial standards as MOF, UML, CWM, QVT, etc." (from OMG/MDA guide).


Metamodels
DSLs
Model-Driven Engineering (MDE)
MDA™ Model-Driven Architecture (OMG)
MIC Model Integrated Computing
MS: Whitehorse
Other subfields of MDE
Common set of principles

Terminological note: MDA and MDD are trademarks of OMG; MDE is not.

# Principles, standards and tools



Principles

Model-Driven Engineering (MDE)

Standards

MDA™ Model-Driven Architecture (OMG)

MIC Model Integrated Computing

Software Factories (MS)

Other Standards

Tools

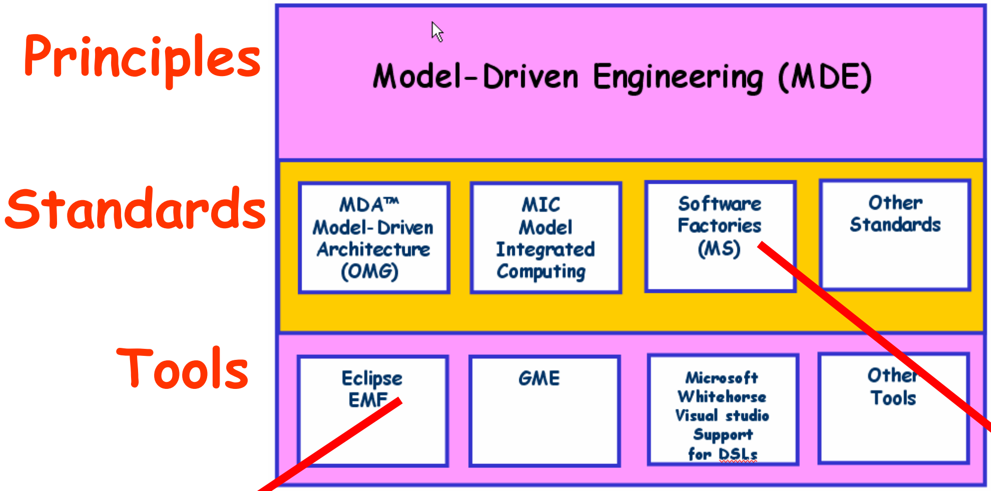
Eclipse EMF

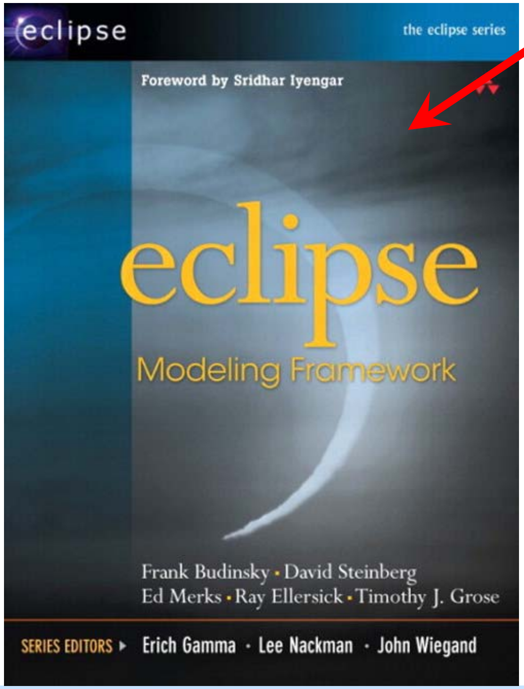
GME

Microsoft Whitehorse Visual studio Support for DSLs

Other Tools

# A very active industrial development area


Principles
Model-Driven Engineering (MDE)
Standards
MDA™ Model-Driven Architecture (OMG)
MIC Model Integrated Computing
Software Factories (MS)
Other Standards
Tools
Eclipse EMF
GME
Microsoft Whitehorse Visual studio Support for DSLs
Other Tools


eclipse

the eclipse series

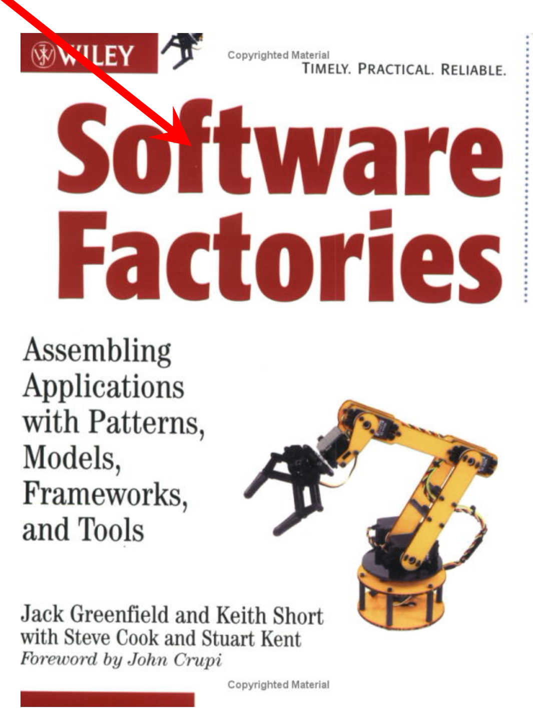
Foreword by Sridhar Iyengar

eclipse

Modeling Framework

Frank Budinsky · David Steinberg
Ed Merks · Ray Ellersick · Timothy J. Grose

SERIES EDITORS ▸ Erich Gamma · Lee Nackman · John Wiegand


WILEY

Copyrighted Material

TIMELY. PRACTICAL. RELIABLE.

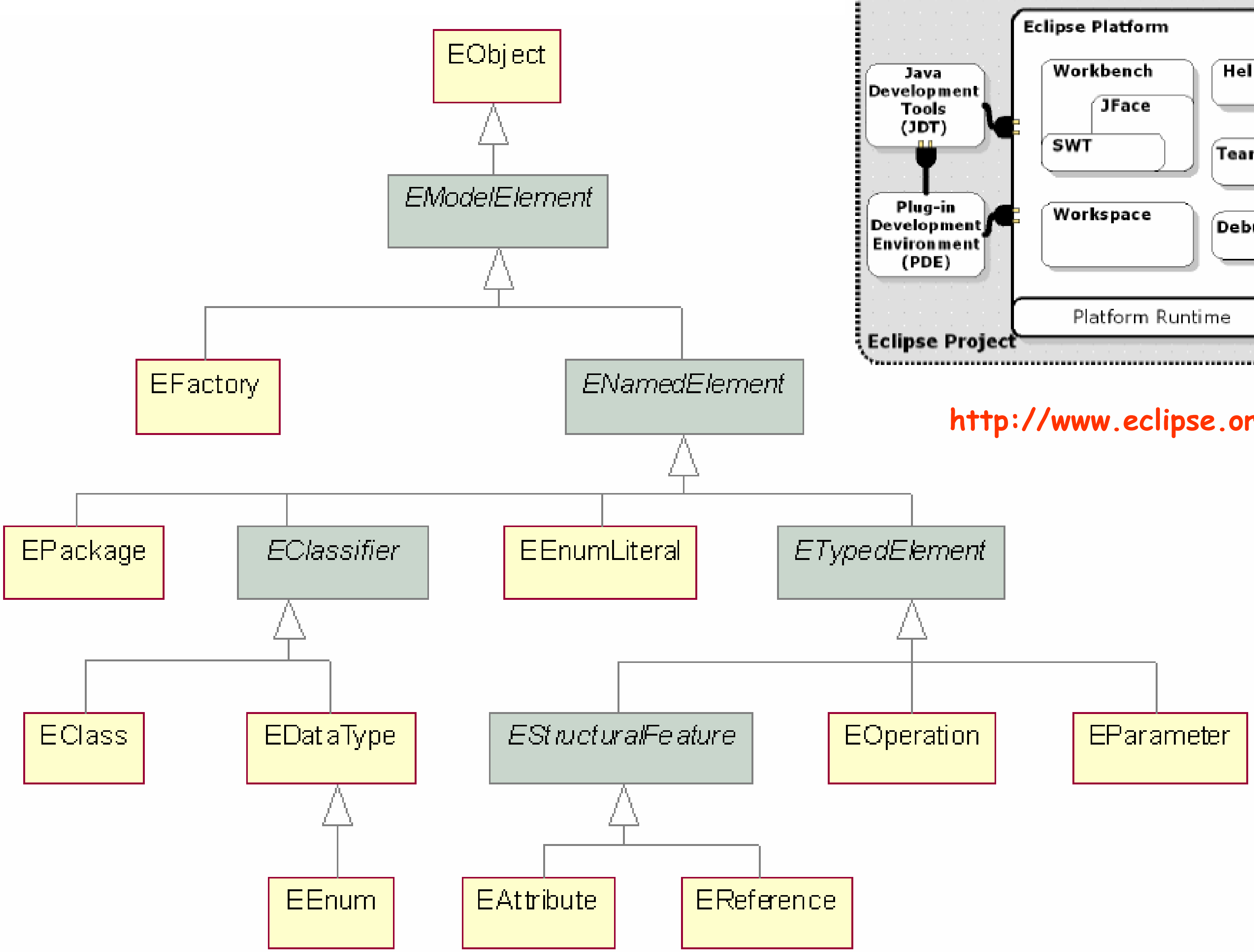
Software Factories

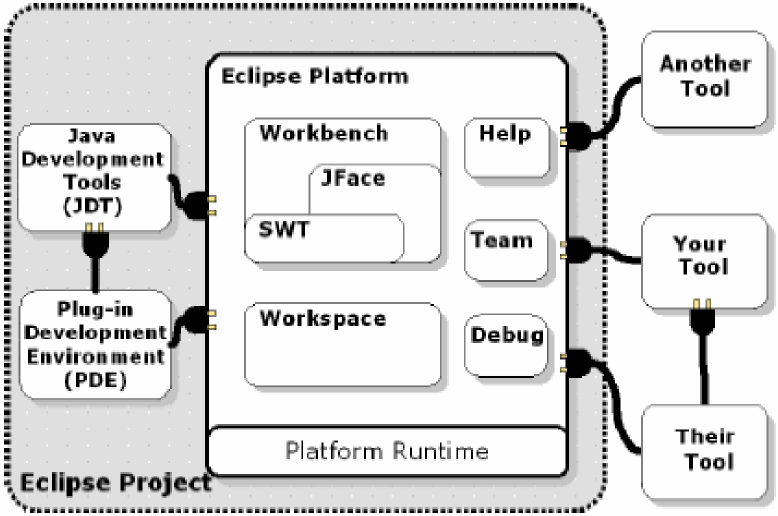
Assembling Applications with Patterns, Models, Frameworks, and Tools

Jack Greenfield and Keith Short
with Steve Cook and Stuart Kent
*Foreword by John Crupi*

Copyrighted Material

# EMF: The ECORE metametamodel





http://www.eclipse.org/emf/

# Microsoft Whitehorse, etc.

- SDK to be released in late 2005
  - First presentation at OOPSLA, Vancouver, Oct. 2004
  - See S. Cook, S. Kent and K. Short Blogs
- Aims to be closer to a metaCASE tool than Eclipse
- Not UML-based
- Models strongly tied to code
  - Reverse engineering/synchronization
  - Reliance on Microsoft's platforms (Visual studio)

**… Modeling is the future …**

Bill Gates

# Whitehorse : Bill Gates on models

… Modeling is the future …

You know UML [Unified Modeling Language] made the meta-models a little complex, so I don't think UML alone is the answer …

And the promise here is that you write a lot less code, that you have a model of the business process. And you just look at that visually and say here is how I want to customize it …

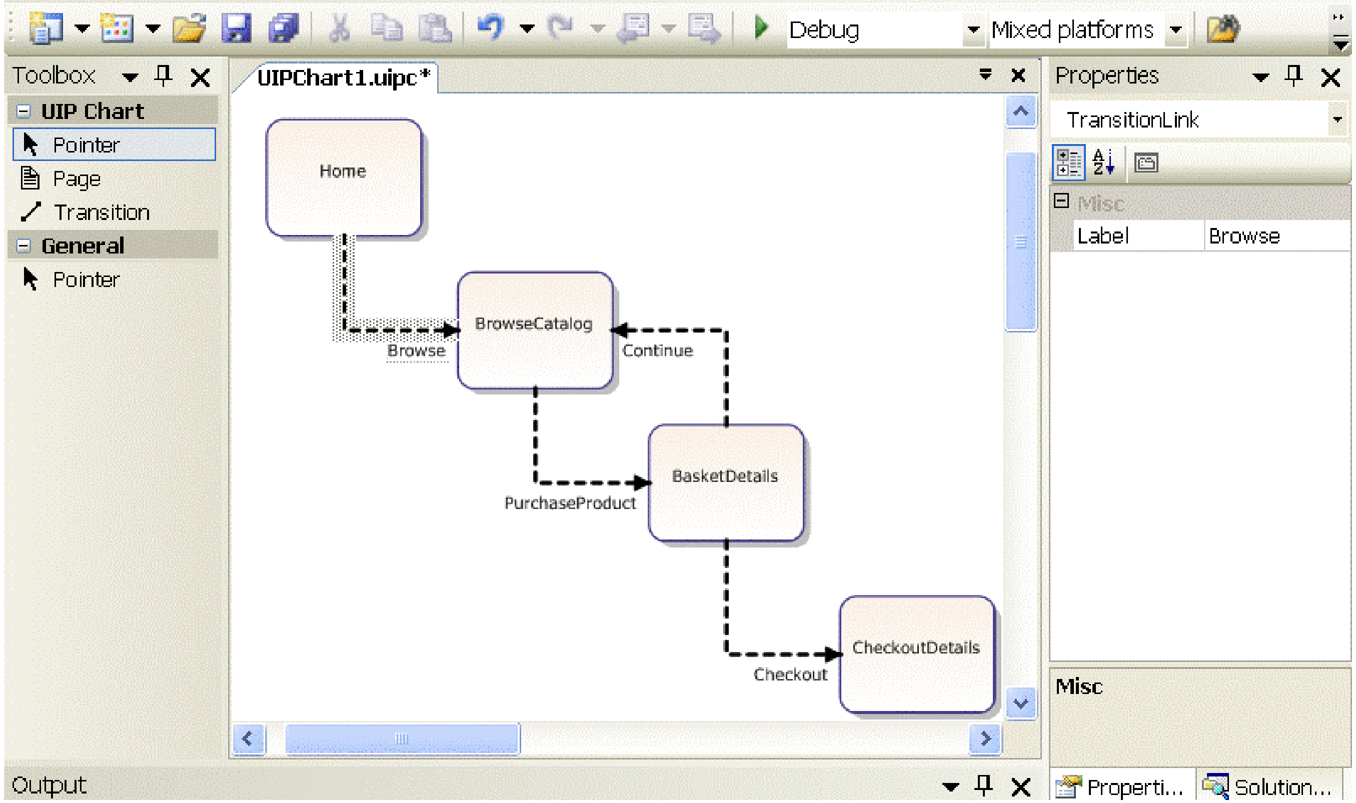
So even a business could express in a formal, modeled way, not just scribbling on paper, how the business process is changing over time or how it's different from other companies. So instead of having lots of code behind that, you just have visual, essentially model, customization …

So, I think we believe that. There are certainly some people from IBM who have that same vision, and I think it'll be healthy competition between the two of us because today's modeling products fall short. That's one part of Visual Studio 2005, that we do have some neat things coming along that will be part of it that we haven't shown completely …

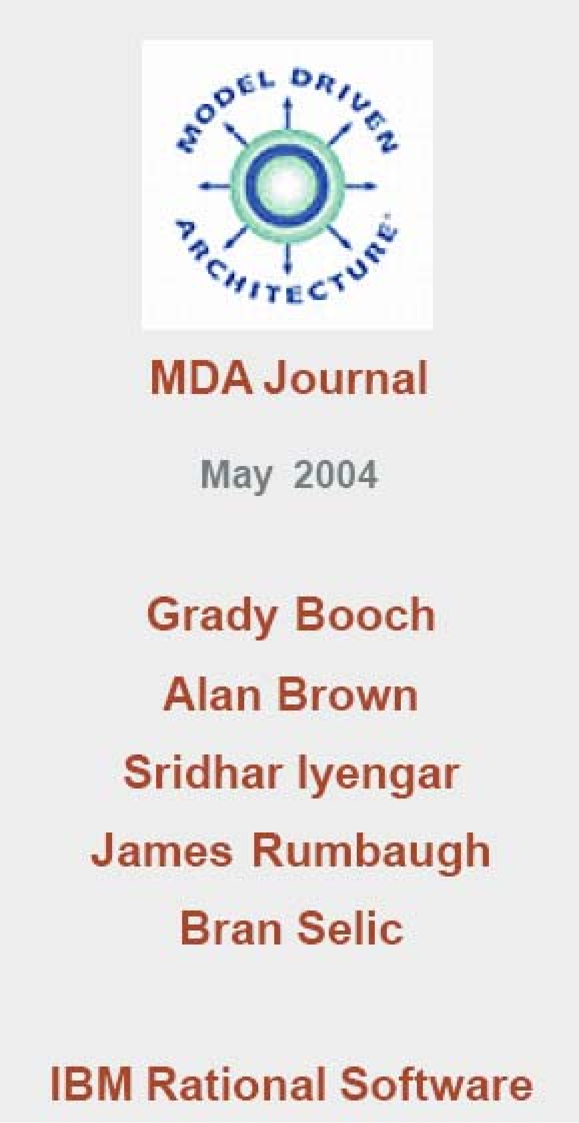
So, modeling is pretty magic stuff, whether it's management problems or business customization problems or work-flow problems, visual modeling. Even the Office group now really gets that for document life-cycle rights management, that this visual modeling will be key to them. Business intelligence, where you let people navigate through things, is another area where modeling could be used. It's probably the biggest thing going on. And both Visual Studio and Office need to be on top of that …

# MDE@Microsoft

- Microsoft is releasing a suite of tools to make it easy to construct graphical designers hosted in Visual Studio for editing domain specific languages (DSL).



# IBM on MDA : Three complementary ideas



MDA Journal

May 2004

Grady Booch

Alan Brown

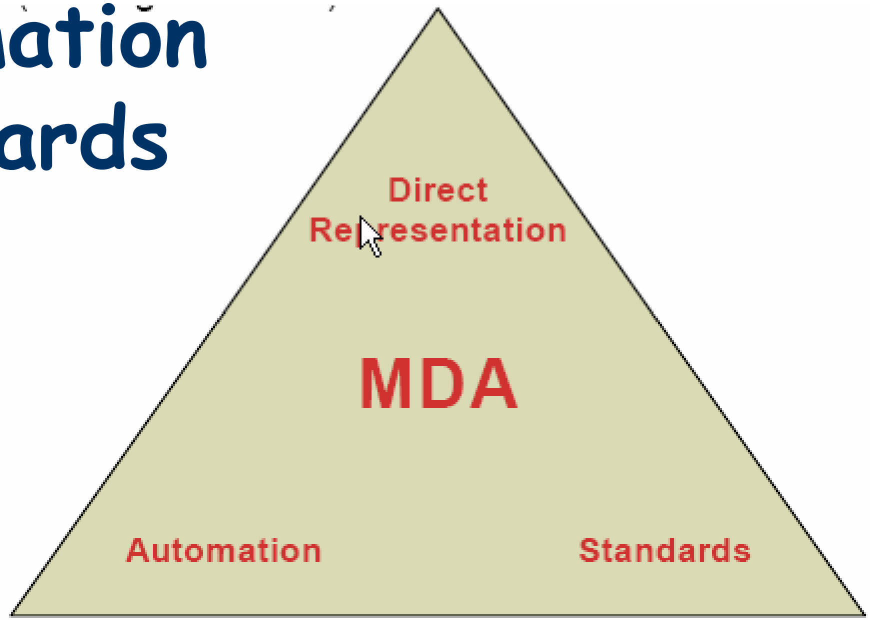
Sridhar Iyengar

James Rumbaugh

Bran Selic

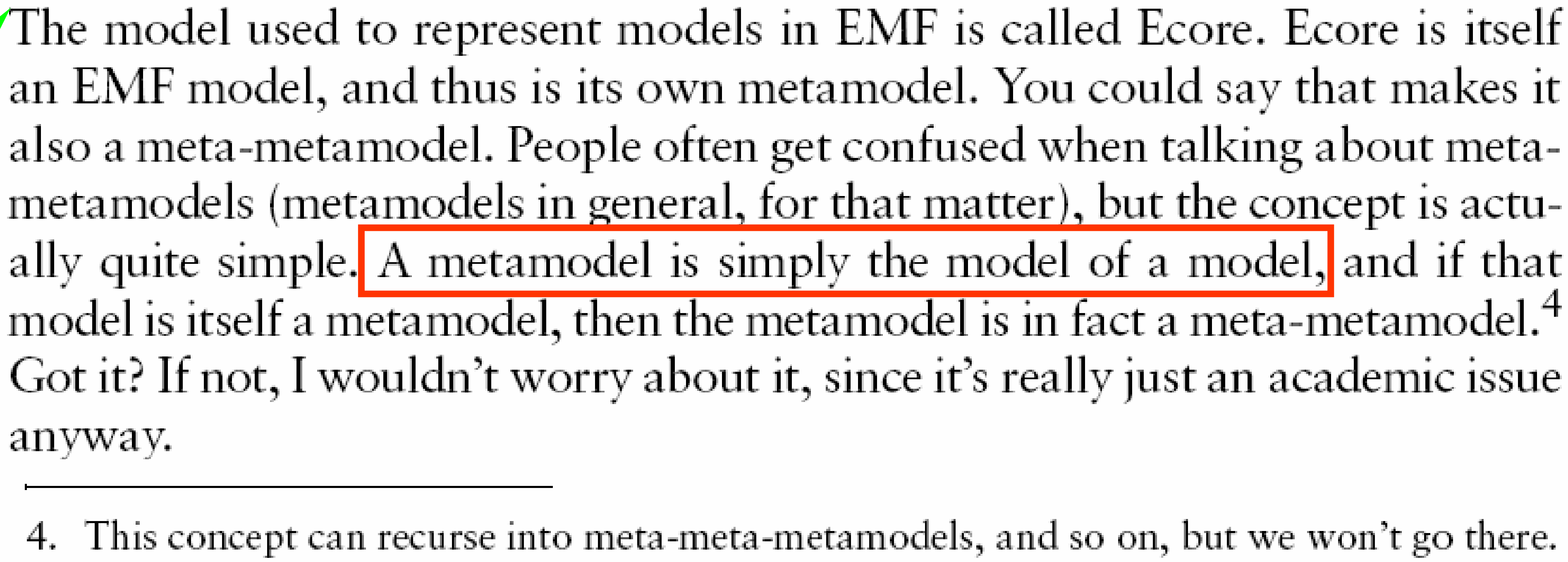
IBM Rational Software

1. Direct representation
2. Automation
3. Standards



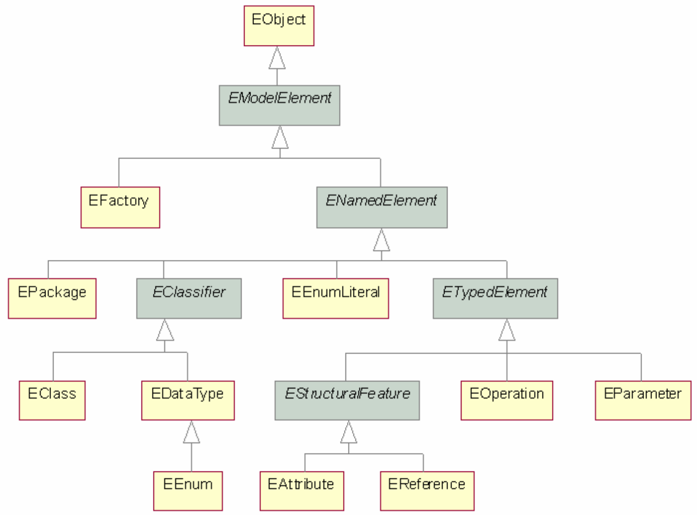
Direct representation => multiple languages  
Danger of fragmentation  
Need coordination  
How to coordinate?  
Short answer: metametamodel  
But what is a metametamodel?

# Just an academic issue anyway?

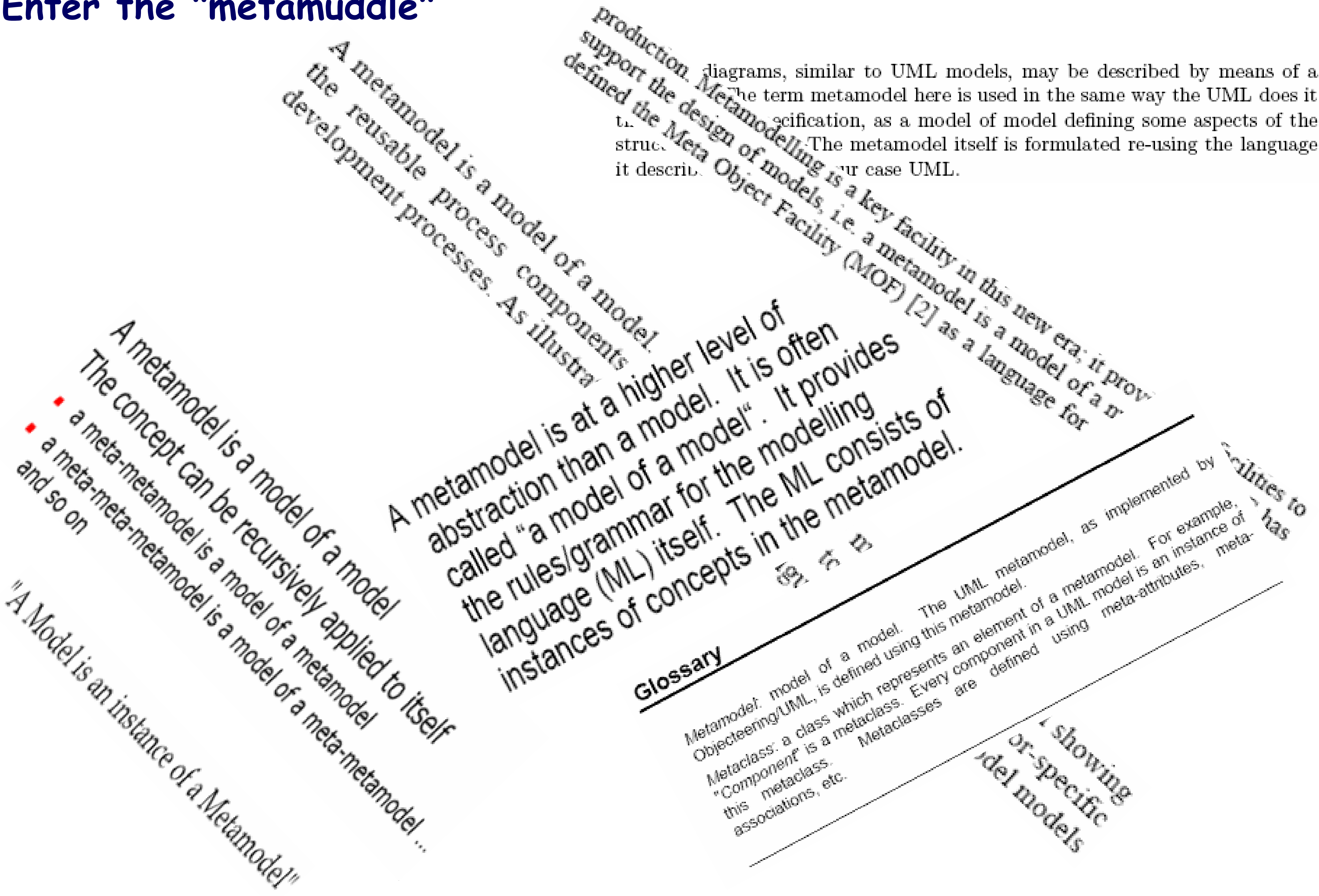


The model used to represent models in EMF is called Ecore. Ecore is itself an EMF model, and thus is its own metamodel. You could say that makes it also a meta-metamodel. People often get confused when talking about meta-metamodels (metamodels in general, for that matter), but the concept is actually quite simple. A metamodel is simply the model of a model, and if that model is itself a metamodel, then the metamodel is in fact a meta-metamodel.[4] Got it? If not, I wouldn't worry about it, since it's really just an academic issue anyway.

[4] This concept can recurse into meta-meta-metamodels, and so on, but we won't go there.



# Enter the "metamuddle"



diagrams, similar to UML models, may be described by means of a … The term metamodel here is used in the same way the UML does it … ecification, as a model of model defining some aspects of the struc… The metamodel itself is formulated re-using the language it describ… ur case UML.

production. Metamodelling is a key facility in this new era; it prov… support the design of models, i.e. a metamodel is a model of a m… defined the Meta Object Facility (MOF) [2] as a language for

A metamodel is a model of a model. … the reusable process components … development processes. As illustra…

A metamodel is a model of a model

The concept can be recursively applied to itself

- a meta-metamodel is a model of a metamodel
- a meta-meta-metamodel is a model of a meta-metamodel ...

and so on

"A Model is an instance of a Metamodel"

A metamodel is at a higher level of abstraction than a model. It is often called "a model of a model". It provides the rules/grammar for the modelling language (ML) itself. The ML consists of instances of concepts in the metamodel.

**Glossary**

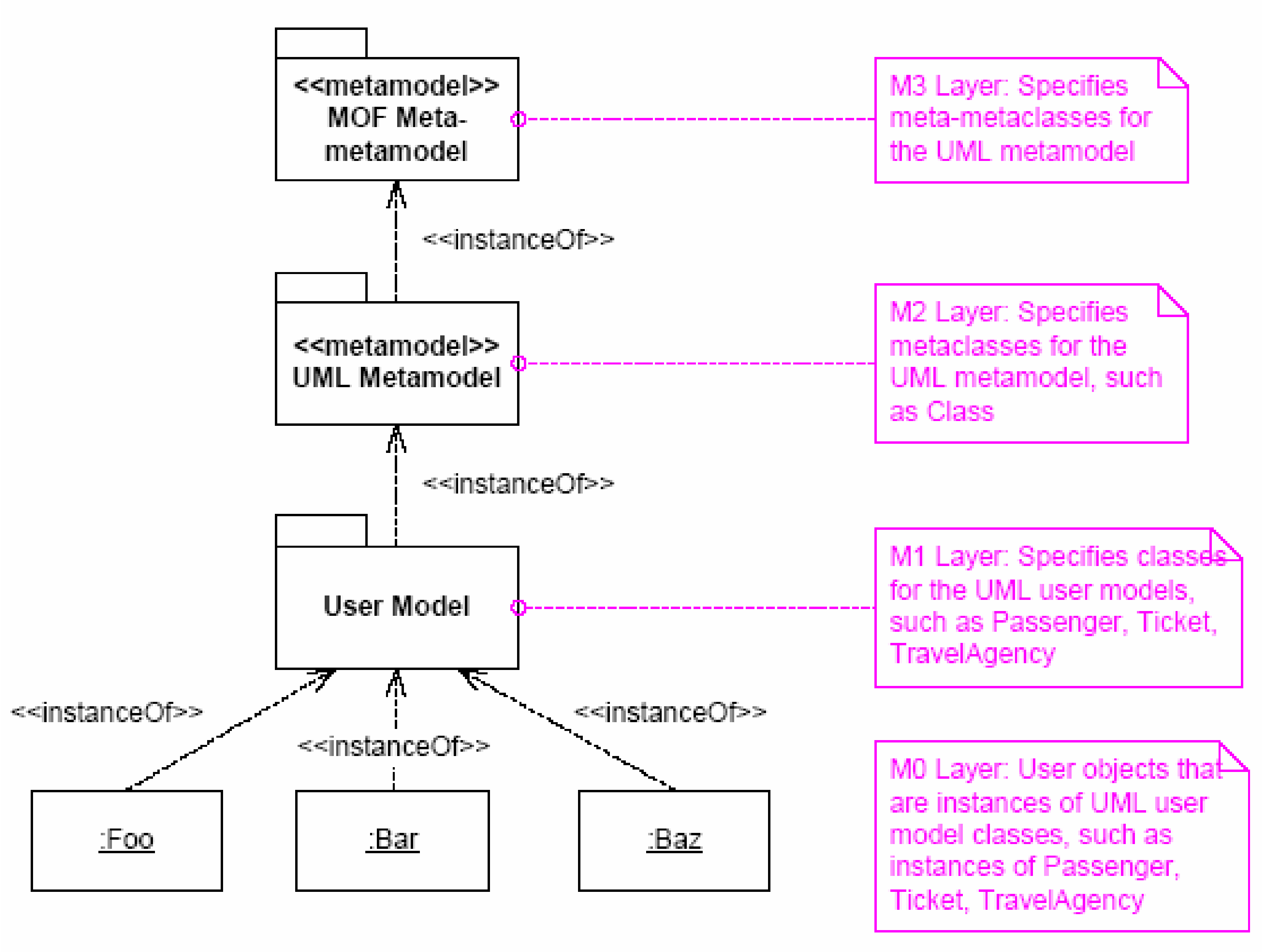
*Metamodel*: model of a model. The UML metamodel, as implemented by Objecteering/UML, is defined using this metamodel.

*Metaclass*: a class which represents an element of a metamodel. For example, *"Component"* is a metaclass. Every component in a UML model is an instance of this metaclass. Metaclasses are defined using meta-attributes, meta-associations, etc.

…ilities to … has

… showing … or-specific … del models

# The standard "official" OMG stack (mimicking OT)



M3 Layer: Specifies meta-metaclasses for the UML metamodel

M2 Layer: Specifies metaclasses for the UML metamodel, such as Class

M1 Layer: Specifies classes for the UML user models, such as Passenger, Ticket, TravelAgency

M0 Layer: User objects that are instances of UML user model classes, such as instances of Passenger, Ticket, TravelAgency

# The metamuddle

- A very rapidly growing industrial application field since november 2000,
- … but …
- We badly need a unifying theory of models

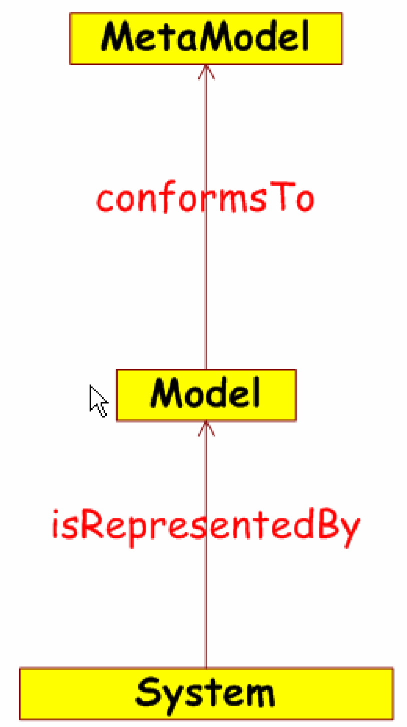
Agenda

# In search of sound principles for MDE

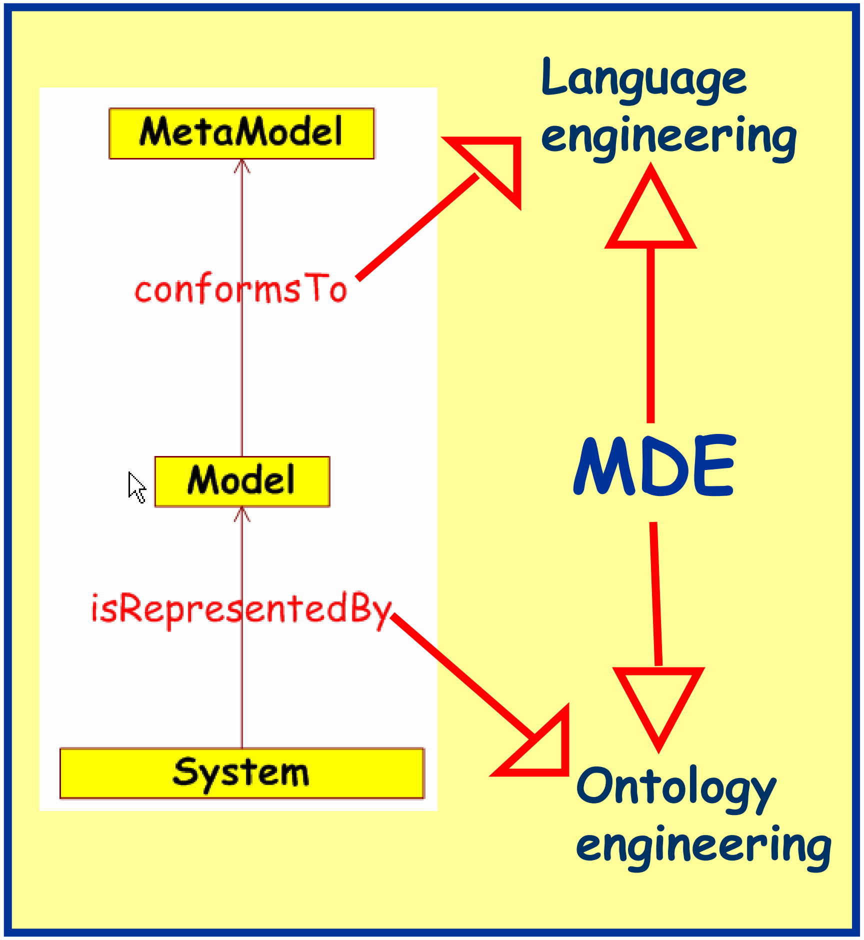
The basic assumptions

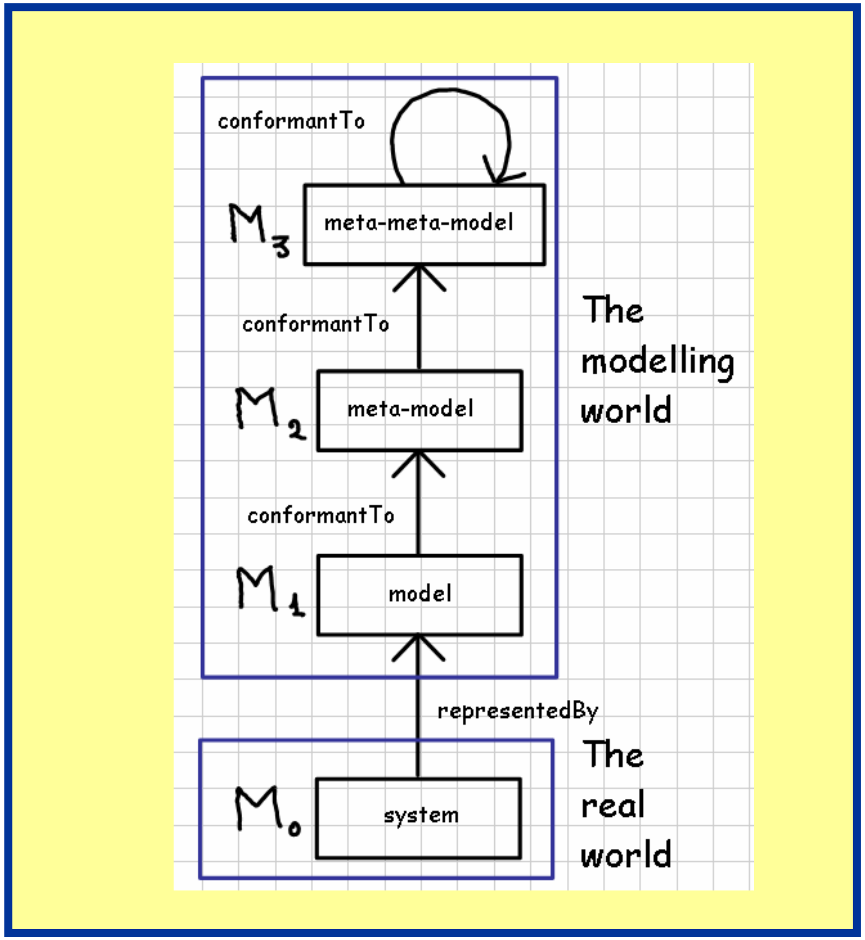
# The basic assumptions

- Models as first class entities
- MDA as a special case of MDE
- Conformance and Representation as kernel relations, central to MDE

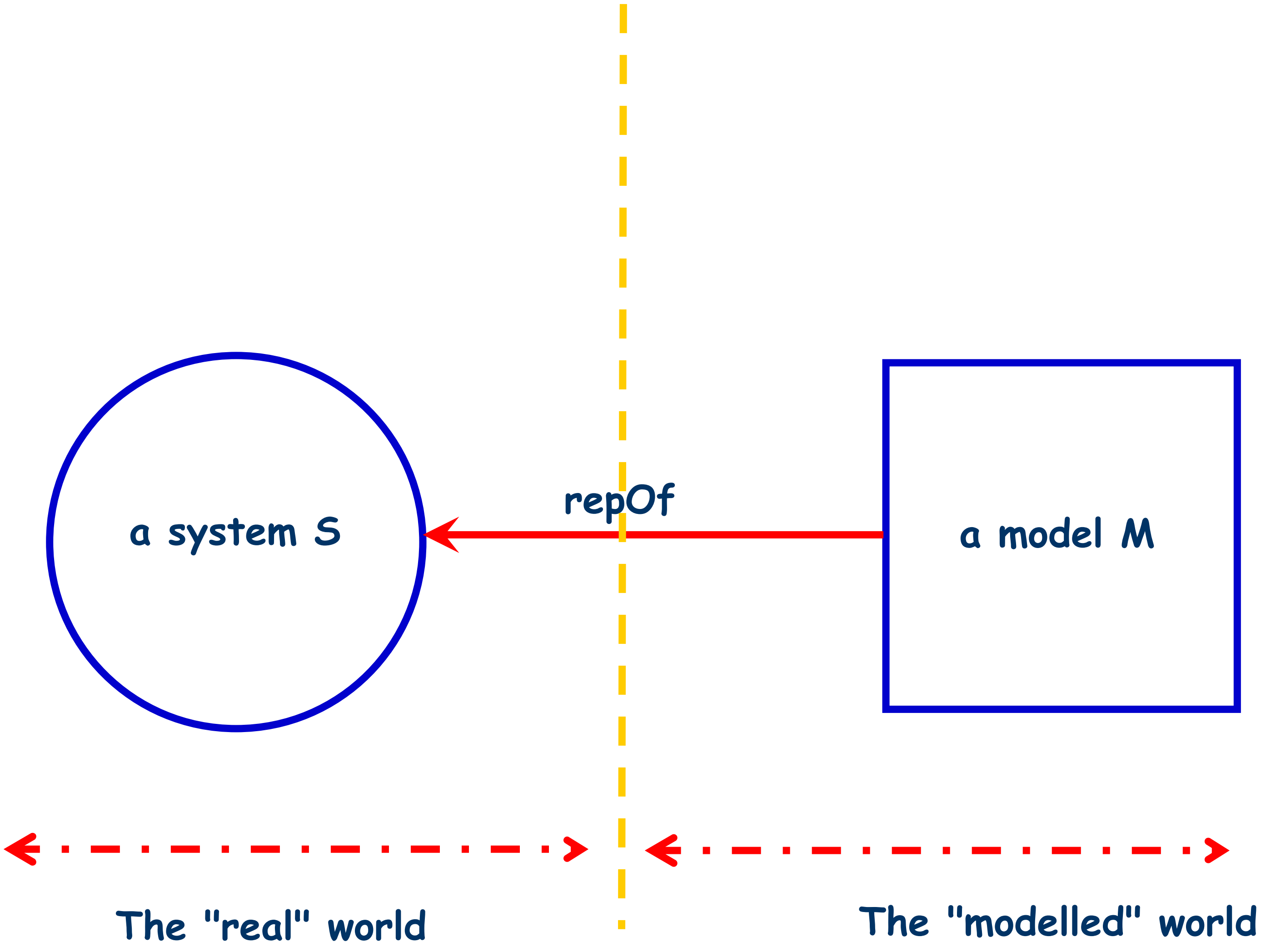


# Credits and MDA compliance





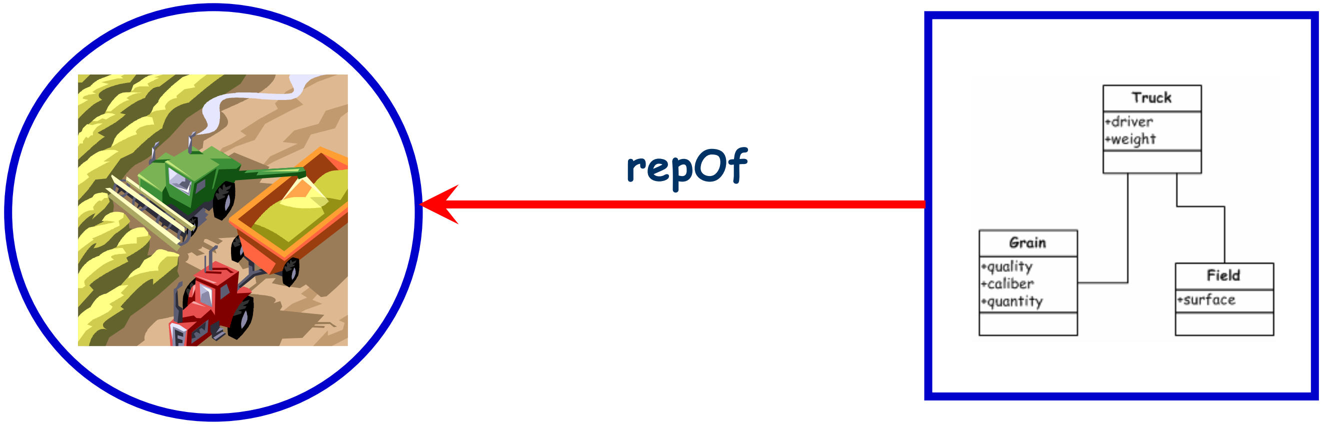
# Systems and models



# A model of a system

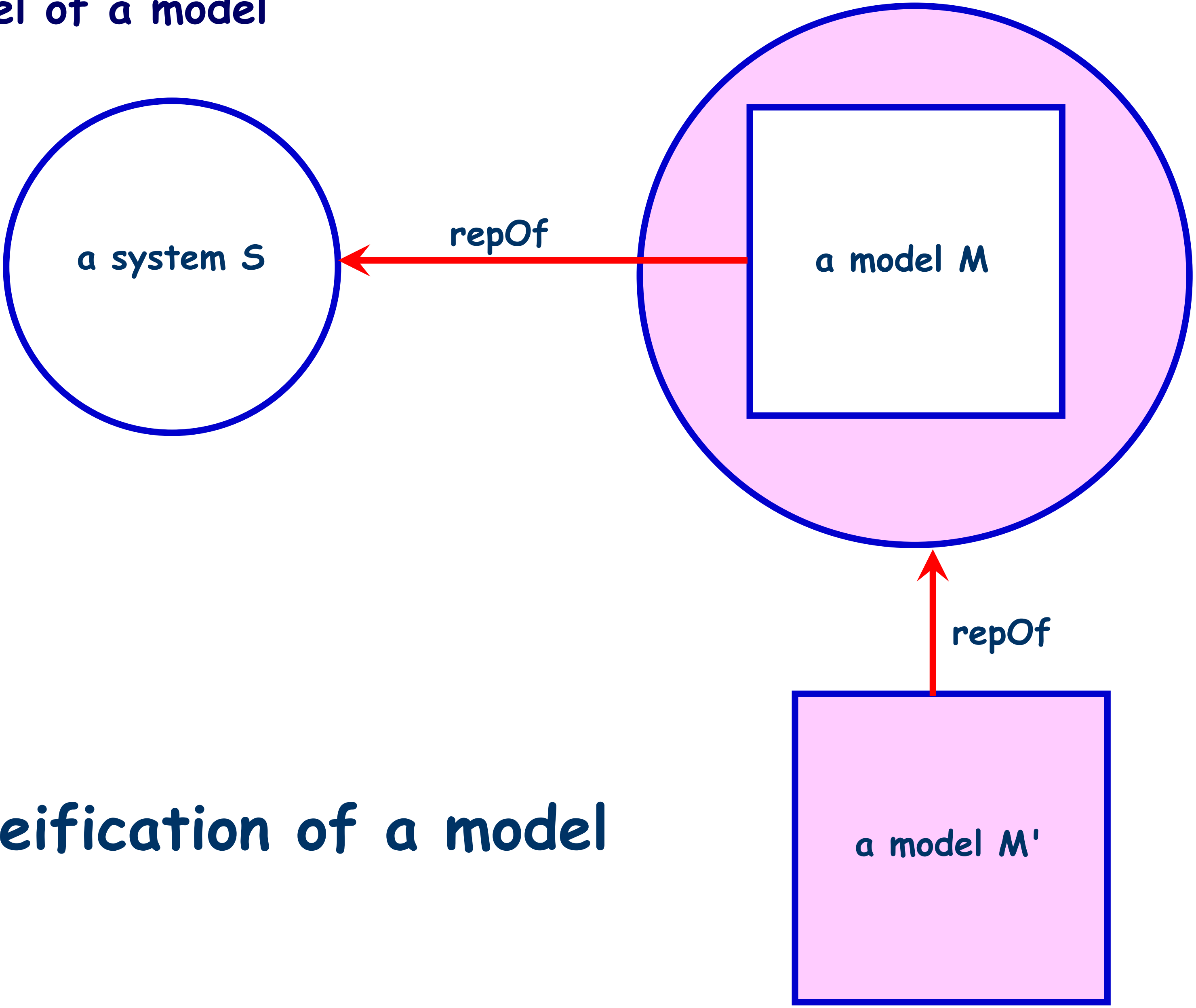
a system S
e.g. agricultural business

a model M



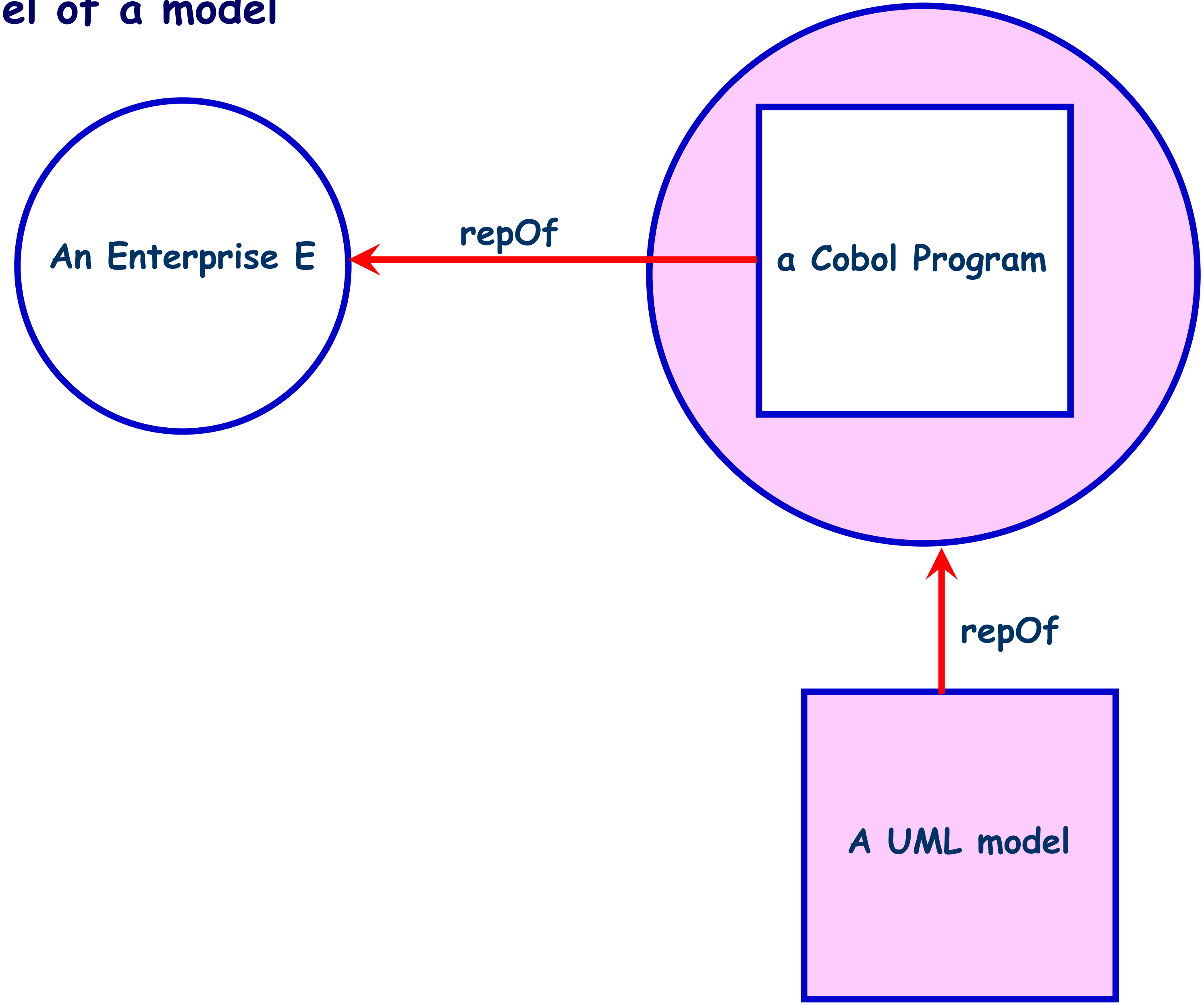
repOf

# A model of a model



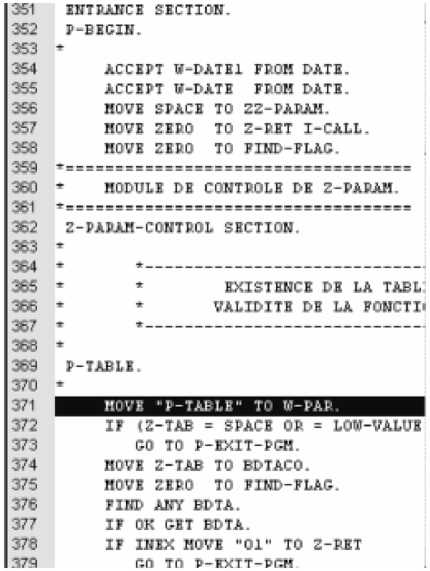
Reification of a model

# A model of a model



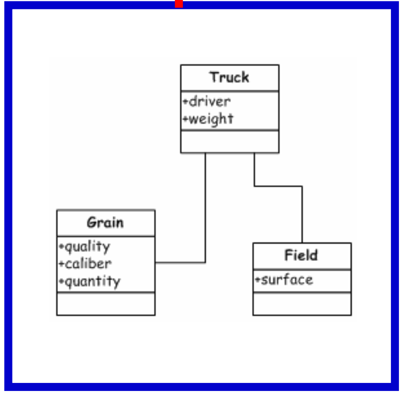
# A model of a model

repOf



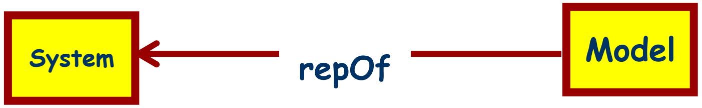
```
 ENTRANCE SECTION.
 P-BEGIN.
*
     ACCEPT W-DATE1 FROM DATE.
     ACCEPT W-DATE  FROM DATE.
     MOVE SPACE TO ZZ-PARAM.
     MOVE ZERO  TO Z-RET I-CALL.
     MOVE ZERO  TO FIND-FLAG.
*====================================
*    MODULE DE CONTROLE DE Z-PARAM.
*====================================
 Z-PARAM-CONTROL SECTION.
*
*       *-----------------------------
*       *         EXISTENCE DE LA TABL
*       *        VALIDITE DE LA FONCTI
*       *-----------------------------
*

 P-TABLE.
*
     MOVE "P-TABLE" TO W-PAR.
     IF (Z-TAB = SPACE OR = LOW-VALUE
        GO TO P-EXIT-PGM.
     MOVE Z-TAB TO BDTACO.
     MOVE ZERO  TO FIND-FLAG.
     FIND ANY BDTA.
     IF OK GET BDTA.
     IF INEX MOVE "01" TO Z-RET
        GO TO P-EXIT-PGM.
```

repOf

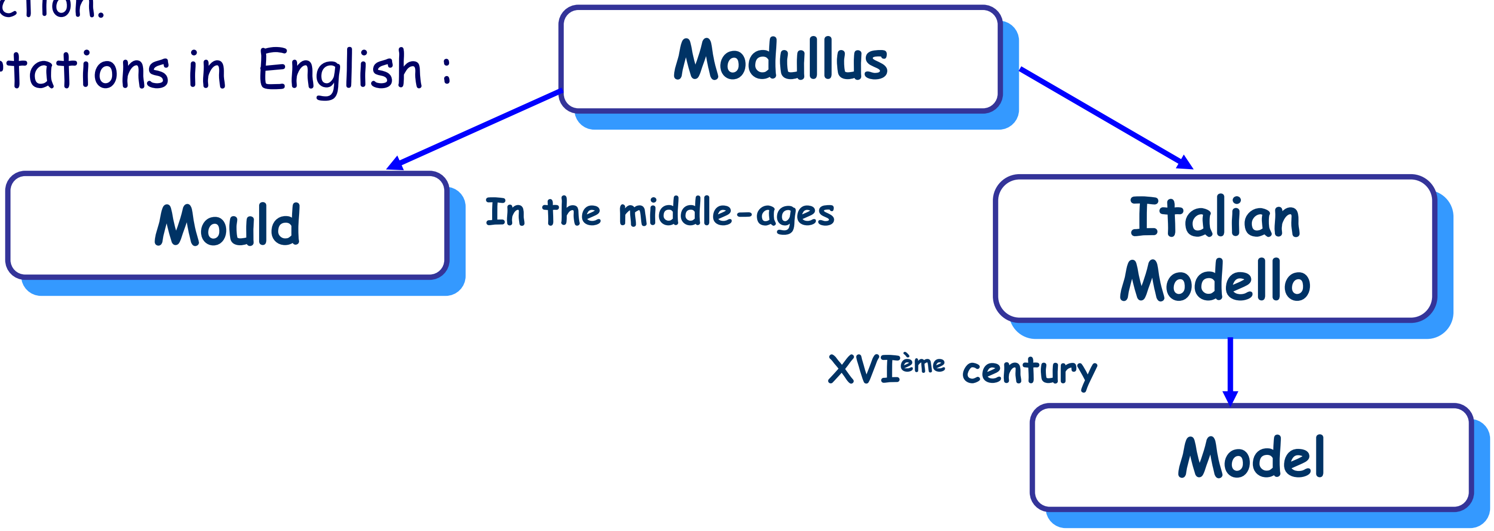


But- what is a model?

# Definitions



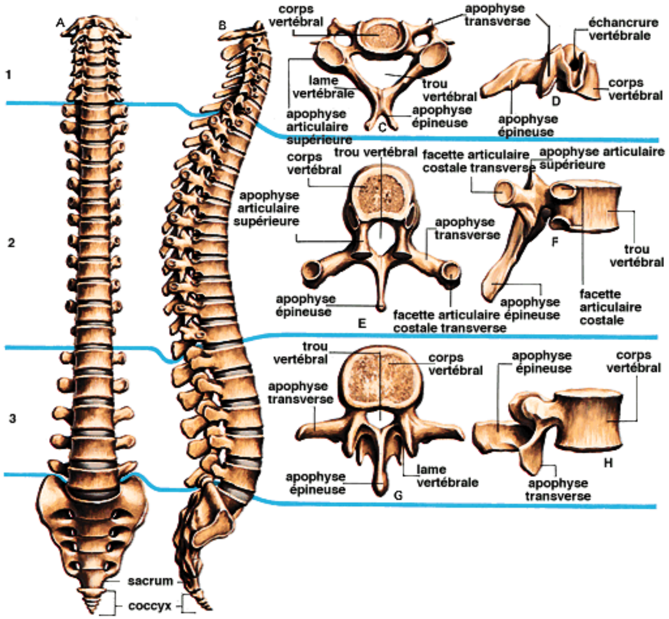
- A model is the simplified image of a system
  - This short definition should be completed
- What is a system ?
  - "A system is a set of elements in interaction " (von Bertalanffy)
  - The word system comes from the Greek "sun-istémi" (I compose)
- Model comes from the Latin "modullus", diminutive of "modus" (measure)
  - Initially it was an architectural term meaning an arbitrary measure used for establishing various ratios between different parts of a building in construction.
- Two importations in English :

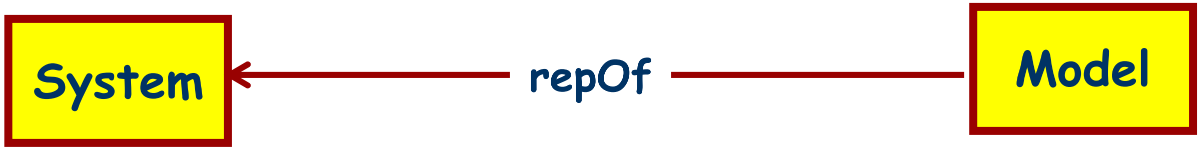


# The word is recent, the idea is old

Plato (427-347 before JC), in *Timeus* compares vertebras to door hinges (74a) or blood vessels to irrigation channels.

This idea will be used again later by the english physiologist William Harvey (1578-1657) who will discover the blood circulation principle:
"de ce que, dans le cœur des vivants,
les valvules semblent être des soupapes ou des portes d'écluse".





System ← repOf — Model

# Model: multiple definitions

## MSN Encarta

**mod·el** [módd'l] noun (plural **mod·els**)

**1. copy of an object:** a copy of an object, especially one made on a smaller scale than the original ( *often used before a noun* )

**2. particular version of manufactured article:** a particular version of a manufactured article
*had traded in her car for the latest model*

**3. something copied:** something that is copied or used as the basis for a related idea, process, or system

**4. somebody paid to wear clothes:** somebody who is paid to wear clothes and demonstrate merchandise as a profession, for example, in fashion shows and photographs for magazines and catalogues

**5. simplified version:** a simplified version of something complex used, for example, to analyze and solve problems or make predictions *a financial model*

**6. perfect example:** an excellent example that deserves to be imitated

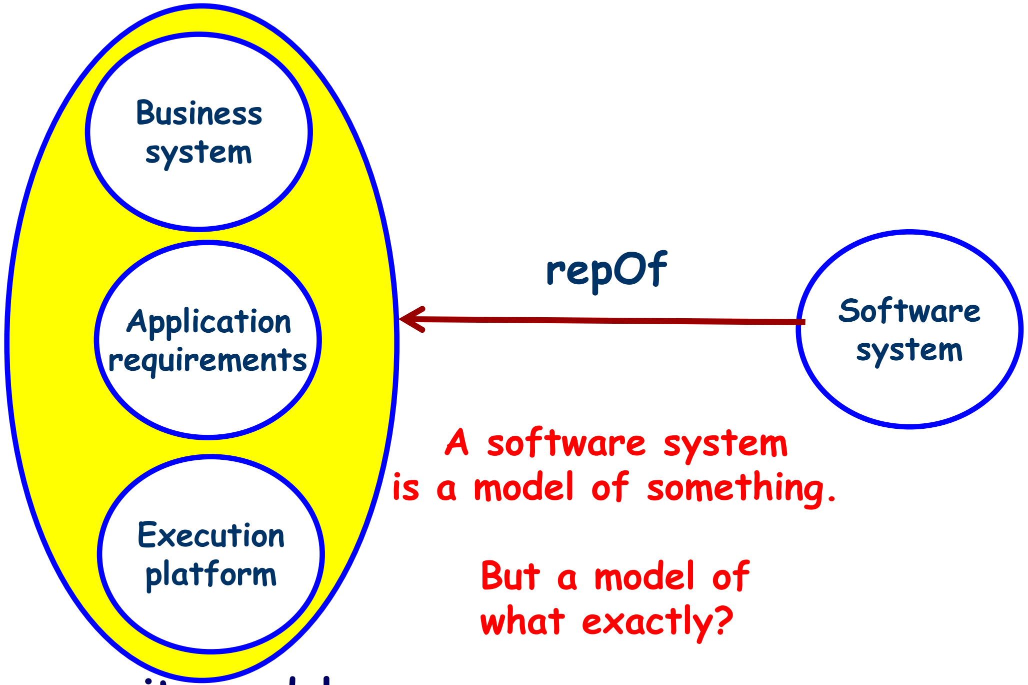
**7. artist's subject:** somebody who poses for a painter, sculptor, photographer, or other artist

**8.** zoology **animal copied by another animal:** an animal species repellent to predators which another animal mimics for protection

**9.** logic **interpretation:** an interpretation of a theory arrived at by assigning referents in such a way as to make the theory true

**10.** *U.K.* fashion **exclusive garment:** the first sewn example of a couturier's or clothing manufacturer's design, from which a new line of garments is produced

# Differents kinds of models



A software system is a model of something.

But a model of what exactly?

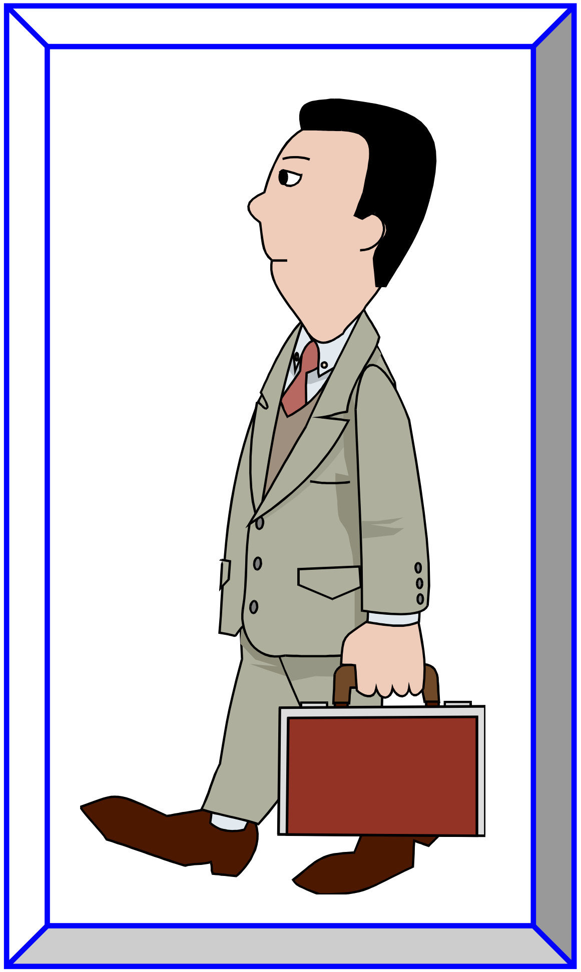
- Numerous examples
  - Mathematical models
  - Hydrological models
  - Biological models
  - Ecological models
  - Economical models
  - Meteorological models
  - Simulation models
  - Descriptive or predictive models
  - etc.
- Software are models:
  - **A software is a complex and composite model**
  - **But a model of what?**
    - Of the organization where it is supposed to function?
    - Of the architecture (hardware/software platform ) on top of which it is supposed to function?
    - Of the applicative requirements it is supposed to satisfy ?
    - Of the team that elaborated the software in a given process?
  - **Expressed in which language?**
    - Until now software was mainly written in so-called programming languages like C# or Java
    - … but things are rapidly changing (code-centric to model-centric, DSLs)

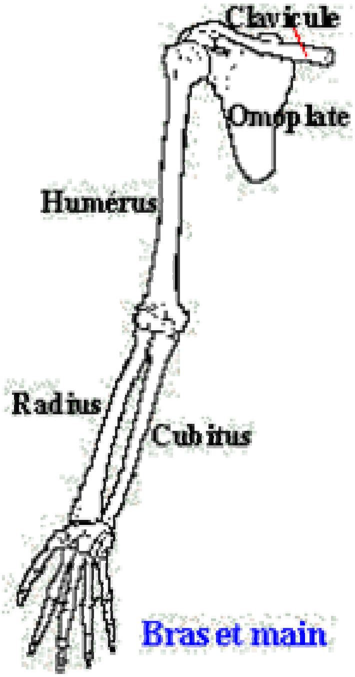
# A model is a view on a system

A system

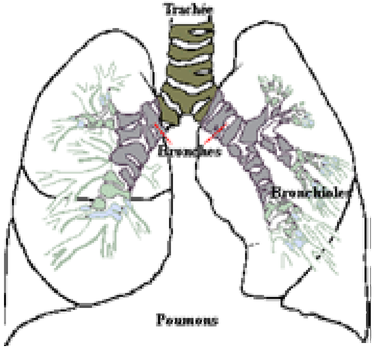
repOf

Several models of this system (partial views)





Skeleton model

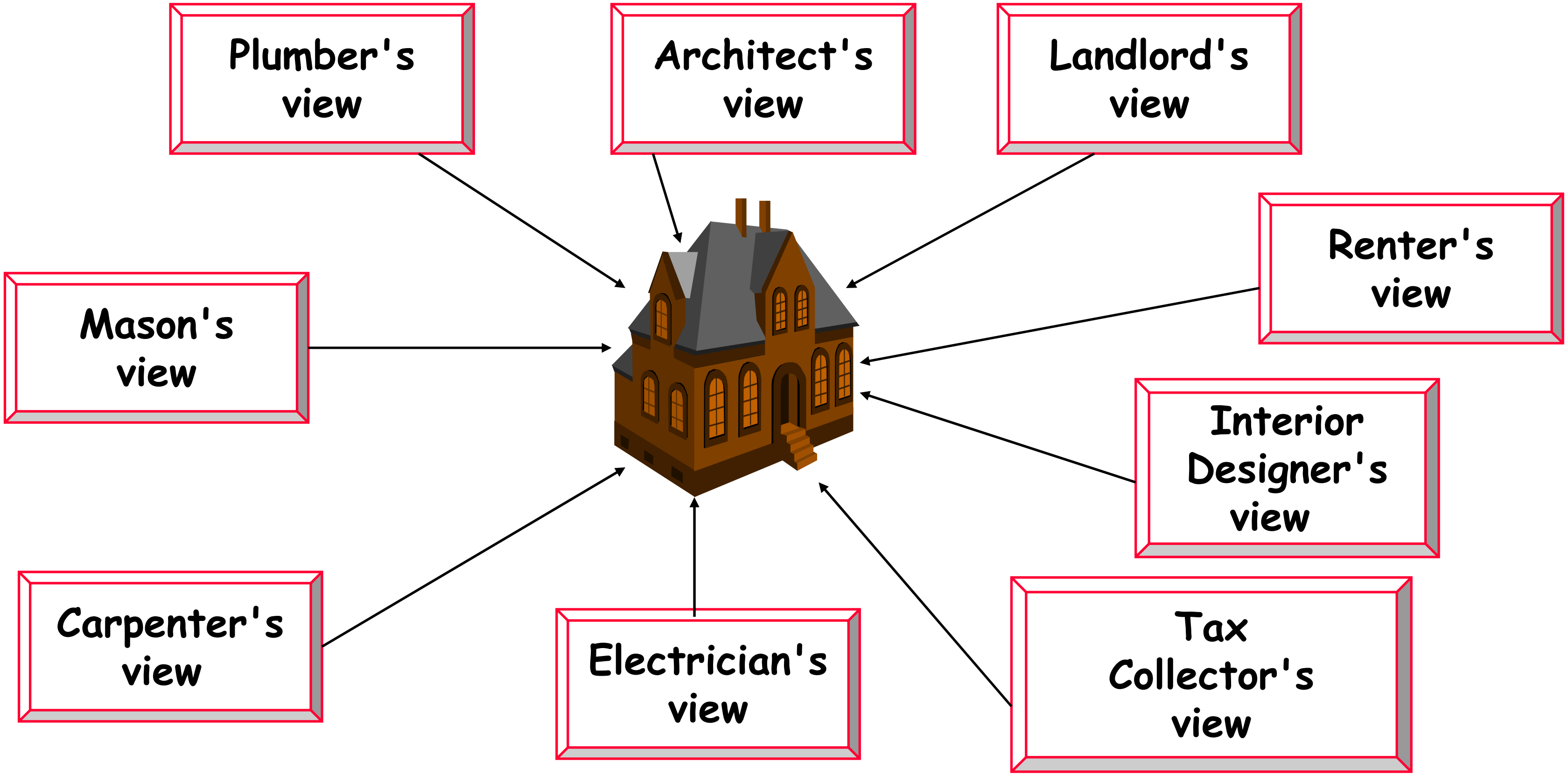


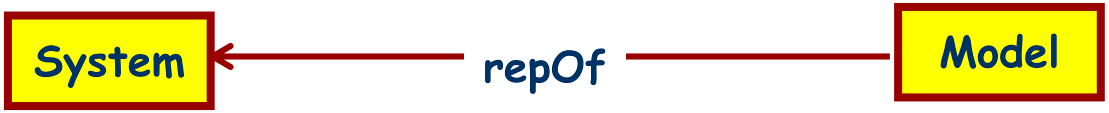
Respiratory model

Other Models muscular, nervous, circulatory, digestive, endocrinous, etc.

# Multiple views and coordinated DSLs

Each view is expressed in a given domain language;
Vocabularies of different corporations are different;
However they allow talking about a common building.



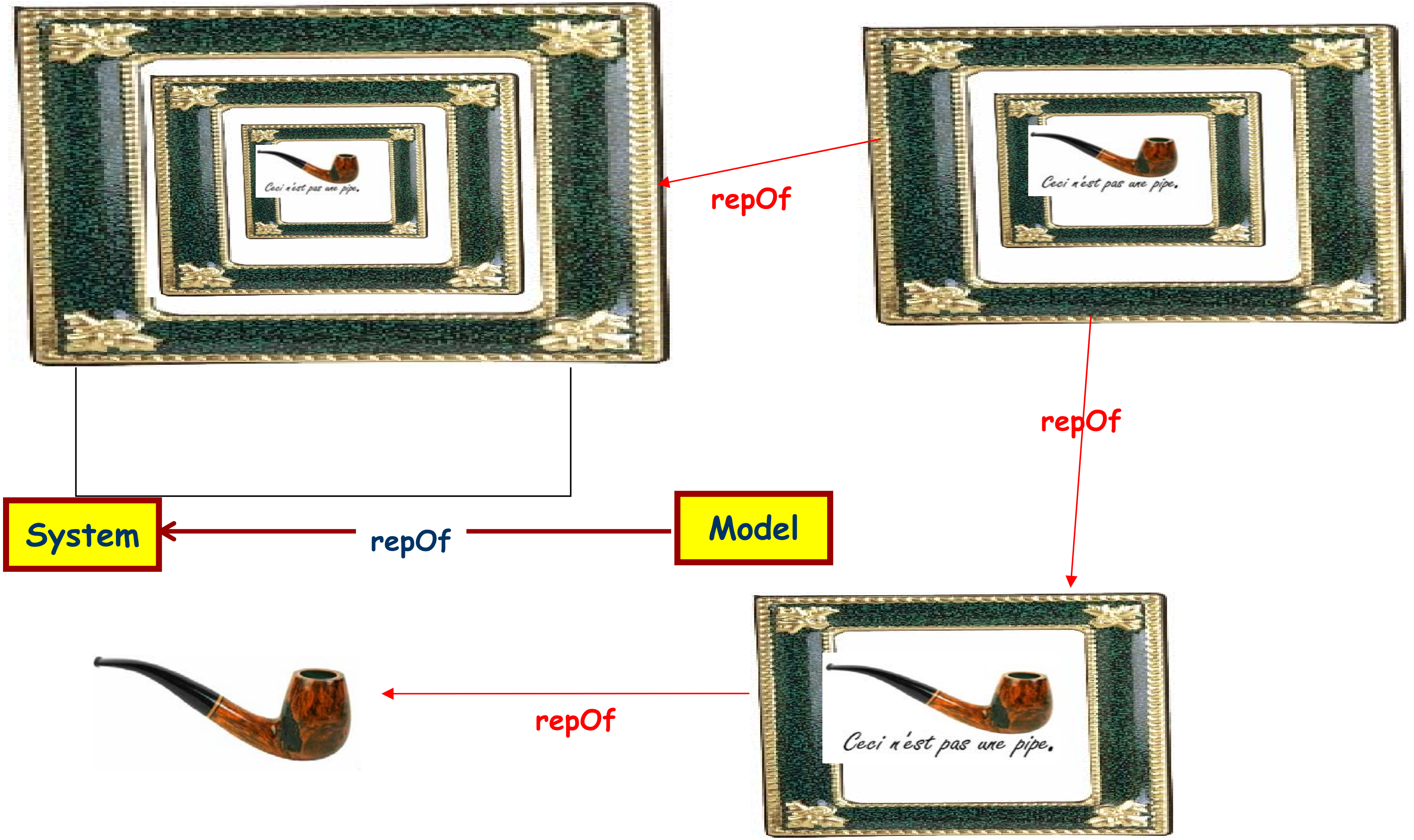


## Don't confuse the model and the system

# This is not a pipe by Magritte

Ceci n'est pas une pipe.

# Don't confuse the model and the system



repOf

repOf

System

repOf

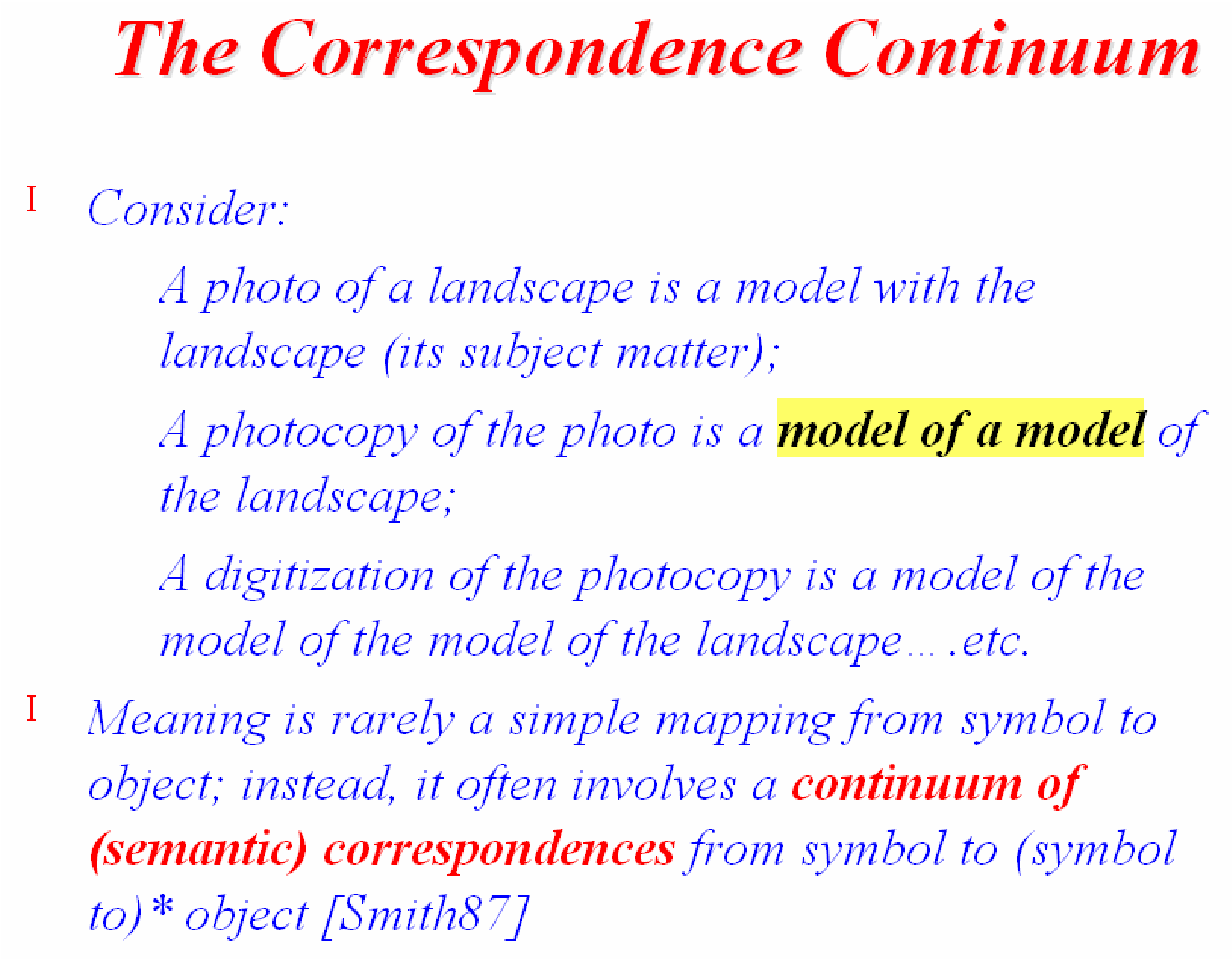
Model

repOf

The system

A model

# Model of a model



## *The Correspondence Continuum*

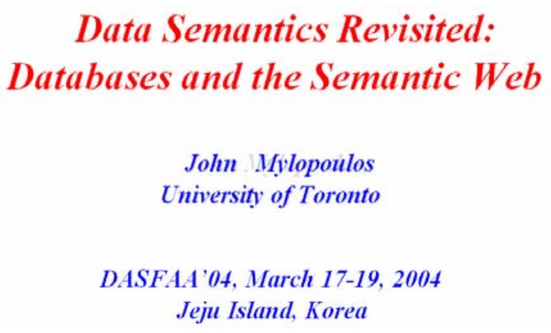
I *Consider:*

*A photo of a landscape is a model with the landscape (its subject matter);*

*A photocopy of the photo is a* ***model of a model*** *of the landscape;*

*A digitization of the photocopy is a model of the model of the model of the landscape….etc.*

I *Meaning is rarely a simple mapping from symbol to object; instead, it often involves a* ***continuum of (semantic) correspondences*** *from symbol to (symbol to)* object [Smith87]*

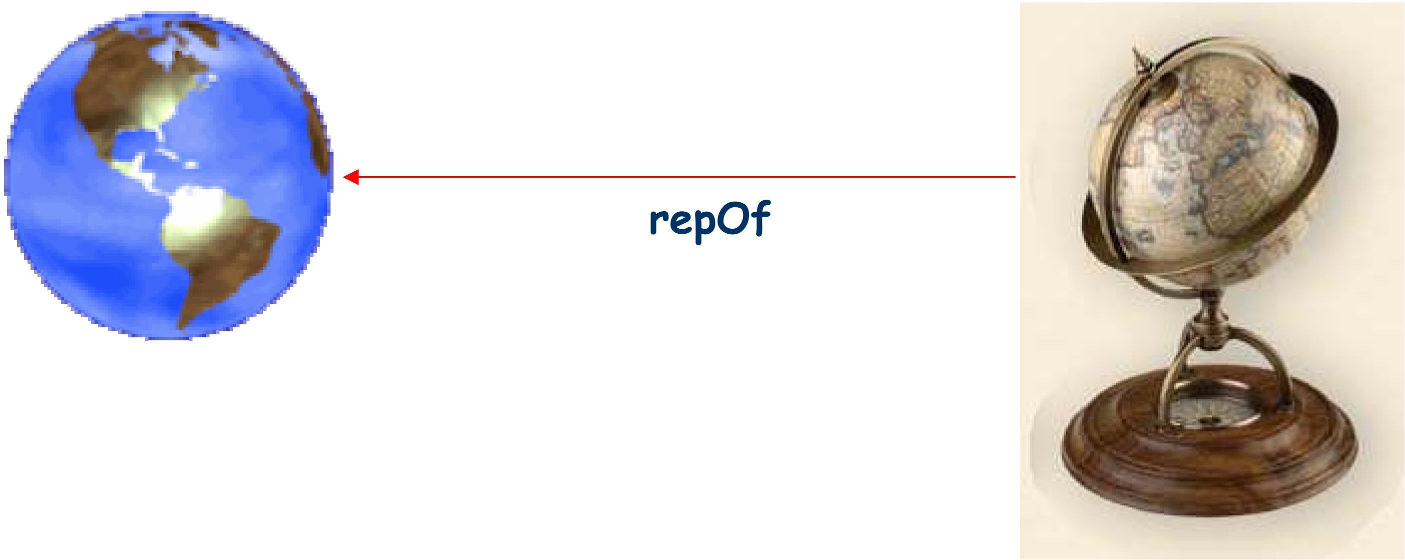


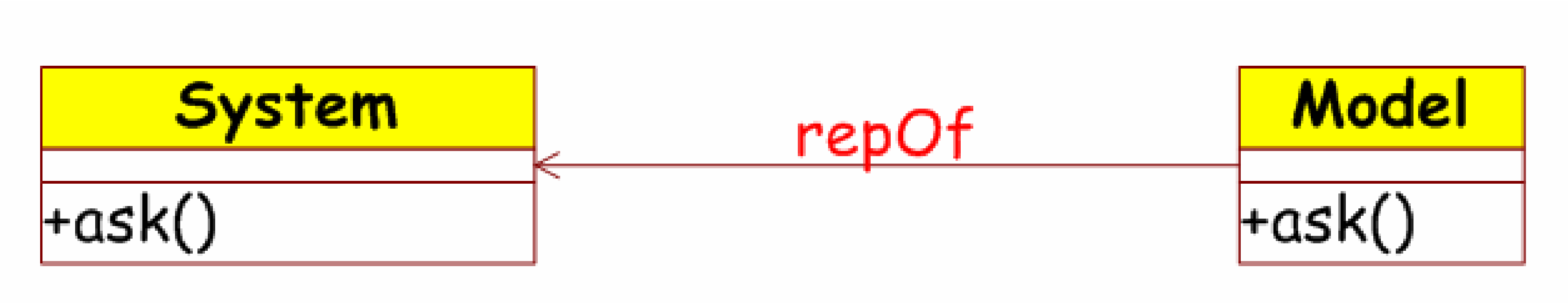
***Data Semantics Revisited: Databases and the Semantic Web***

*John Mylopoulos*
*University of Toronto*

*DASFAA'04, March 17-19, 2004*
*Jeju Island, Korea*

# The globe is a model of the earth





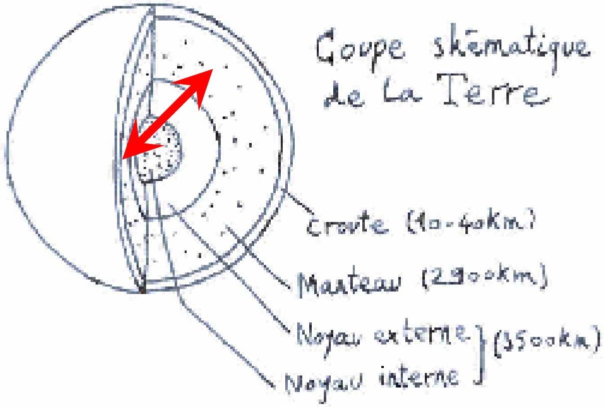
# The globe is a model of the earth

- Allowing to ask certain questions …

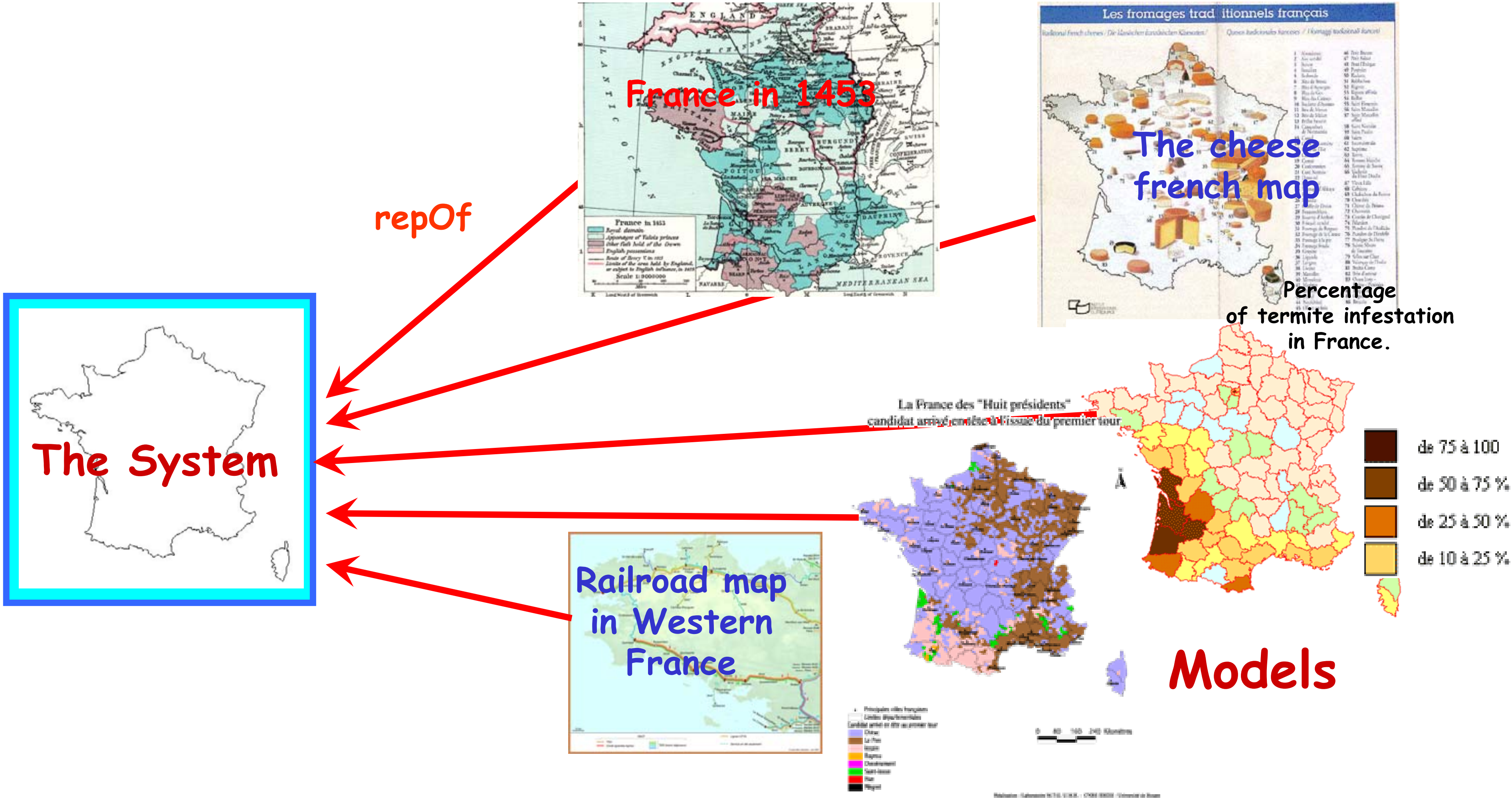
Could I walk from Brest to San Francisco without using a boat?

- But not others

What is the temperature at the bottom if a dig a 100 km deep hole at the surface of the earth ?



# A very popular model: geographical maps


Les fromages traditionnels français
La France des "Huit présidents"
candidat arrivé en tête à l'issue du premier tour
de 75 à 100
de 50 à 75 %
de 25 à 50 %
de 10 à 25 %
France in 1453
ENGLAND
ATLANTIC OCEAN
MEDITERRANEAN SEA

France in 1453

repOf

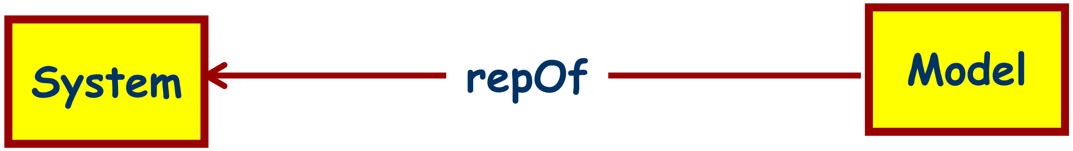
The cheese french map

Percentage of termite infestation in France.

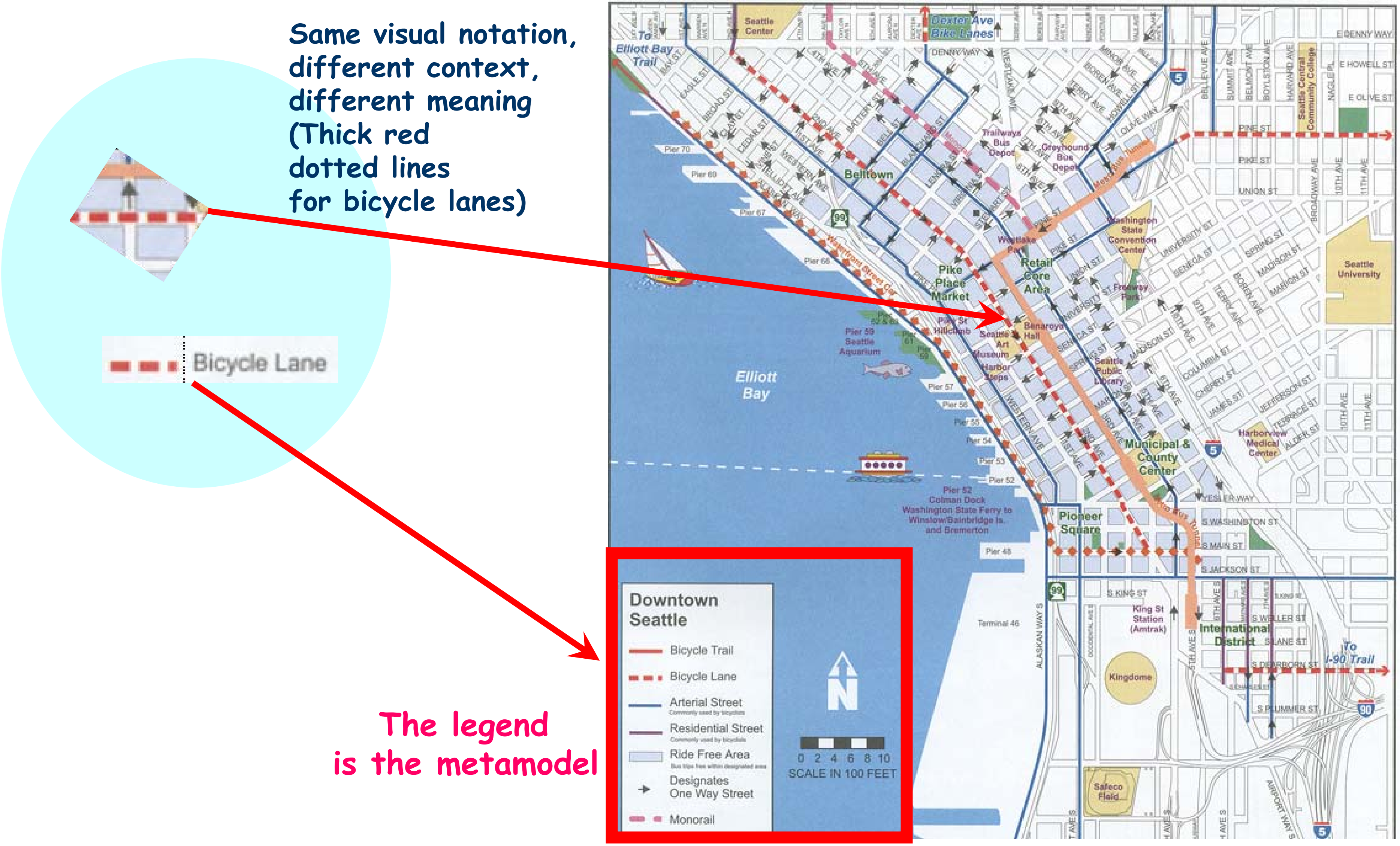
The System

Railroad map in Western France

Models



# Every map has a legend (implicit or explicit)



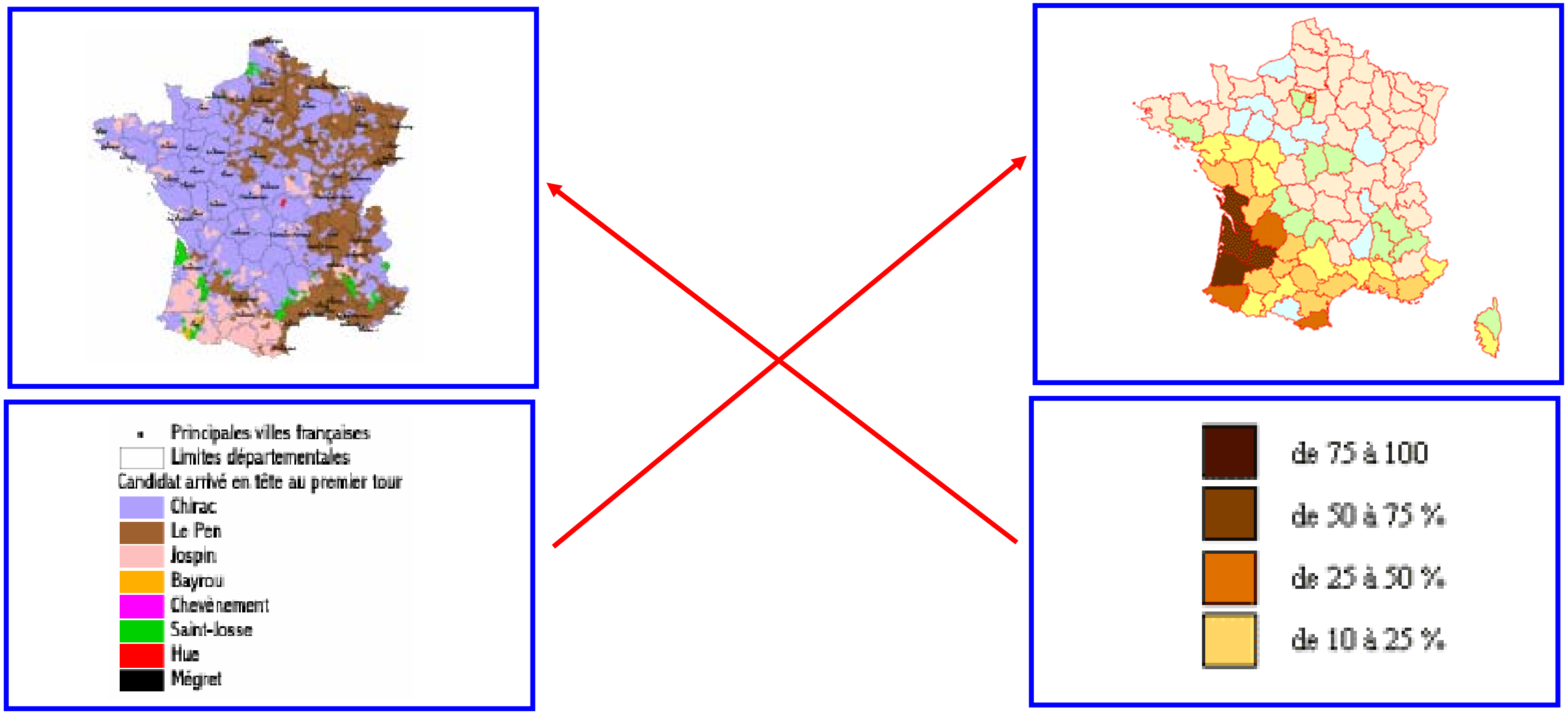
Same visual notation, different context, different meaning (Thick red dotted lines for bicycle lanes)

The legend is the metamodel

# a Model has no meaning when separated from its metamodel

First round of political election in France in 2002.

Percentage of places infested by termites in France.



- Principales villes françaises

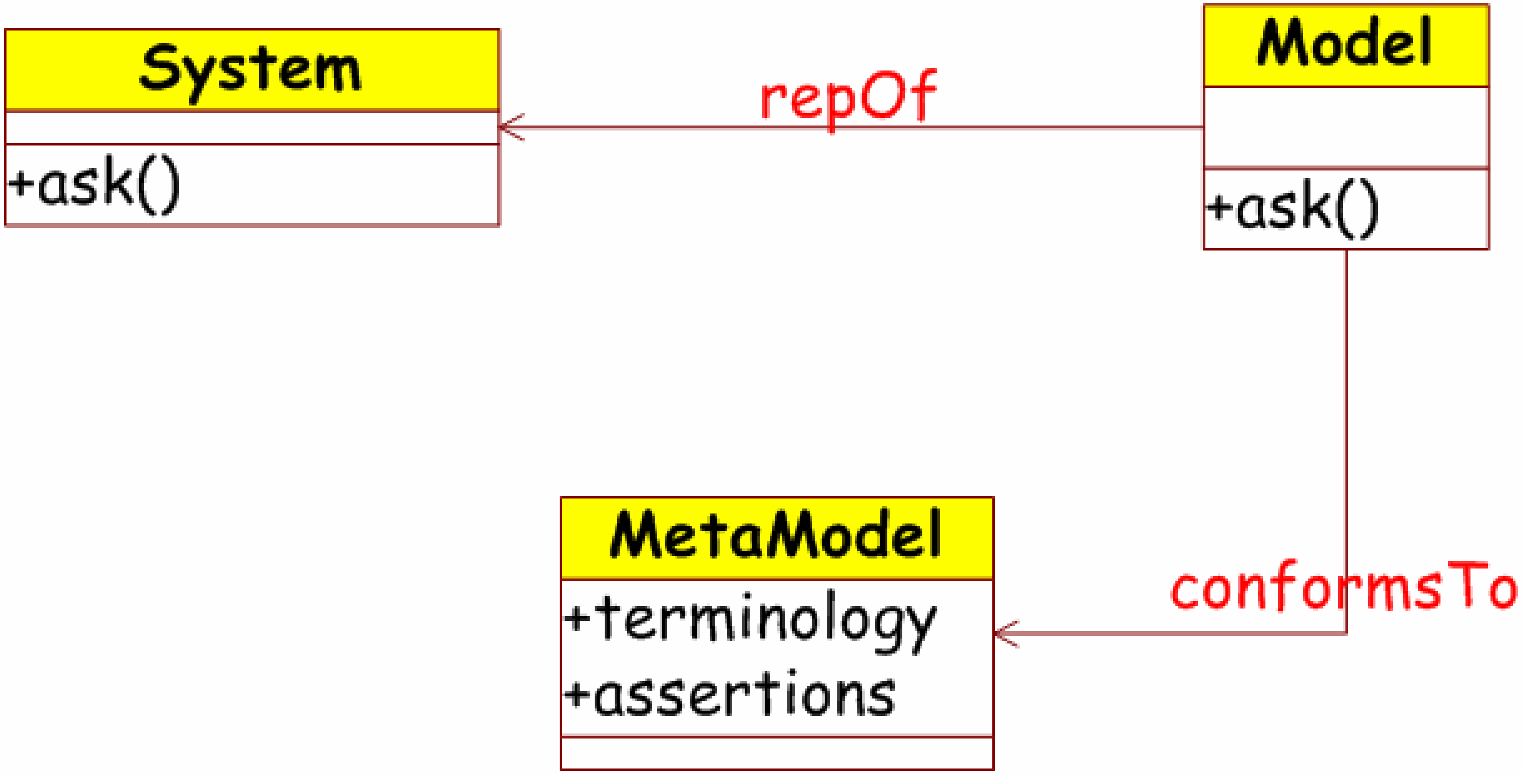
□ Limites départementales

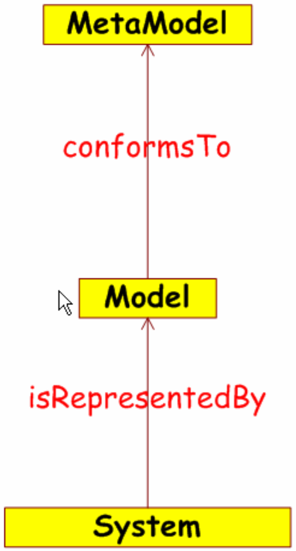
Candidat arrivé en tête au premier tour

| | |
|---|---|
| 🟪 | Chirac |
| 🟫 | Le Pen |
| ▨ | Jospin |
| 🟧 | Bayrou |
| 🟪 | Chevènement |
| 🟩 | Saint-Josse |
| 🟥 | Hue |
| ■ | Mégret |

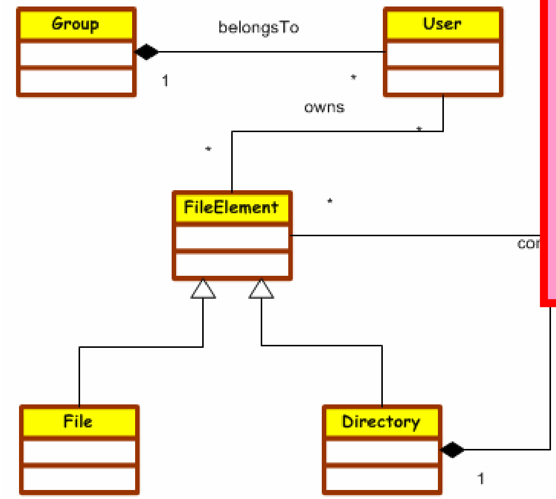
| | |
|---|---|
| 🟫 | de 75 à 100 |
| 🟫 | de 50 à 75 % |
| 🟧 | de 25 à 50 % |
| 🟨 | de 10 à 25 % |

# The legend is a metamodel





```
class(Group);
class(User);
class(FileElement);
class (File);
class (Directory);
association(belongsTo,User*,Group)
association(owns,User,FileElement*)
association(contains,Directory,FileElement*)
inherits(File,FileElement);
inherits(Directory,FileElement);
```



```
meta(Student,Group);
meta(Teacher,Group);
meta(Bob,User);
meta(Jim,User);
meta(Esther,User);
meta(Mary,User);
meta(F1,File);
meta(F2,File);
meta(F3,File);
meta(F4,File);
meta(F5,File);
meta(D1,Directory);
meta(D2,Directory);
meta(D3,Directory);
```

```
belongsTo(Esther,Teacher);
belongsTo(Mary,Teacher);
belongsTo(Jim,Student);
belongsTo(Bob,Student);
owns(Esther,D1);
owns(Esther,F1);
contains(D1,F1);
owns(Mary,D2);
owns(Mary,F2);
contains(D2,F2);
owns(Bob,D3);
owns(Bob,F3);
contains(D3,F3);
contains(D2,D3);
owns(Jim,F4);
owns(Jim,F5);
contains(D1,F4);
contains(D2,F5);
```

The System S

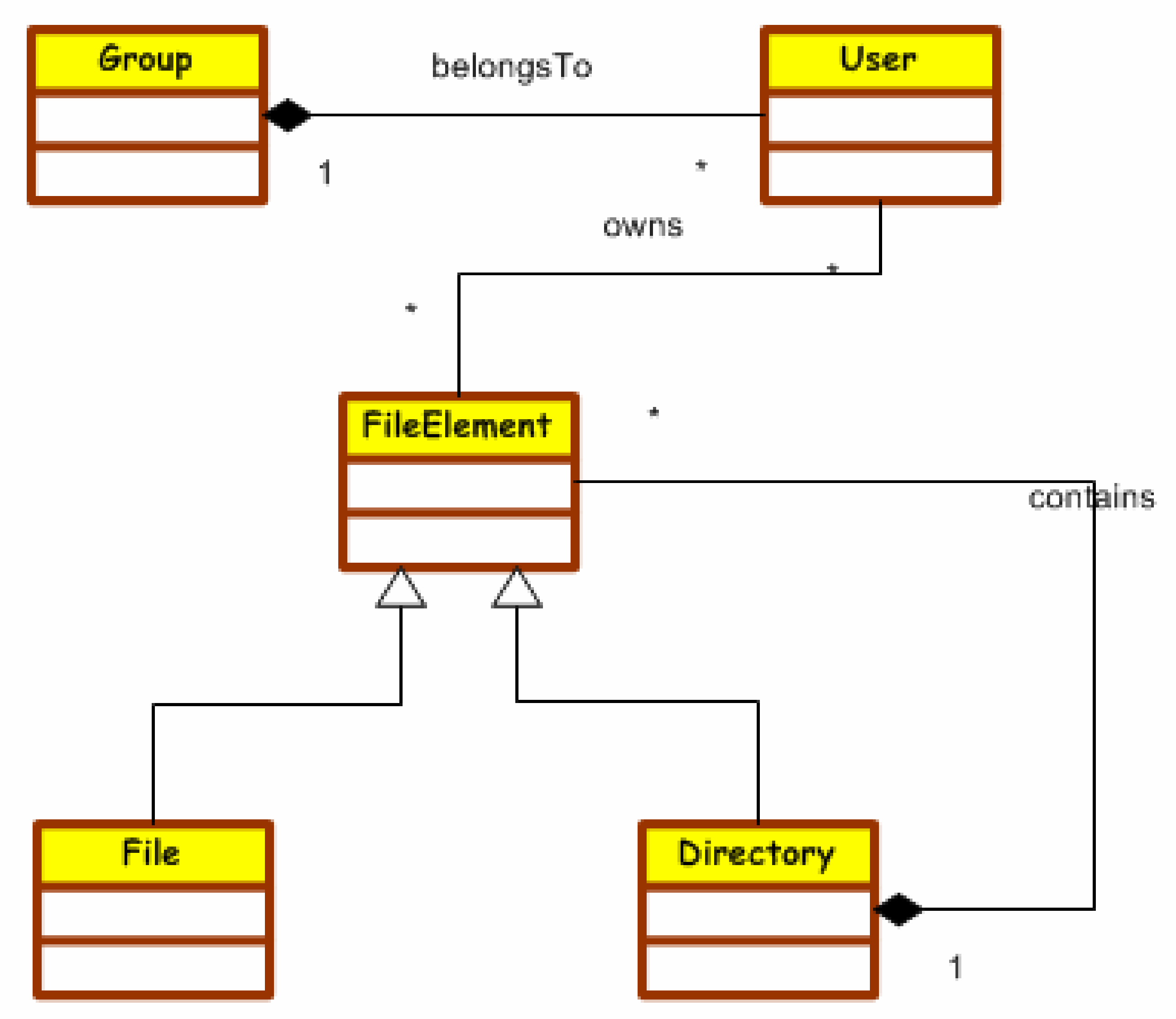
The Metamodel

A Model

# The metamodel

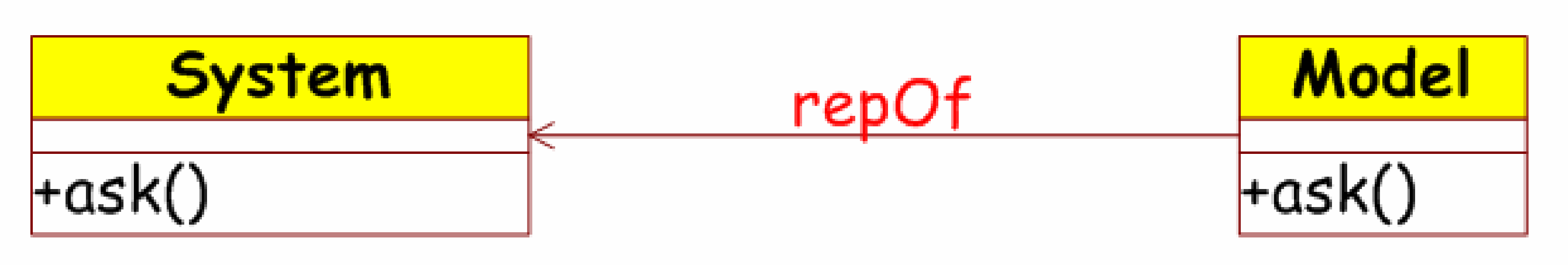
```
class (Group);
class (User);
class (FileElement);
class (File);
class (Directory);
association (belongsTo, User*, Group)
association (owns,User, FileElement*)
association (contains, Directory, FileElement*)
inherits (File, FileElement);
inherits (Directory, FileElement);
```

# The metamodel (another representation)



# Limited Substitutability Principle

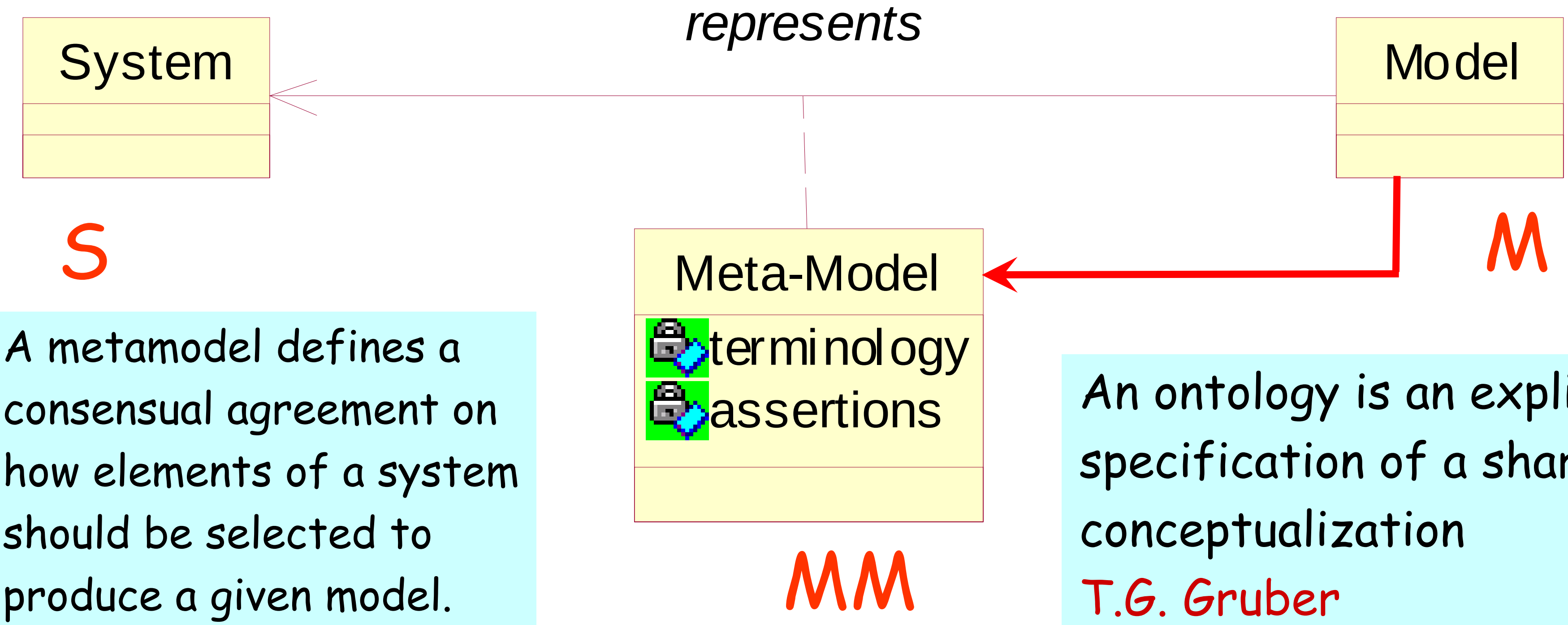
- The purpose of a model is always to be able to answer some specific sets of questions in place of the system, exactly in the same way the system itself would have answered similar questions.



- *A* model represents certain specific aspects of a system and only these aspects, for a specific purpose.

# Metamodels as simple ontologies

- Metamodels as precise abstraction filters.
- Each metamodel defines a DSL.
- Each metamodel is used to specify which particular "aspect" of a system should be considered to constitute the model.

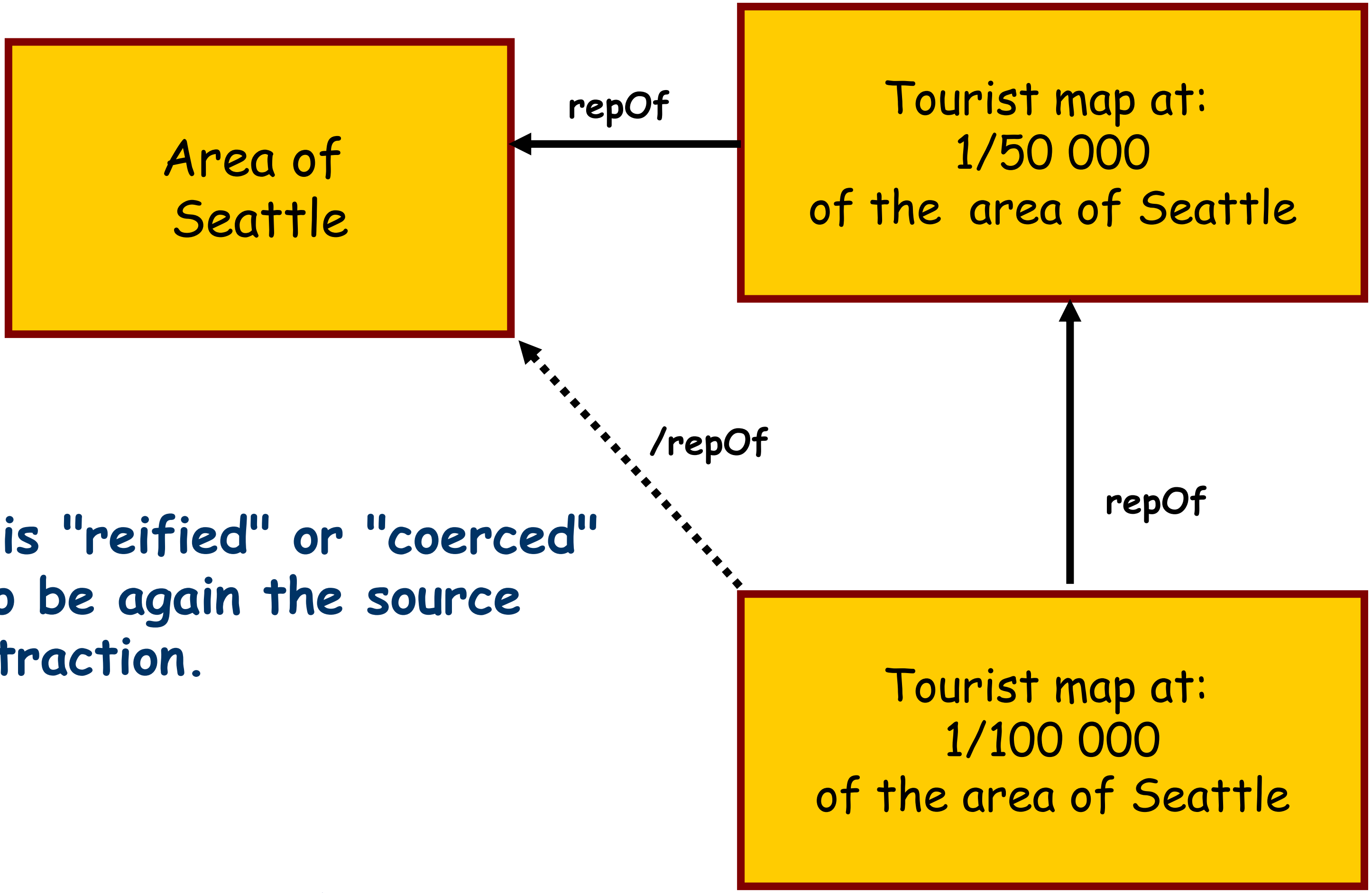


A metamodel defines a consensual agreement on how elements of a system should be selected to produce a given model.

An ontology is an explicit specification of a shared conceptualization
T.G. Gruber

The correspondence between a system and a model is precisely and computationally defined by a metamodel.

# The model of a model is not a metamodel


Area of Seattle
repOf
Tourist map at: 1/50 000 of the area of Seattle
/repOf
repOf
Tourist map at: 1/100 000 of the area of Seattle

Here a model is "reified" or "coerced" as a system to be again the source of a model extraction.

A map may represent a given territory with a certain perspective.
It has a certain scale, for example the map A of the Seattle area at 1/50 000.
If we take a map of the same area at a different scale, this may be considered either as a map B of the same Seattle territory or alternatively as a model of map A.
As illustrated, a model of a model may be thus apparently be a model.

# Lewis Carroll and the 1:1 map

“That’s another thing we’ve learned from *your* Nation,” said Mein Herr, “map-making. But we’ve carried it much further than you. What do you consider the *largest* map that would be really useful?”

“About six inches to the mile.”

“Only *six inches!”* exclaimed Mein Herr. “We very soon got to six *yards* to the mile. Then we tried a *hundred* yards to the mile. And then came the grandest idea of all! We actually made a map of the country, on the scale of *a mile to the mile!”*

"Have you used it much?” I enquired.

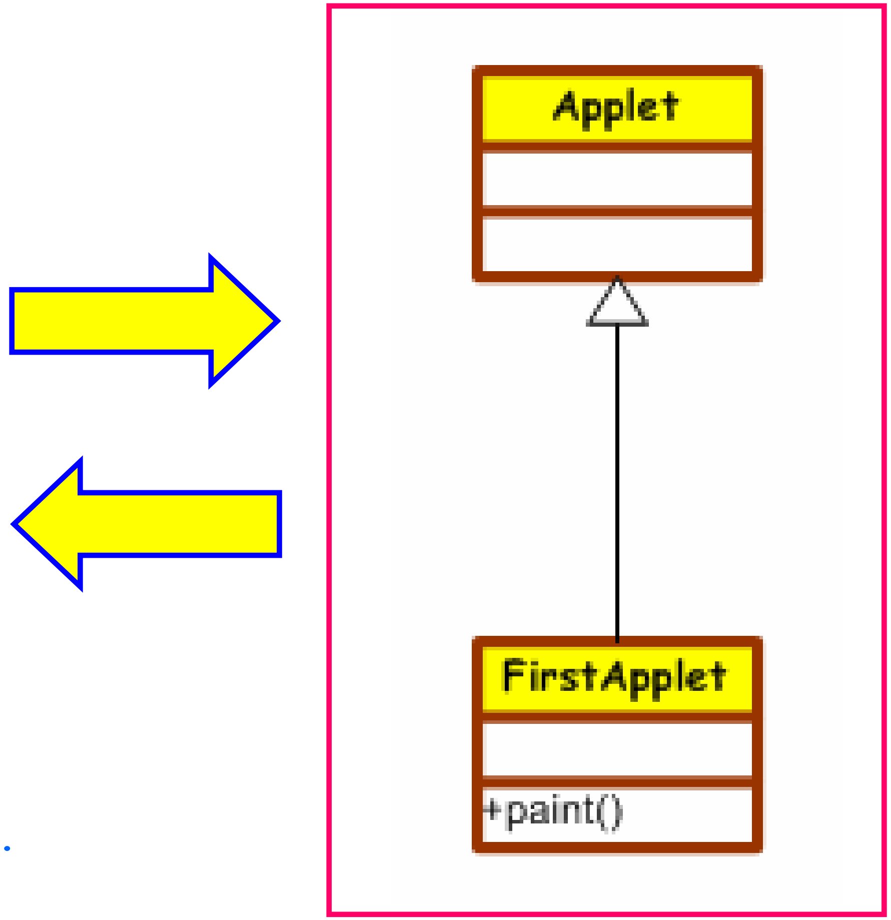
“It has never been spread out, yet,” said Mein Herr: “the farmers objected: they said it would cover the whole country, and shut out the sunlight! So we now use the country itself, as its own map, and I assure you it does nearly as well."

Lewis Carroll, *Sylvie and Bruno concluded* (London, 1893)

See also J.-L. Borges, J. François, and more recently Umberto Eco.

# UML is not a graphical syntax for C++ or Java

```
import java.applet.*;
import java.awt.*;

public class FirstApplet extends Applet {
    public void paint (Graphics g) {
        g.drawString("Hello World",25, 50);
    }
}
```



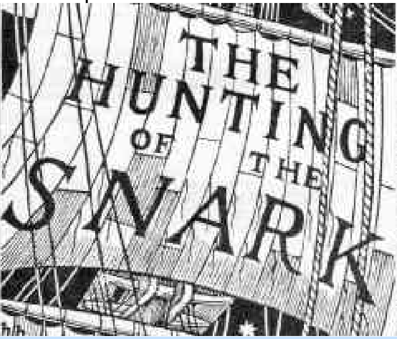
There is no one to one canonical correspondence.

# Lewis Carroll and the blank map

He had bought a large map representing the sea,
Without the least vestige of land:
And the crew were much pleased when they found it to be
A map they could all understand.

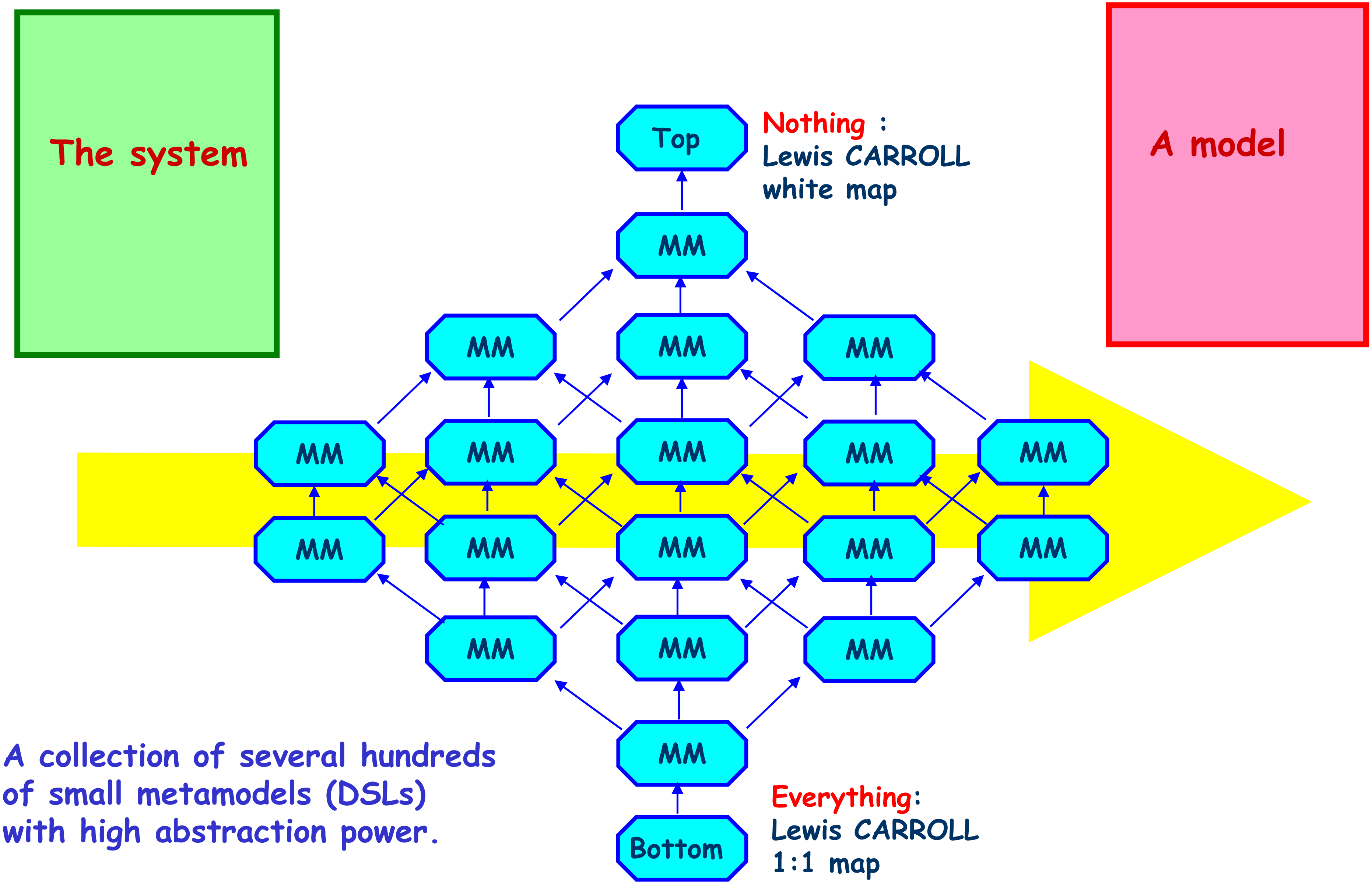
"What's the good of Mercator's North Poles and Equators,
Tropics, Zones, and Meridian Lines?"
So the Bellman would cry: and the crew would reply
"They are merely conventional signs!

"Other maps are such shapes, with their islands and capes!
But we've got our brave Captain to thank:
(So the crew would protest) "that he's bought us the best--
A perfect and absolute blank!"



THE HUNTING OF THE SNARK
an Agony in Eight Fits by Lewis Carroll

# A "lattice" of metamodels



Nothing :
Lewis CARROLL
white map

Everything:
Lewis CARROLL
1:1 map

A collection of several hundreds
of small metamodels (DSLs)
with high abstraction power.

## Small is beautiful

"Inside every large metamodel
is a small metamodel
struggling to get out"

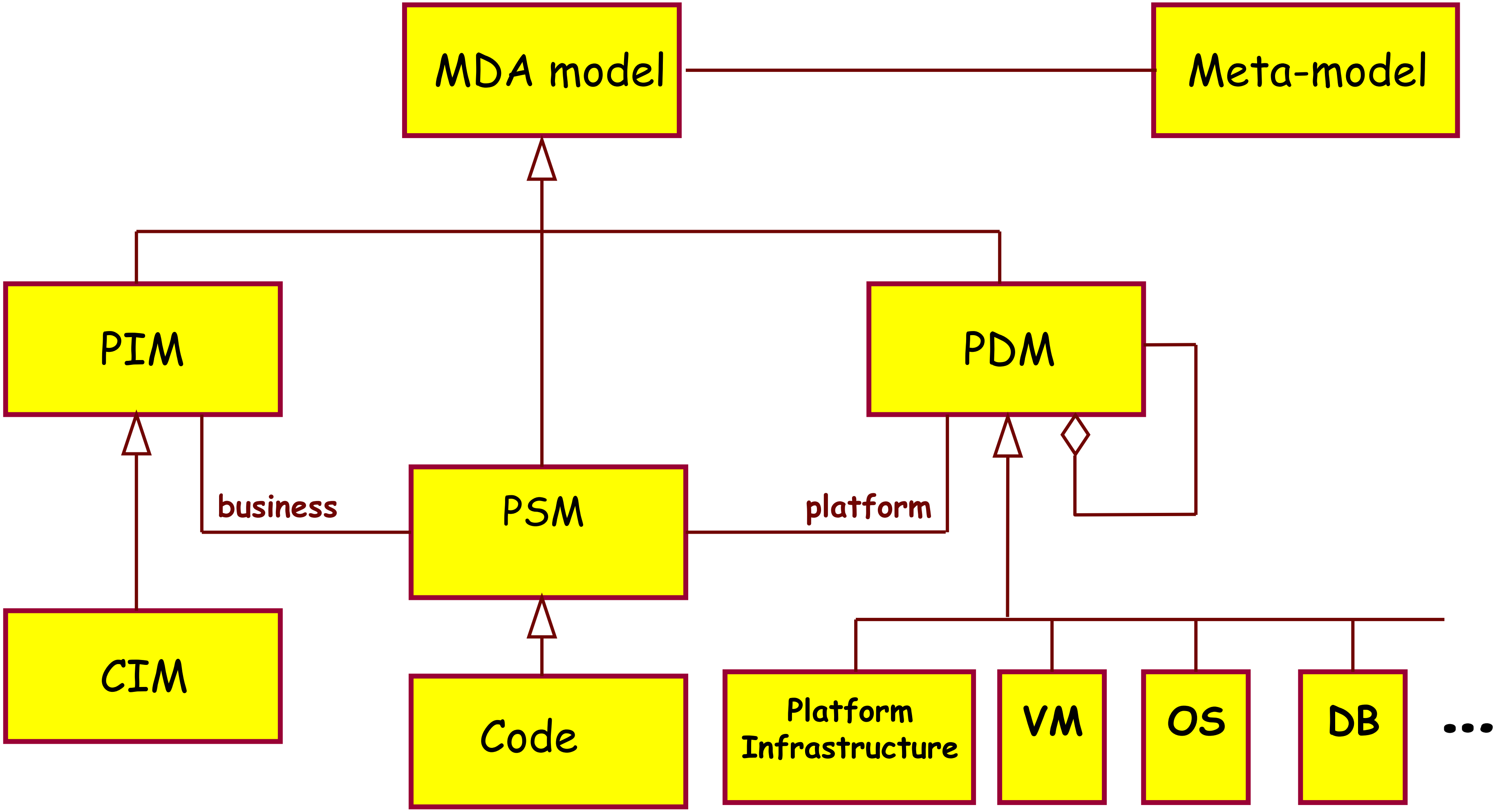
Paraphrasing Tony Hoare

[Compare to UML 2.0 approach]

# Research agenda: Everything is a model

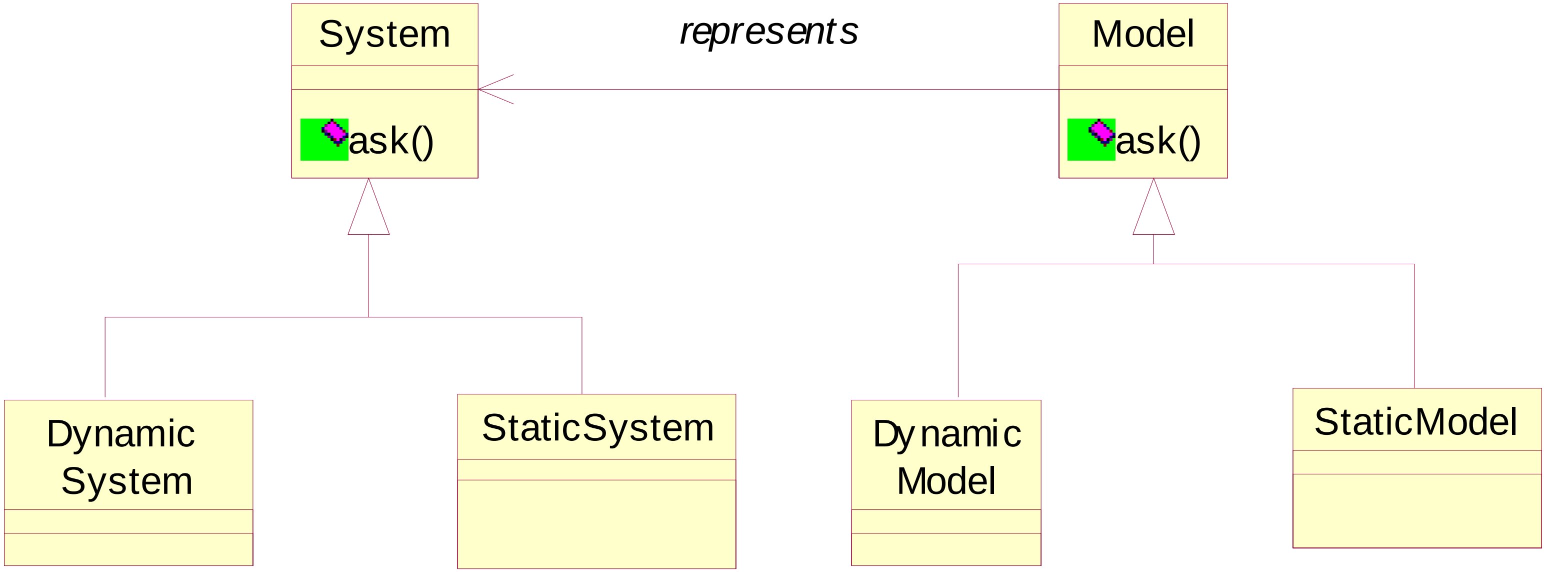
- What is a model?
  - A model is a representation of a system
  - A model is written in the language of its unique metamodel
  - A metamodel is written in the language of its unique metametamodel
    - The unique MMM of the MDA is the MOF
  - A model is a constrained directed labeled graph
  - A model may have a visual representation
- Where do models come from?
- What are the various kinds of models?

# A classification of explicit models (incomplete)



# Various kinds of models

- Products and processes
- Legacy and components
- Static and dynamic
- Functional and non-functional aspects
- Executable and non executable
- etc.

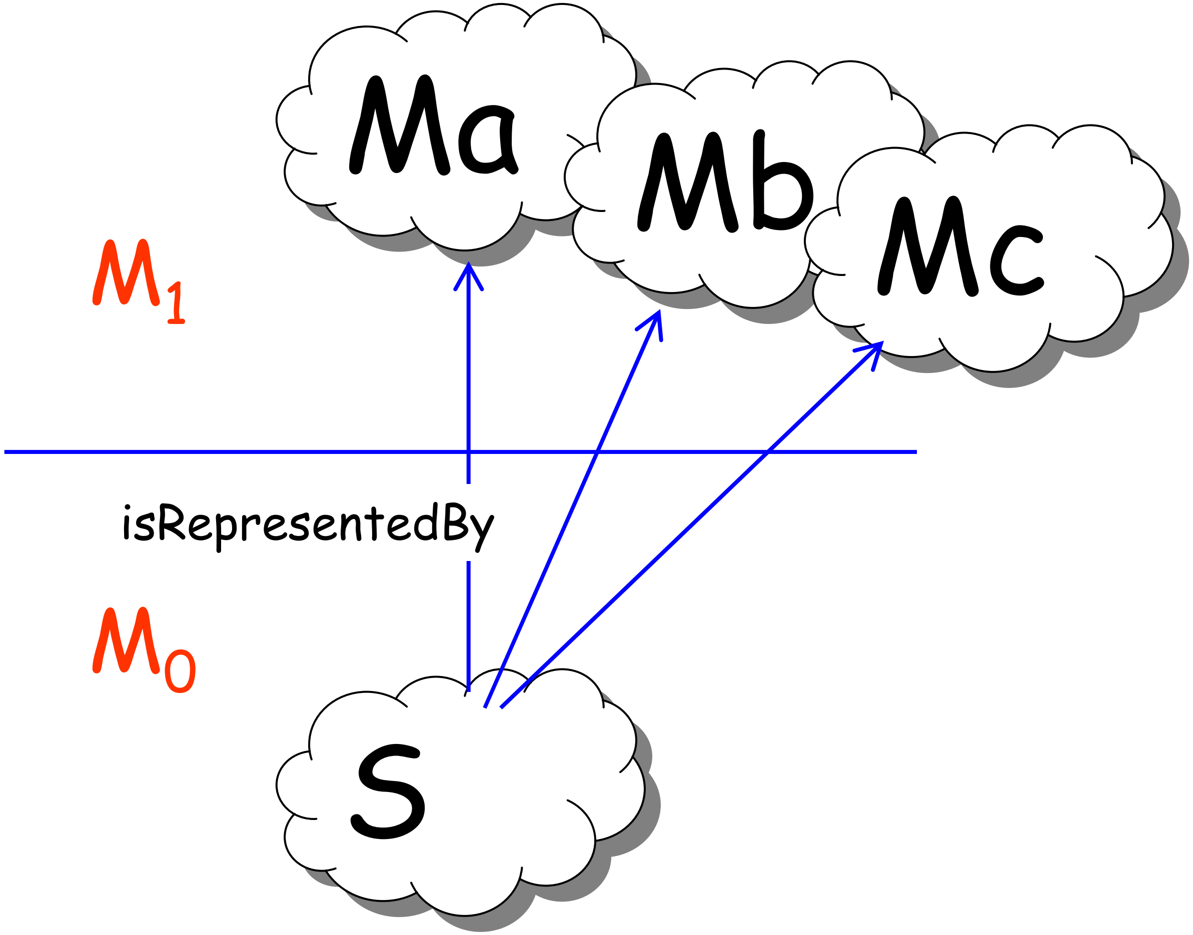


# MDA proposed R&D Agenda : "Everything is a model …"

… (or may be converted into a model, or represents a model, or refers to a model, etc. ), not only PIMs and PSMs

1. A process is a model
2. A platform is a model
3. A transformation is a model
4. A system is a model
5. A metamodel is a model
6. A model-element is a model
7. A program is a model
8. An execution trace is a model
9. A measure is a model
10. A test is a model
11. A decoration is a model
12. An aspect is a model
13. A pattern is a model
14. A legacy system is a model
15. etc.

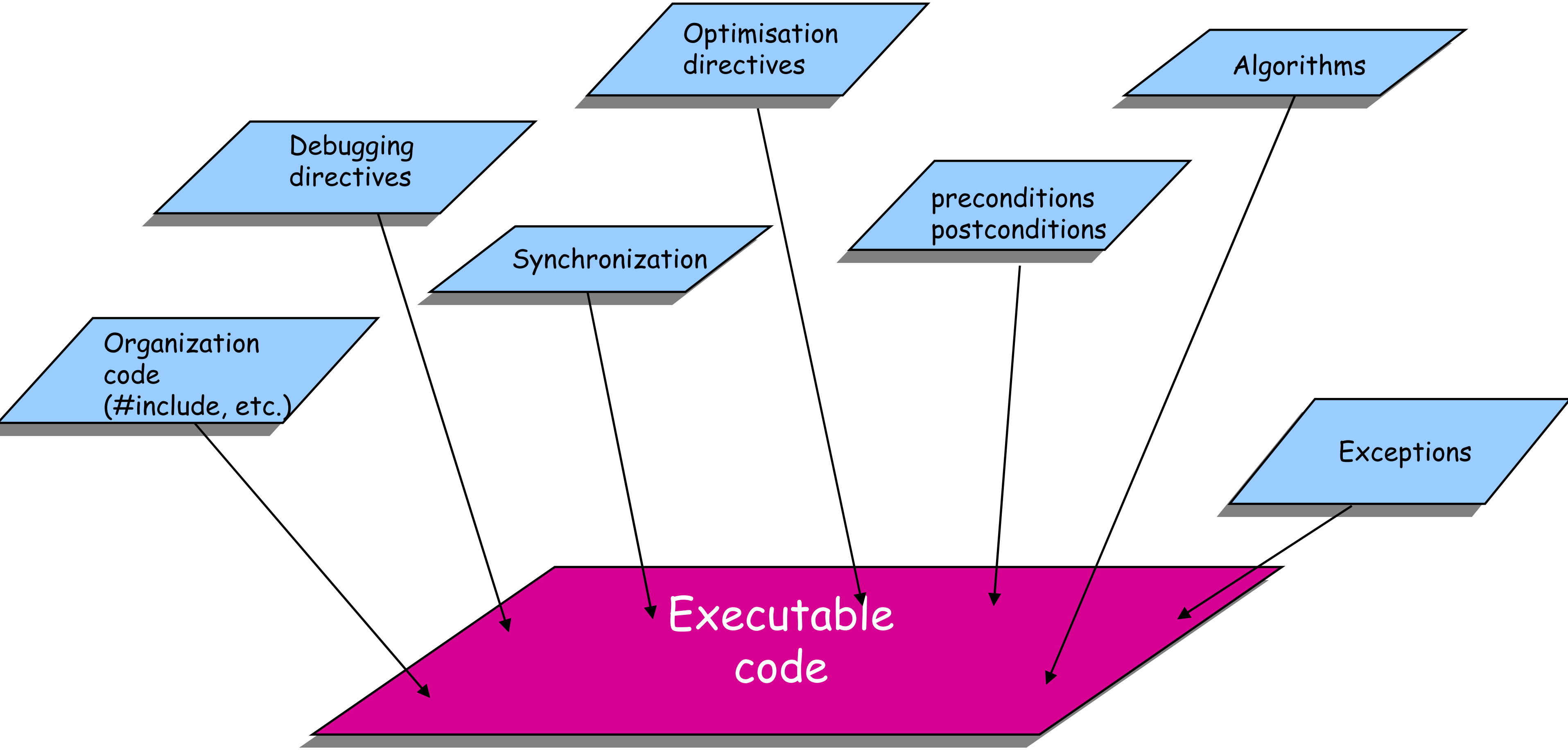
# Aspects as models



A given system may have plenty of different models.

Each model represents a given aspect of the system.

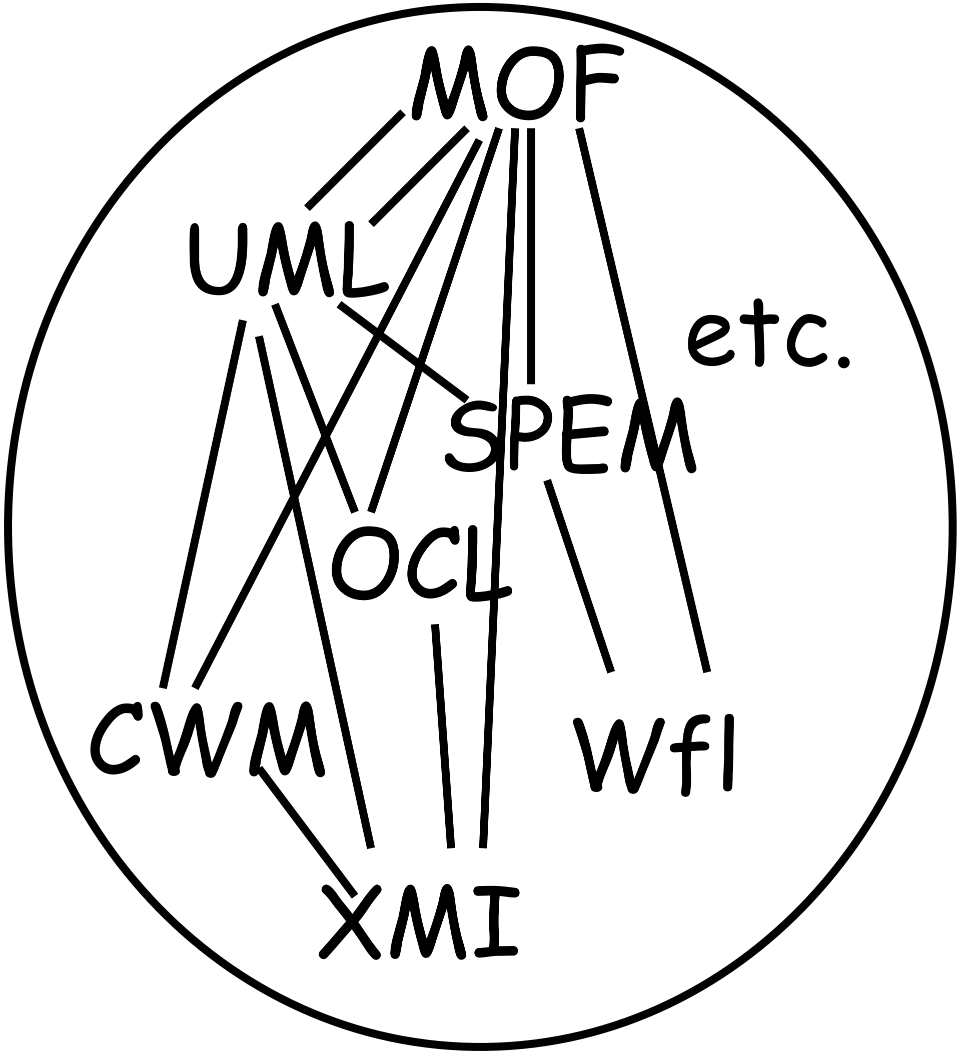
# An aspect is a model



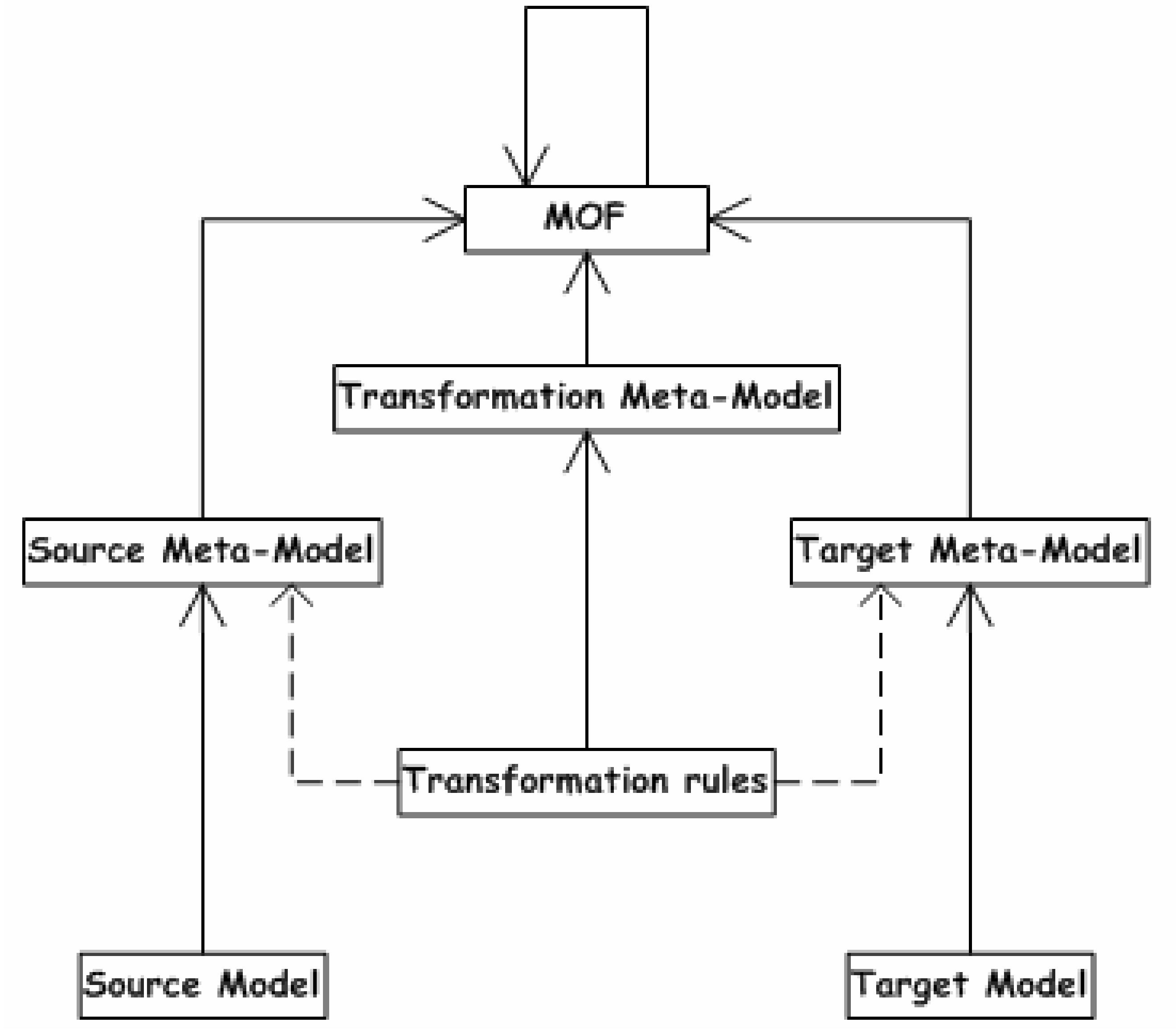
Relations between AOP and MDE mainly related to conveniency of representation systems.
Aspect weaving and model weaving are two techniques with partially overlapping scope.

# Model elements as models

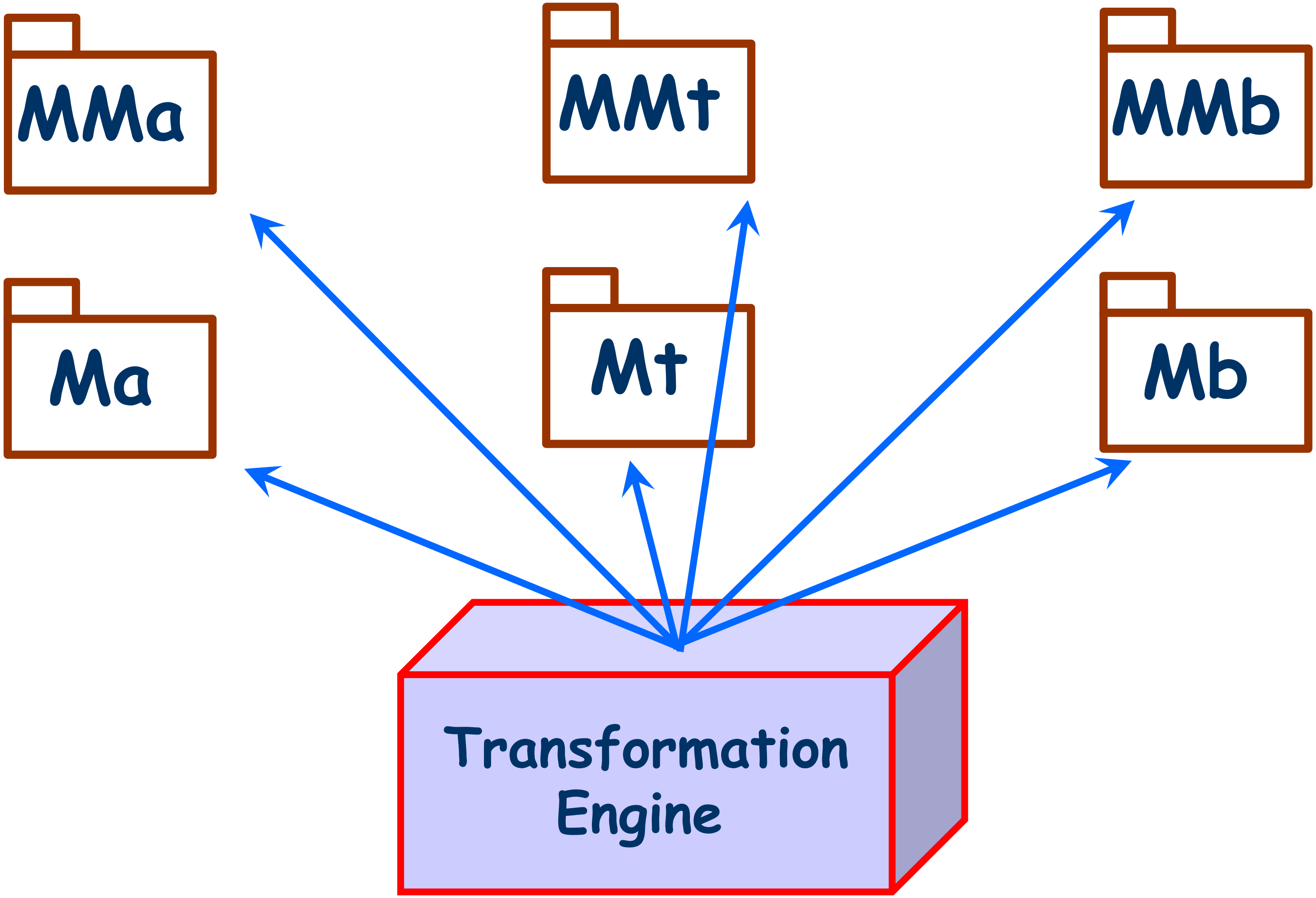
- Need global model and metamodel registies
- Need metadata on models and metamodels
- Need global view on global resources
- Modeling in the small and modeling in the large
- The notion of "megamodel"
  - Meta Object Facility (MOF)
  - Unified Modeling Language (UML)
  - XML Model Interchange (XMI)
  - Common Warehouse Meta-model (CWM)
  - Software Process Engineering Meta-model (SPEM)
  - Enterprise Distributed Object Computing (EDOC)
  - Corba Component Model (CCM)
  - Action Semantics Language (ASL)
  - Human Readable Textual Notation (HUTN)
  - Different UML profiles
  - ...



# Transformations as models



# Uniform access to models and metamodels



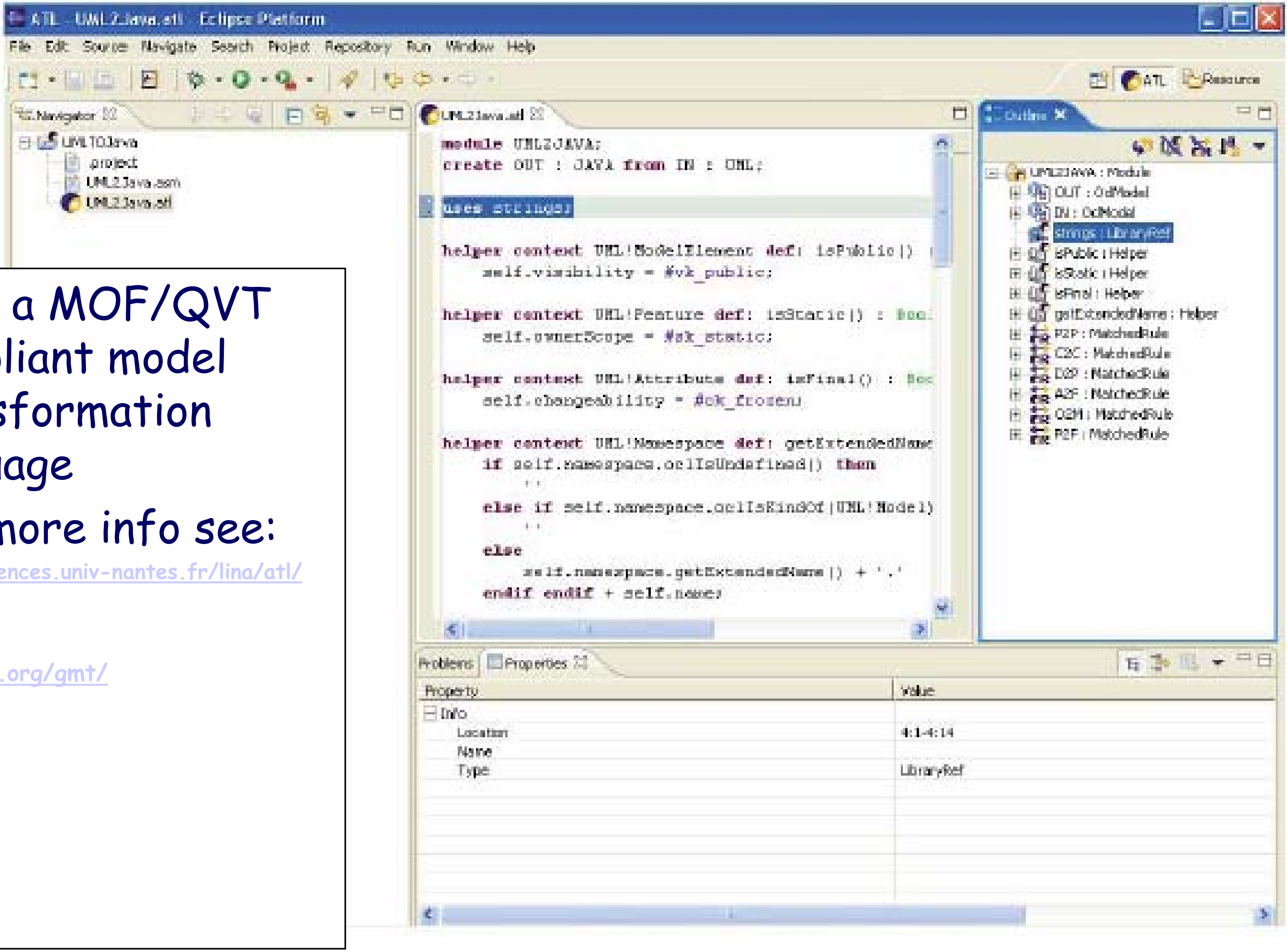
# ATL: a model transformation language, engine and IDE

- ATL: a MOF/QVT compliant model transformation language
- For more info see:

http://www.sciences.univ-nantes.fr/lina/atl/

- or

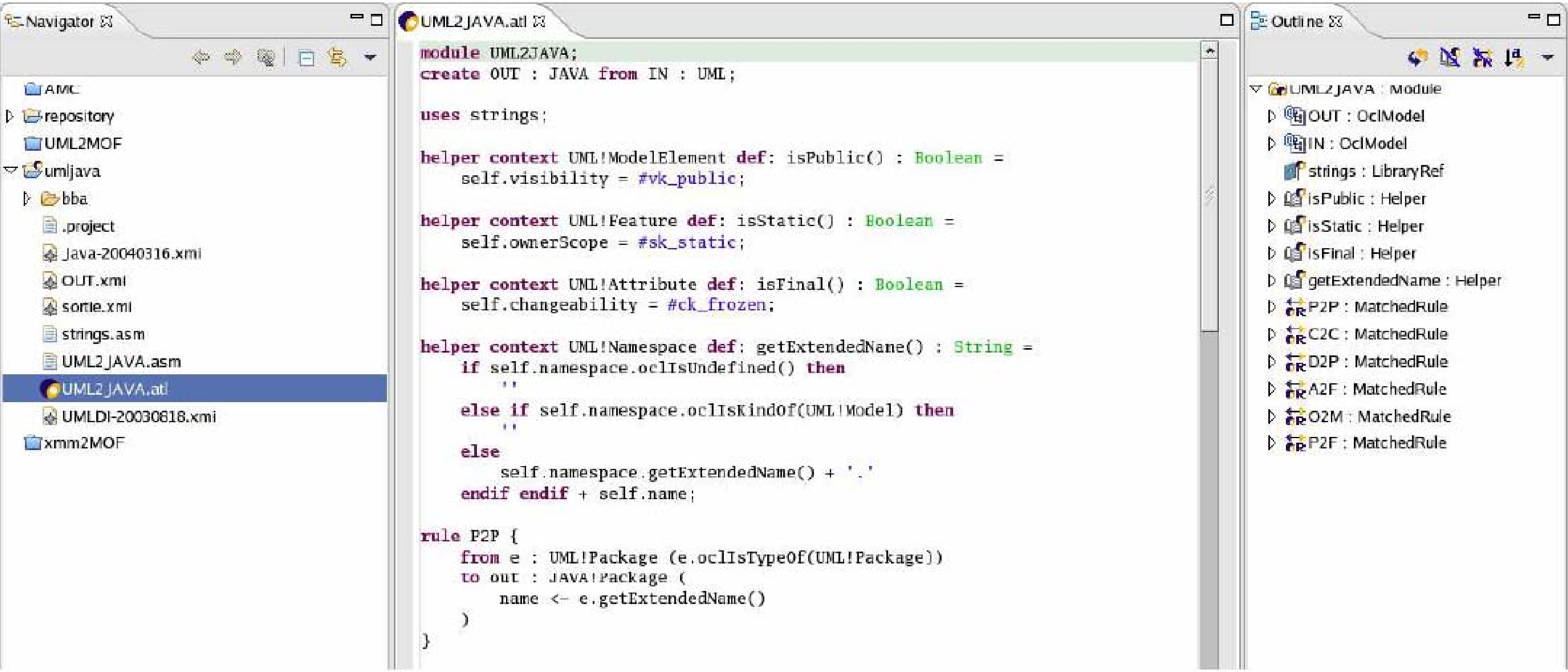
http://eclipse.org/gmt/



```
module UML2JAVA;
create OUT : JAVA from IN : UML;

uses strings;

helper context UML!ModelElement def: isPublic() :
    self.visibility = #vk_public;

helper context UML!Feature def: isStatic() : Boo
    self.ownerScope = #sk_static;

helper context UML!Attribute def: isFinal() : Boo
    self.changeability = #ck_frozen;

helper context UML!Namespace def: getExtendedName
    if self.namespace.oclIsUndefined() then
        ''
    else if self.namespace.oclIsKindOf(UML!Model)
        ''
    else
        self.namespace.getExtendedName() + '.'
    endif endif + self.name;
```

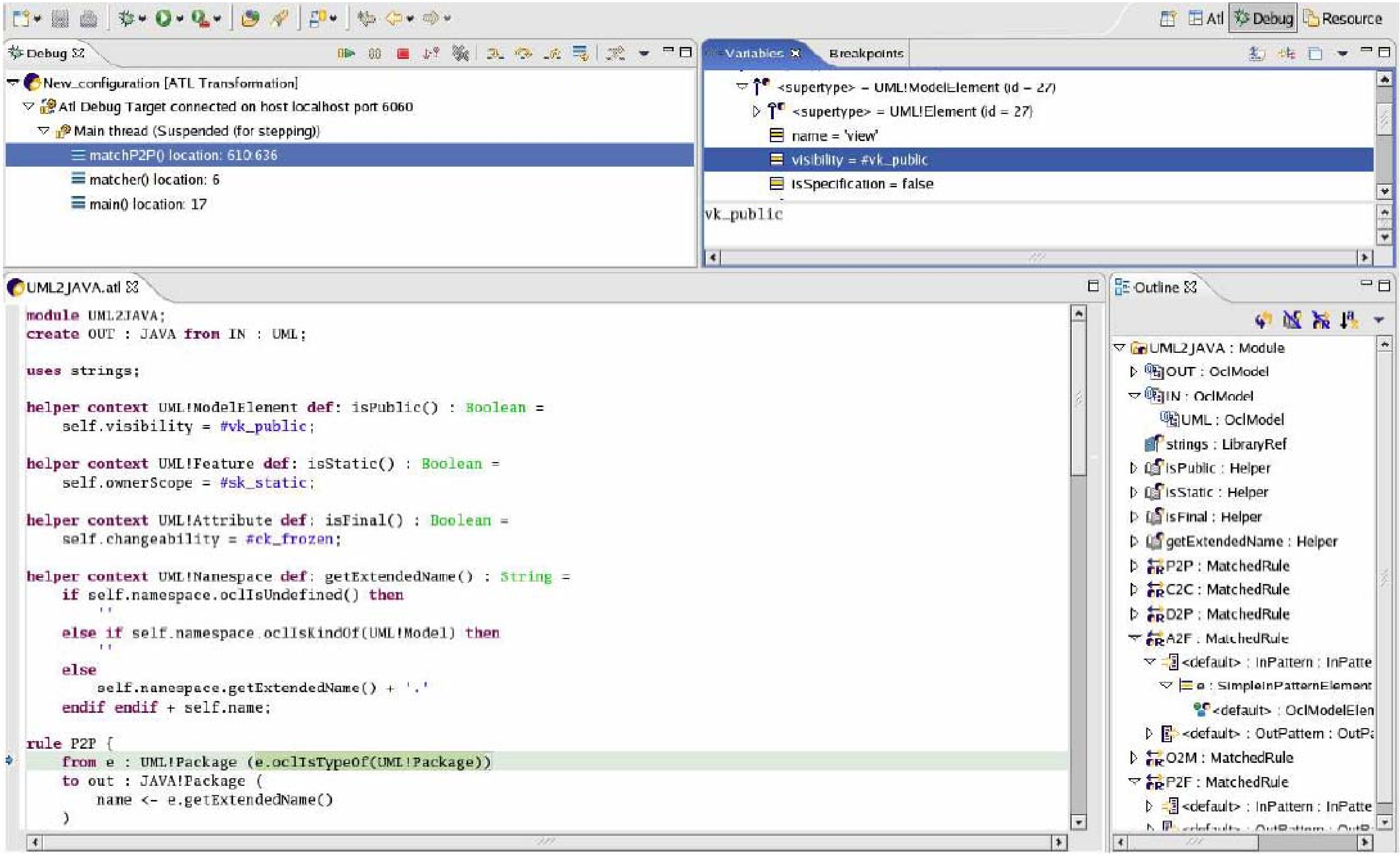
| Property | Value |
|---|---|
| Info | |
| Location | 4:1-4:14 |
| Name | |
| Type | LibraryRef |

# ATL editor (part of ATL Integrated Development Environment)



```
module UML2JAVA;
create OUT : JAVA from IN : UML;

uses strings;

helper context UML!ModelElement def: isPublic() : Boolean =
    self.visibility = #vk_public;

helper context UML!Feature def: isStatic() : Boolean =
    self.ownerScope = #sk_static;

helper context UML!Attribute def: isFinal() : Boolean =
    self.changeability = #ck_frozen;

helper context UML!Namespace def: getExtendedName() : String =
    if self.namespace.oclIsUndefined() then
        ''
    else if self.namespace.oclIsKindOf(UML!Model) then
        ''
    else
        self.namespace.getExtendedName() + '.'
    endif endif + self.name;

rule P2P {
    from e : UML!Package (e.oclIsTypeOf(UML!Package))
    to out : JAVA!Package (
        name <- e.getExtendedName()
    )
}
```

ATL Editor with its content outline



```
New_configuration [ATL Transformation]
  Atl Debug Target connected on host localhost port 6060
    Main thread (Suspended (for stepping))
      matchP2P() location: 610:636
      matcher() location: 6
      main() location: 17
```

```
<supertype> = UML!ModelElement (id = 27)
  <supertype> = UML!Element (id = 27)
  name = 'view'
  visibility = #vk_public
  isSpecification = false
vk_public
```

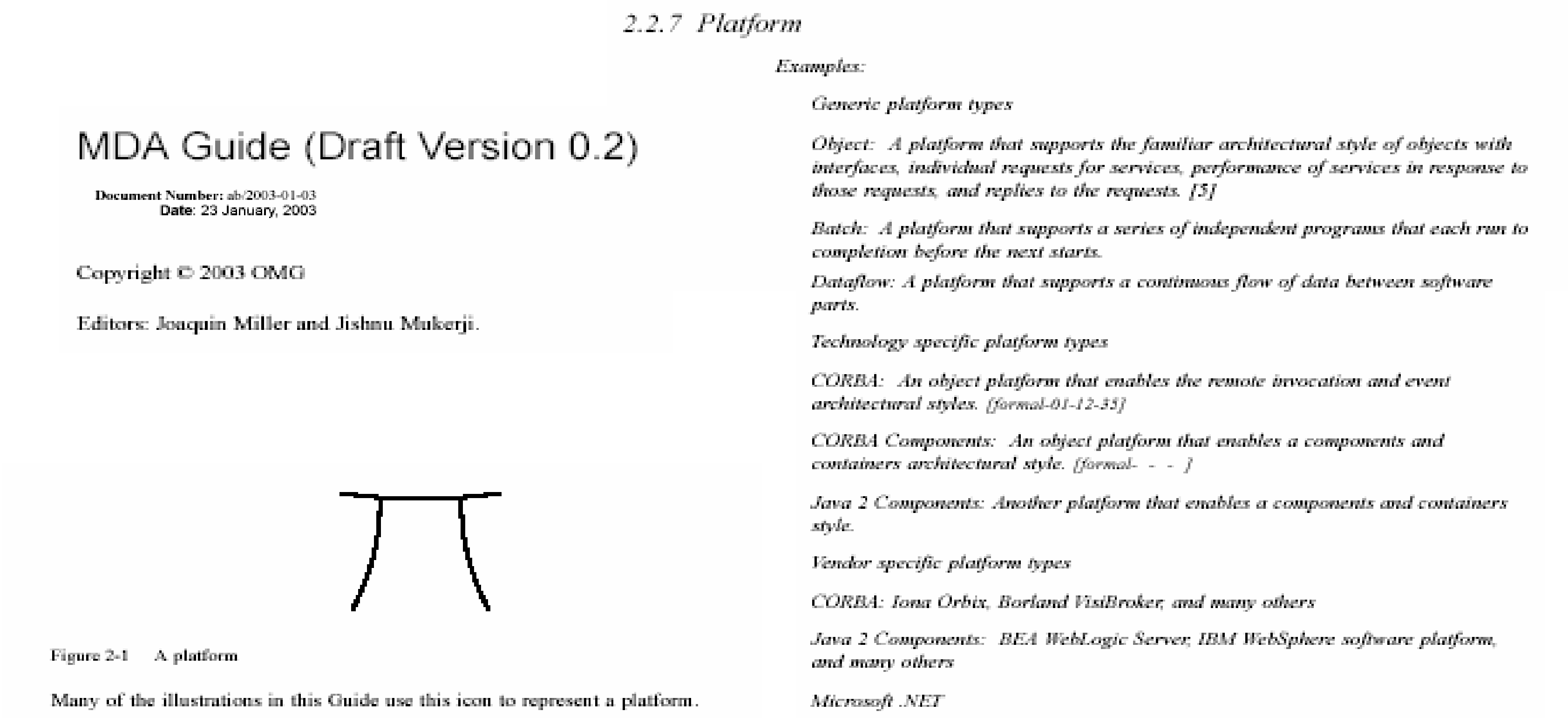
```
module UML2JAVA;
create OUT : JAVA from IN : UML;

uses strings;

helper context UML!ModelElement def: isPublic() : Boolean =
    self.visibility = #vk_public;

helper context UML!Feature def: isStatic() : Boolean =
    self.ownerScope = #sk_static;

helper context UML!Attribute def: isFinal() : Boolean =
    self.changeability = #ck_frozen;

helper context UML!Namespace def: getExtendedName() : String =
    if self.namespace.oclIsUndefined() then
        ''
    else if self.namespace.oclIsKindOf(UML!Model) then
        ''
    else
        self.namespace.getExtendedName() + '.'
    endif endif + self.name;

rule P2P {
    from e : UML!Package (e.oclIsTypeOf(UML!Package))
    to out : JAVA!Package (
        name <- e.getExtendedName()
    )
```

```
UML2JAVA : Module
  OUT : OclModel
  IN : OclModel
    UML : OclModel
  strings : LibraryRef
  isPublic : Helper
  isStatic : Helper
  isFinal : Helper
  getExtendedName : Helper
  P2P : MatchedRule
  C2C : MatchedRule
  D2P : MatchedRule
  A2F : MatchedRule
    <default> : InPattern : InPatte
      e : SimpleInPatternElement
        <default> : OclModelElen
    <default> : OutPattern : OutPa
  O2M : MatchedRule
  P2F : MatchedRule
    <default> : InPattern : InPatte
```

ATL Debug perspective with the ATL Editor, Debug, Variable and Outline view

# A platform is a model

- It is not possible to define platform specific models without having  defined precisely what a platform is.



MDA Guide (Draft Version 0.2)

**Document Number:** ab/2003-01-03
Date: 23 January, 2003

Copyright © 2003 OMG

Editors: Joaquin Miller and Jishnu Mukerji.

Figure 2-1 A platform

Many of the illustrations in this Guide use this icon to represent a platform.

*2.2.7 Platform*

*Examples:*

*Generic platform types*

*Object: A platform that supports the familiar architectural style of objects with interfaces, individual requests for services, performance of services in response to those requests, and replies to the requests. [5]*

*Batch: A platform that supports a series of independent programs that each run to completion before the next starts.*

*Dataflow: A platform that supports a continuous flow of data between software parts.*

*Technology specific platform types*

*CORBA: An object platform that enables the remote invocation and event architectural styles. [formal-01-12-35]*

*CORBA Components: An object platform that enables a components and containers architectural style. [formal- - - ]*

*Java 2 Components: Another platform that enables a components and containers style.*

*Vendor specific platform types*

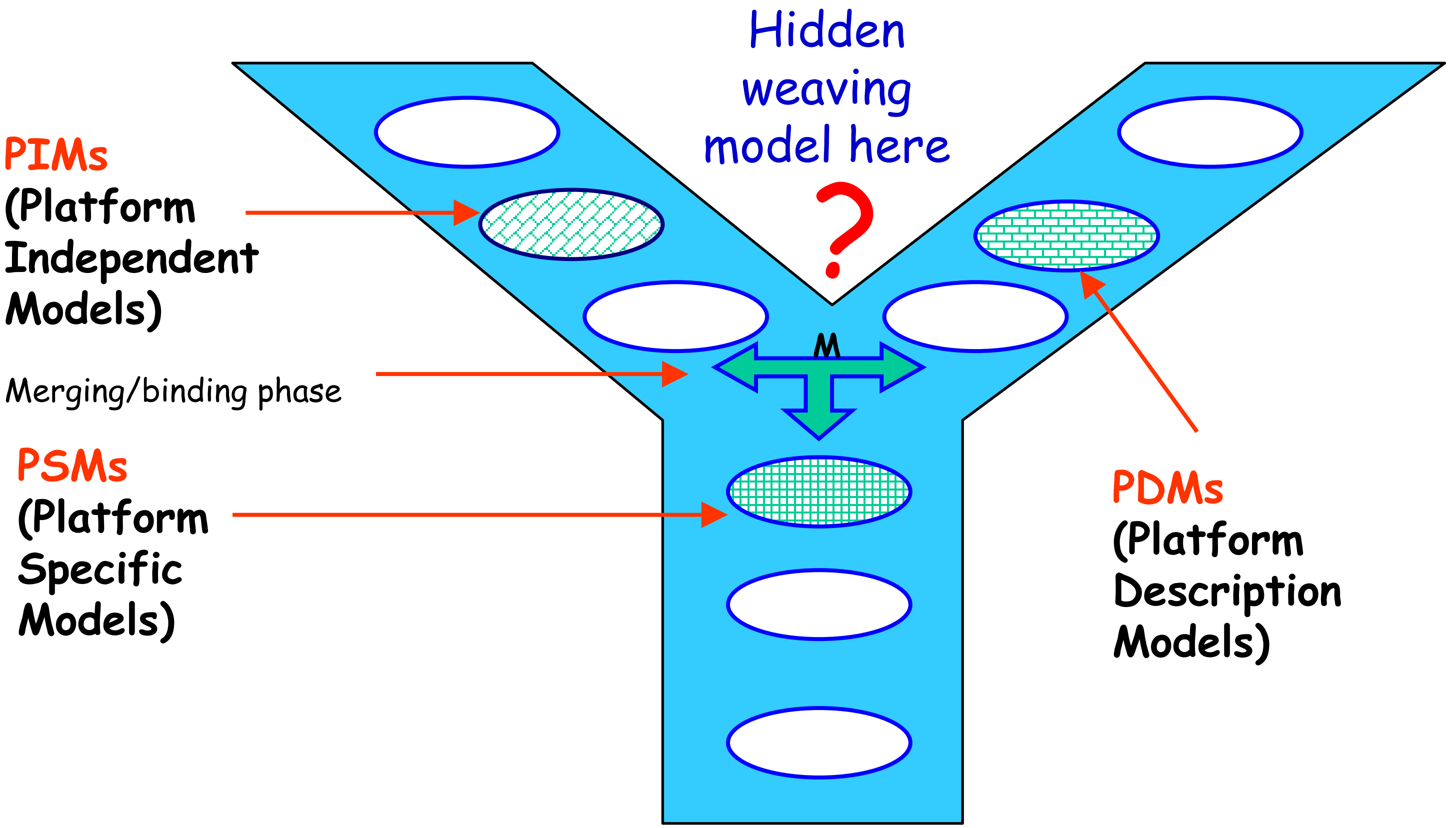
*CORBA: Iona Orbix, Borland VisiBroker, and many others*

*Java 2 Components: BEA WebLogic Server, IBM WebSphere software platform, and many others*

*Microsoft .NET*

# A correspondence is a model

- It is not possible to weave two models without having exactly defined the weaving model.


Hidden weaving model here
?
PIMs (Platform Independent Models)
Merging/binding phase
M
PSMs (Platform Specific Models)
PDMs (Platform Description Models)

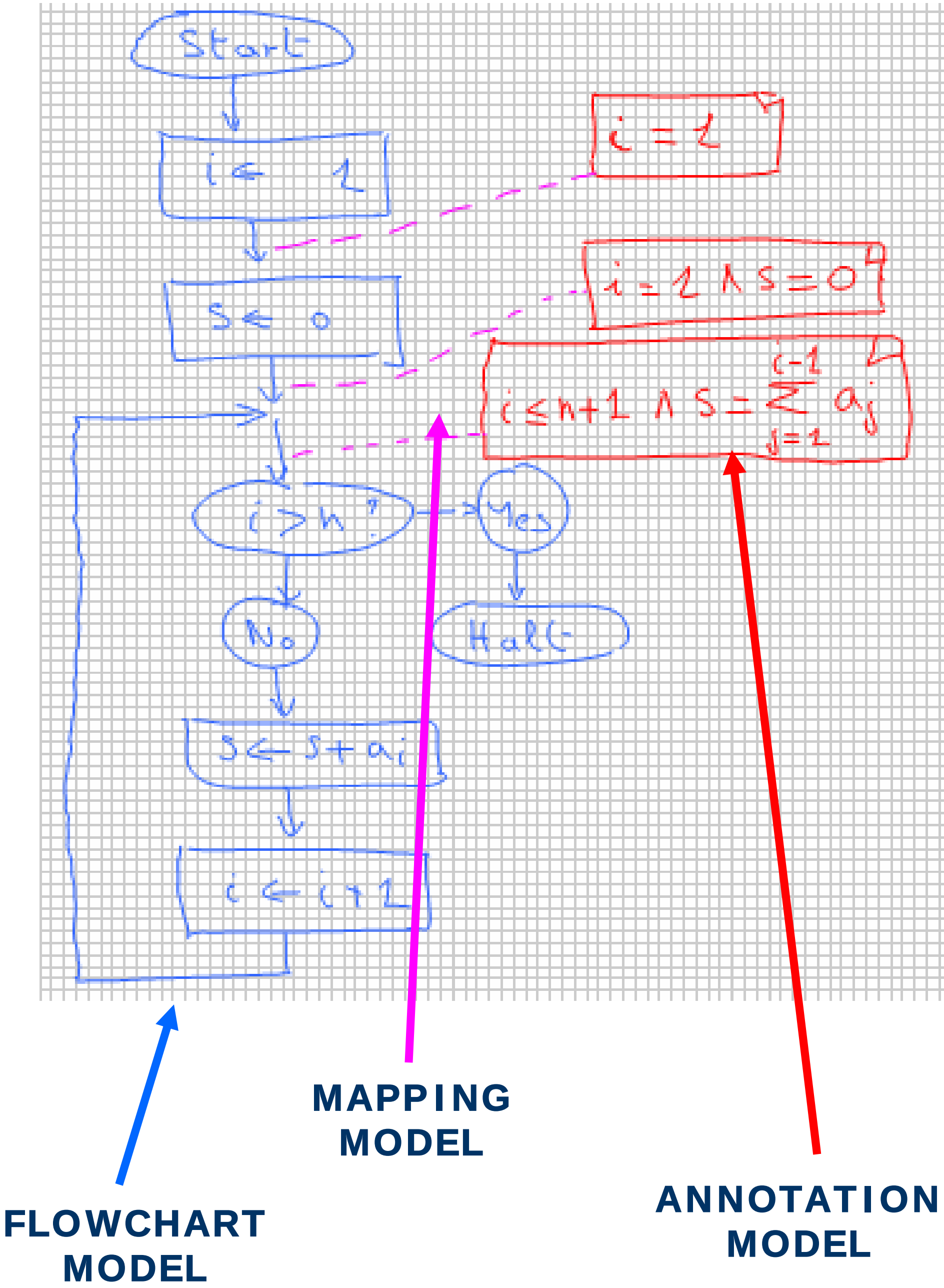
# Assigning meanings to models

- Floyd established the foundation of modern assertion techniques by proposing to decorate a program with specific annotations (pre and post conditions)
- "An interpretation I of a flowchart is a mapping of its edges on propositions"

Robert W Floyd

"Assigning meanings to programs"

Symposia in applied mathematics, 1965

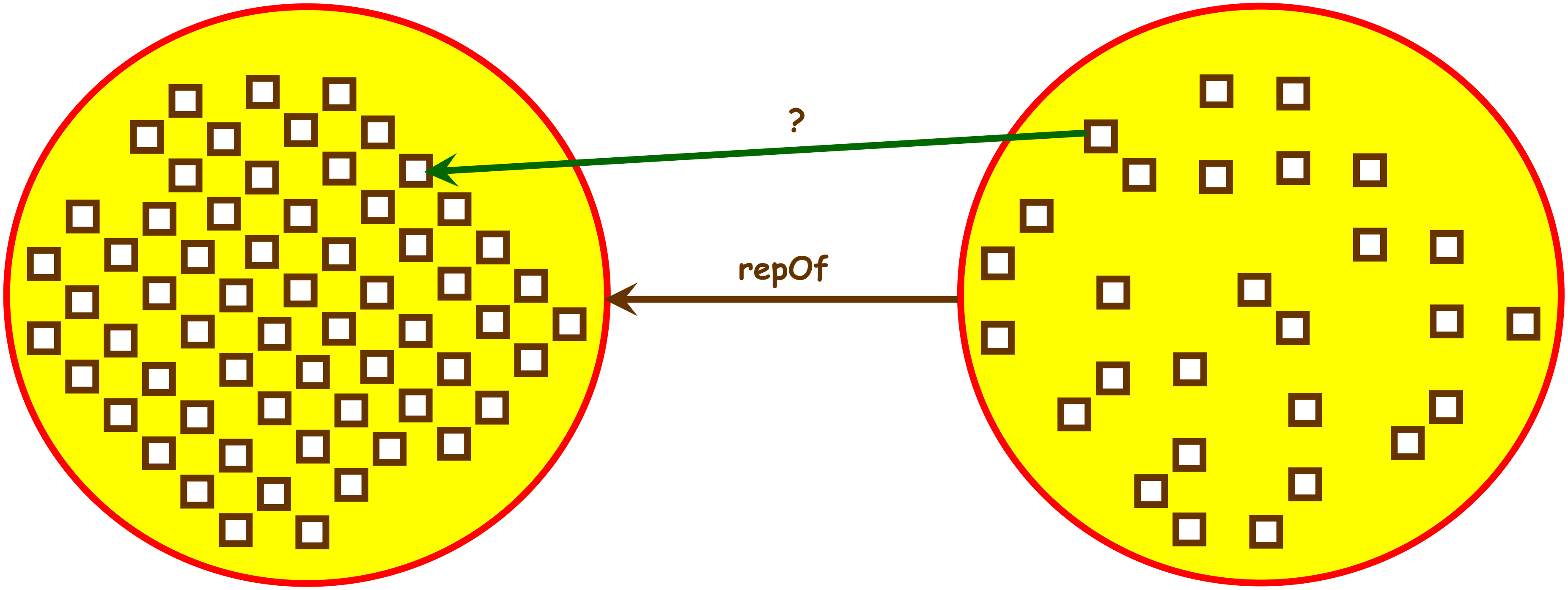


MAPPING MODEL

ANNOTATION MODEL

FLOWCHART MODEL

# The "representation" relation



System and System elements

Model and Model elements

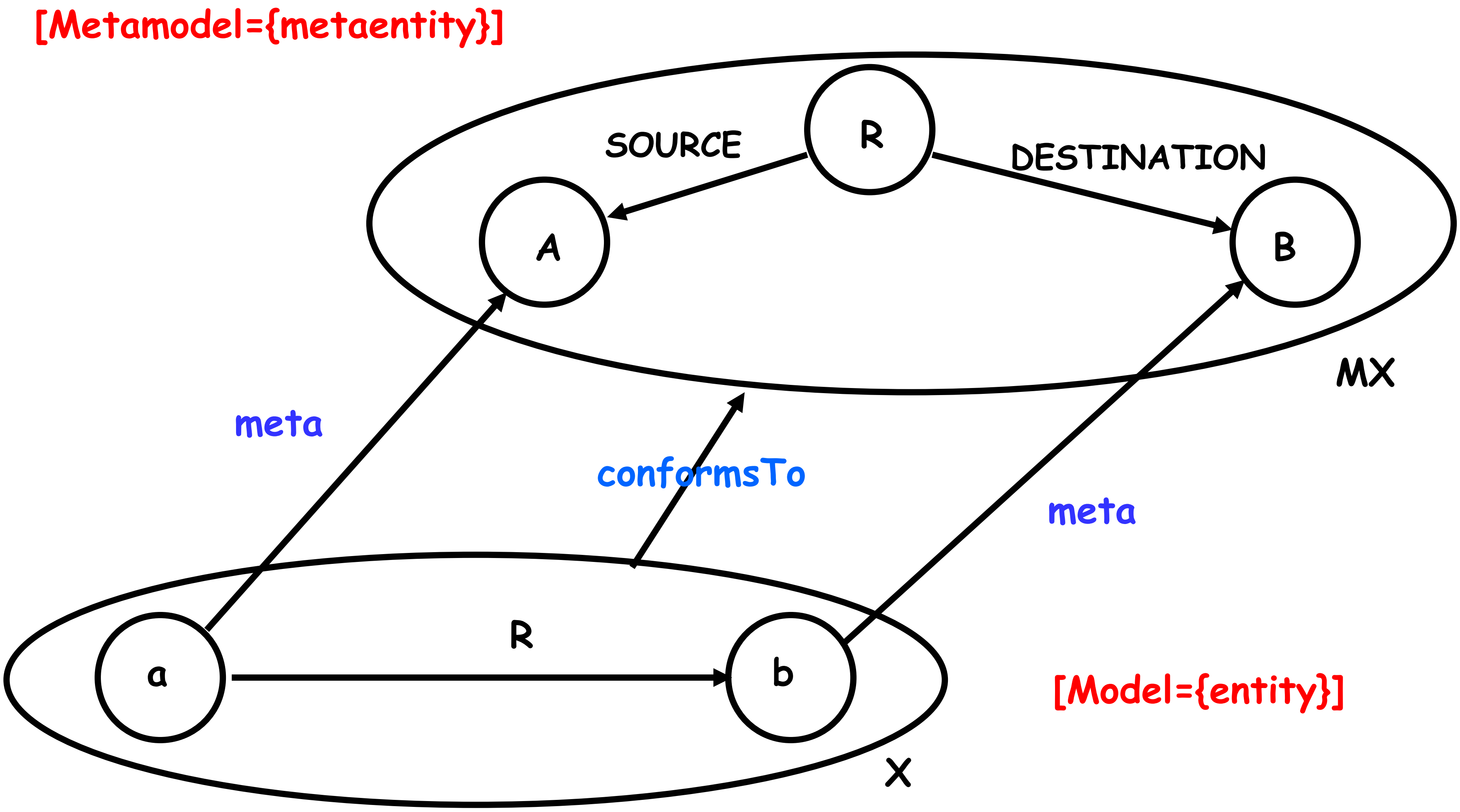
Simple set interpretation of the *repOf* relation is probably as correct as simple set interpretation of the *instanceOf* relation in object technology.

# On the *repOf* relation

"What about the [relationship between model and real-world]? The answer, and one of the main points I hope you will take away from this discussion, is that, at this point in intellectual history, we have no theory of this [...] relationship."

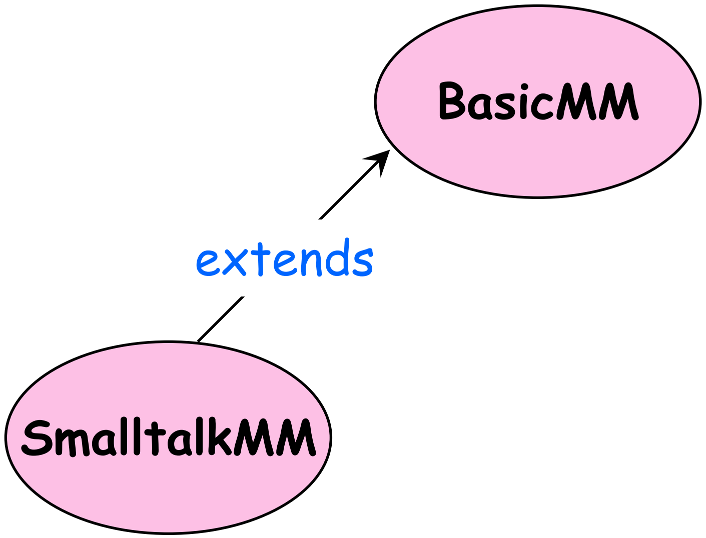
Cantwell Smith, B. Limits of Correctness in Computers, Report CSLI-85-36, Center for the Study of Language and Information, Stanford University, California, October 1985.

# The "conformance" (c2) relation

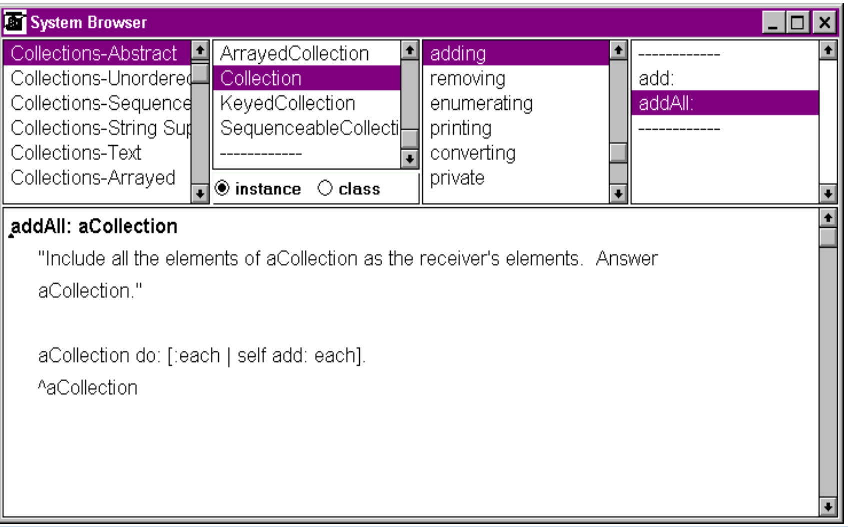


$\forall$ X, MX : Spaces | conformsTo(X,MX) $\Rightarrow$ $\forall$e$\in$X, $\exists$ me$\in$MX | meta(e,me)

# A Smalltalk metamodel

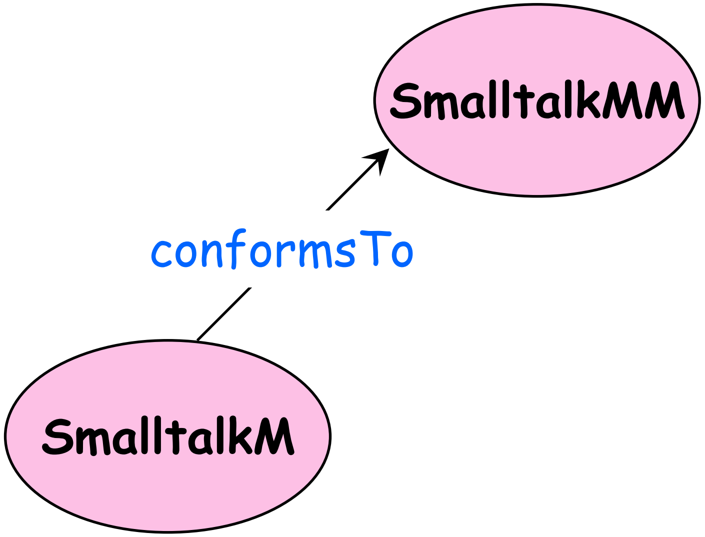


```
metamodel SmalltalkMM extends BasicMM;
class
        Category, Class, Protocol, Method;
association
        categoryOf (Class*,  Category);
        protocolOf (Protocol*, Class);
        methodOf (Method*, Protocol);
        textOf (Method, String);
endOfMetamodel SmalltalkMM;
```

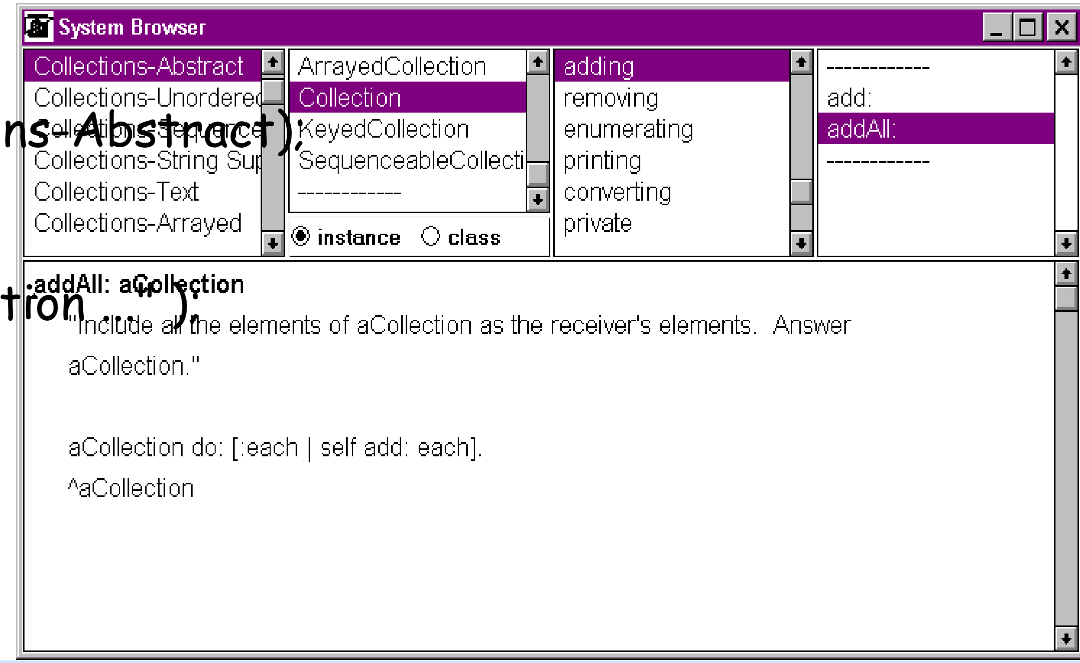


```
addAll: aCollection
    "Include all the elements of aCollection as the receiver's elements.  Answer
    aCollection."

    aCollection do: [:each | self add: each].
    ^aCollection
```

# A Smalltalk model

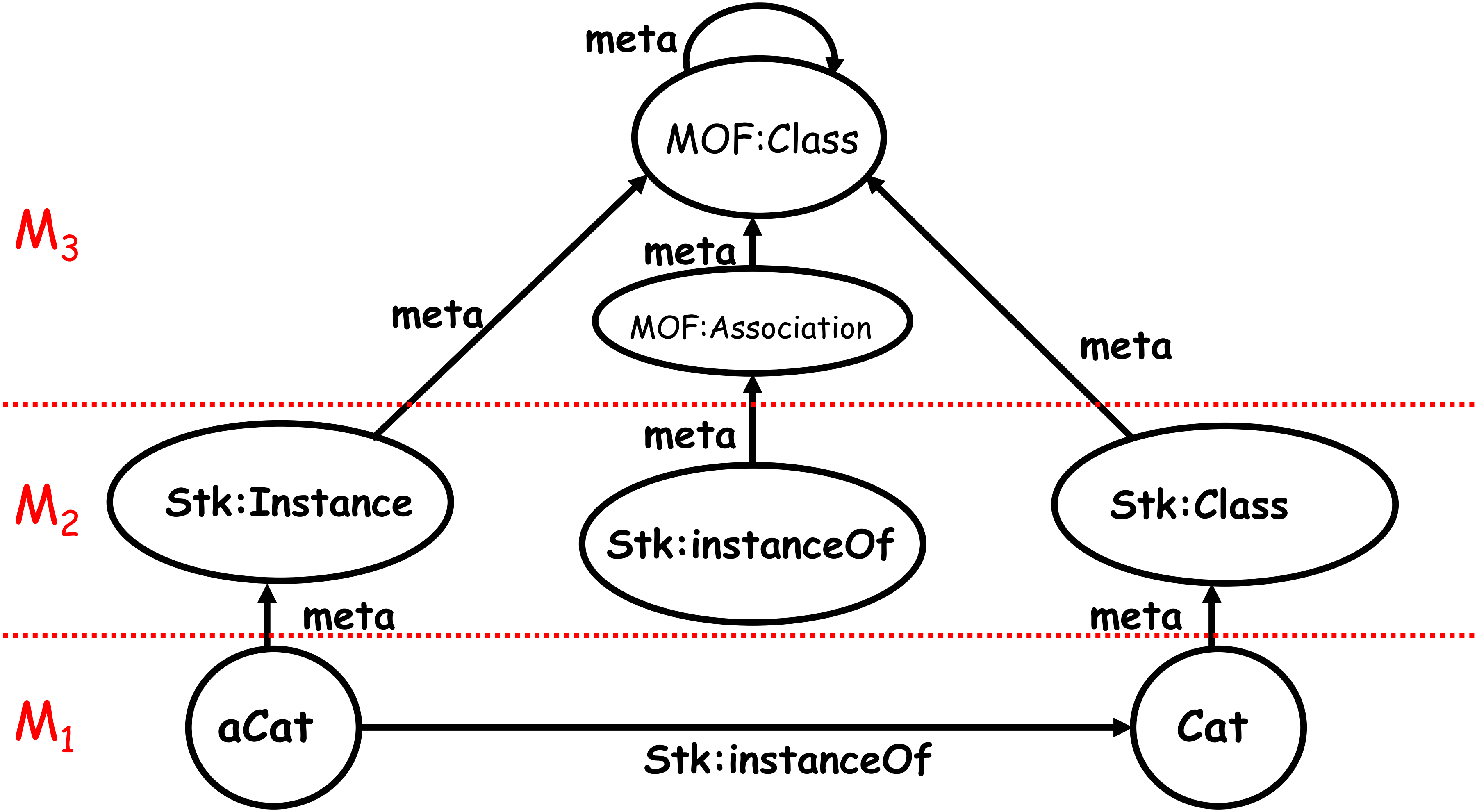


```
model SmalltalkM  conformsTo SmalltalkMM;
class
    Collections-Abstract:Category;
    Collection:Class;
    adding: Protocol;
    addAll: Method;
    "addAll: a Collection ..." : String;
association
    categoryOf (Collection, Collections-Abstract);
    protocolOf (adding,Collection);
    methodOf (addAll:,adding);
    textOf (addAll, "addAll: a Collection ..." );
endOfModel SmalltalkM;
```



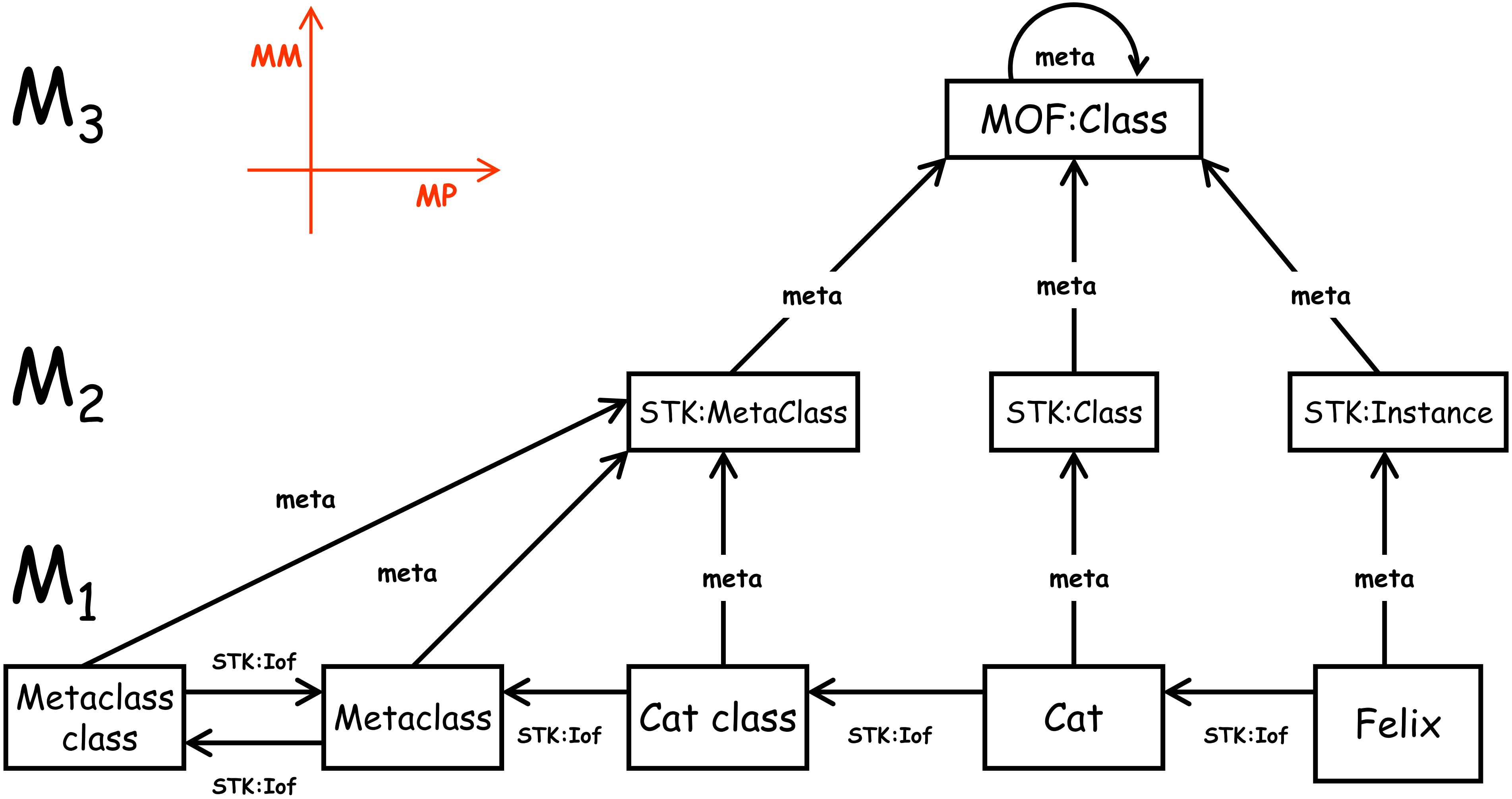
```
addAll: aCollection
    "Include all the elements of aCollection as the receiver's elements.  Answer
    aCollection."

    aCollection do: [:each | self add: each].
    ^aCollection
```

# Local and global definitions



# Metaprogramming vs. Metamodelling

(there is no unique metamodel for a given language/system/environment, etc.)



Agenda

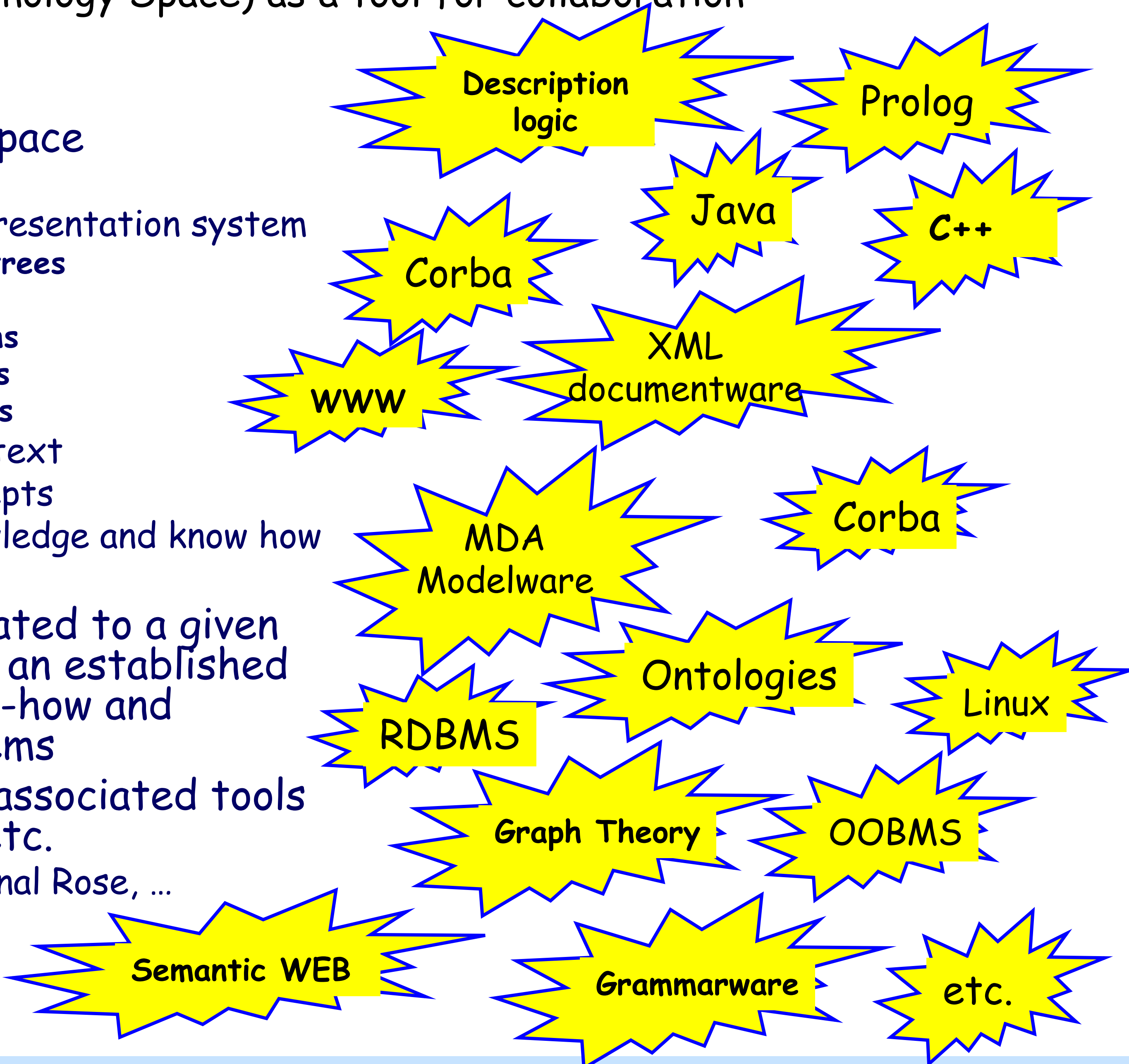
# Technology spaces

# Technical Spaces and Working Contexts

- Technical Spaces
  - Examples: MDE, EBNF, XML, DBMS, ontologies, etc.
  - Conjecture:
    - Each TS is represented by a metametamodel
    - Each TS is organized in a 3 metalevel architecture
- Working contexts
  - Local
    - MM specific, e.g. UML
  - Global
    - TS specific, MM independent, e.g. MOF
  - Universal
    - Across several TSs

# The notion of TS (Technology Space) as a tool for collaboration

- A Technology Space corresponds to:
  - A uniform representation system
    - **Syntactic trees**
    - **XML trees**
    - **Sowa graphs**
    - **UML graphs**
    - **MOF graphs**
  - A working context
  - A set of concepts
  - A shared knowledge and know how
  - etc.
- It is usually related to a given community with an established expertise, know-how and research problems
- It has a set of associated tools and practices, etc.
  - Protégé, Rational Rose, …



**Description logic**

Prolog

Java

**C++**

Corba

XML documentware

**WWW**

MDA Modelware

Corba

Ontologies

Linux

RDBMS

**Graph Theory**

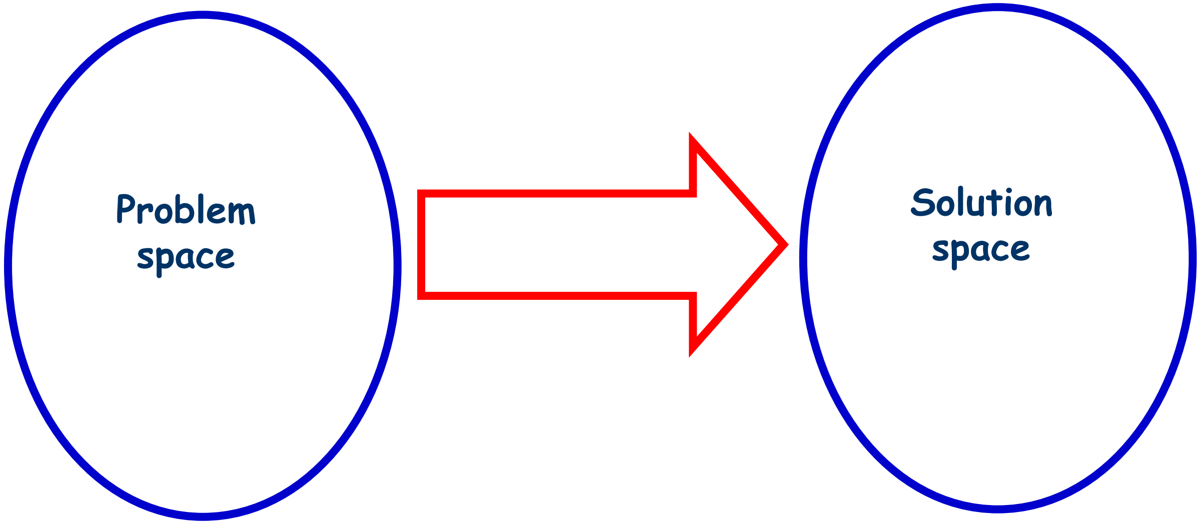
OOBMS

**Semantic WEB**

**Grammarware**

etc.

# The solution space is multiple and complex



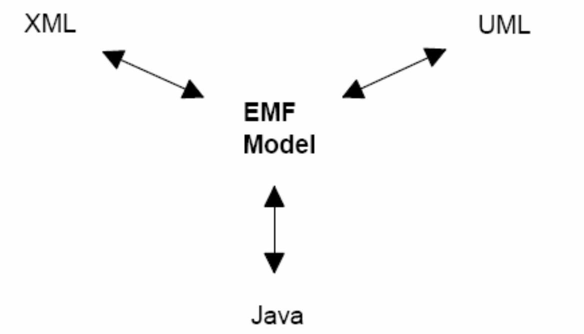
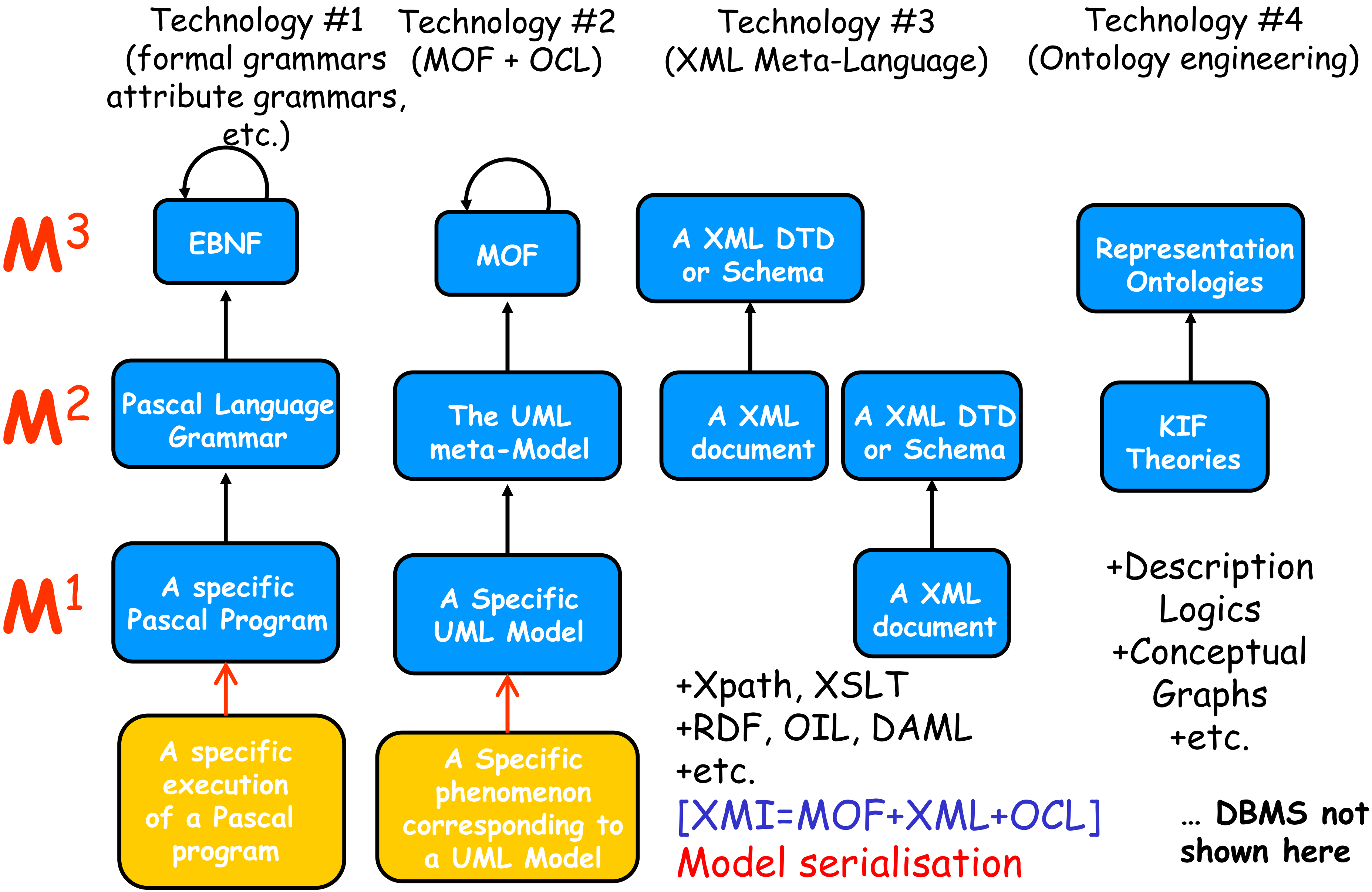


**Figure 2.2** EMF unifies Java, XML, and UML.

## How to map the problem space onto the composite solution space?

# Abstract Syntax Systems Compared



+Xpath, XSLT
+RDF, OIL, DAML
+etc.

[XMI=MOF+XML+OCL]

Model serialisation

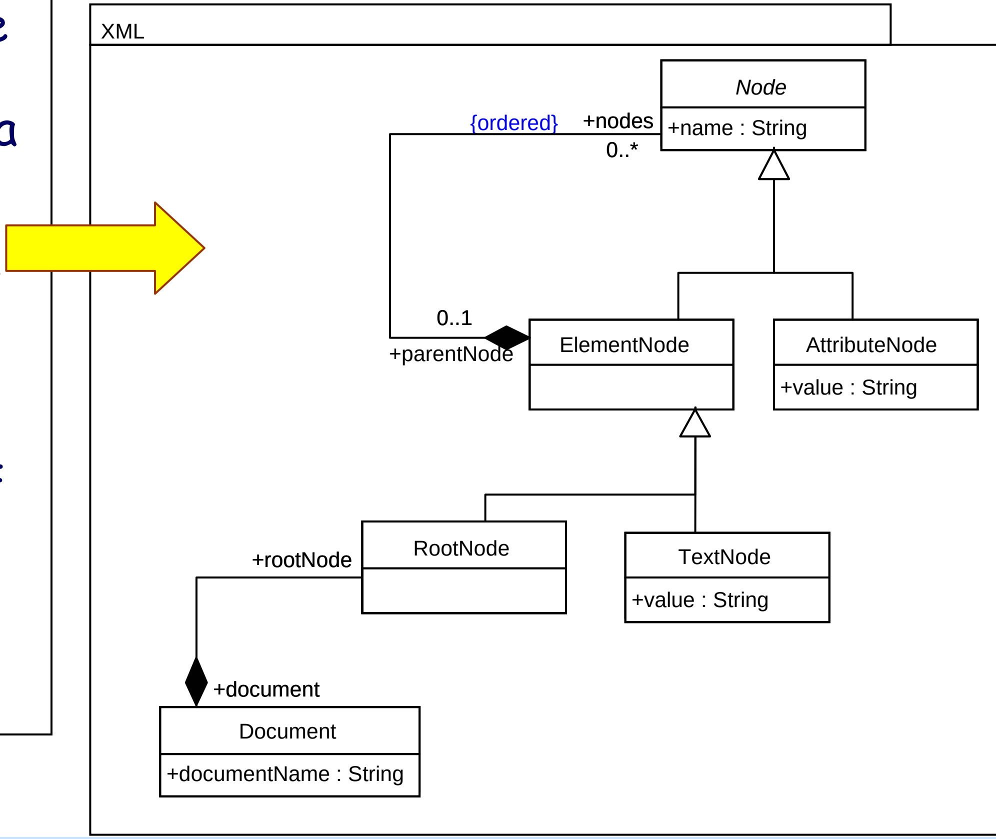
+Description Logics
+Conceptual Graphs
+etc.

… **DBMS not shown here**

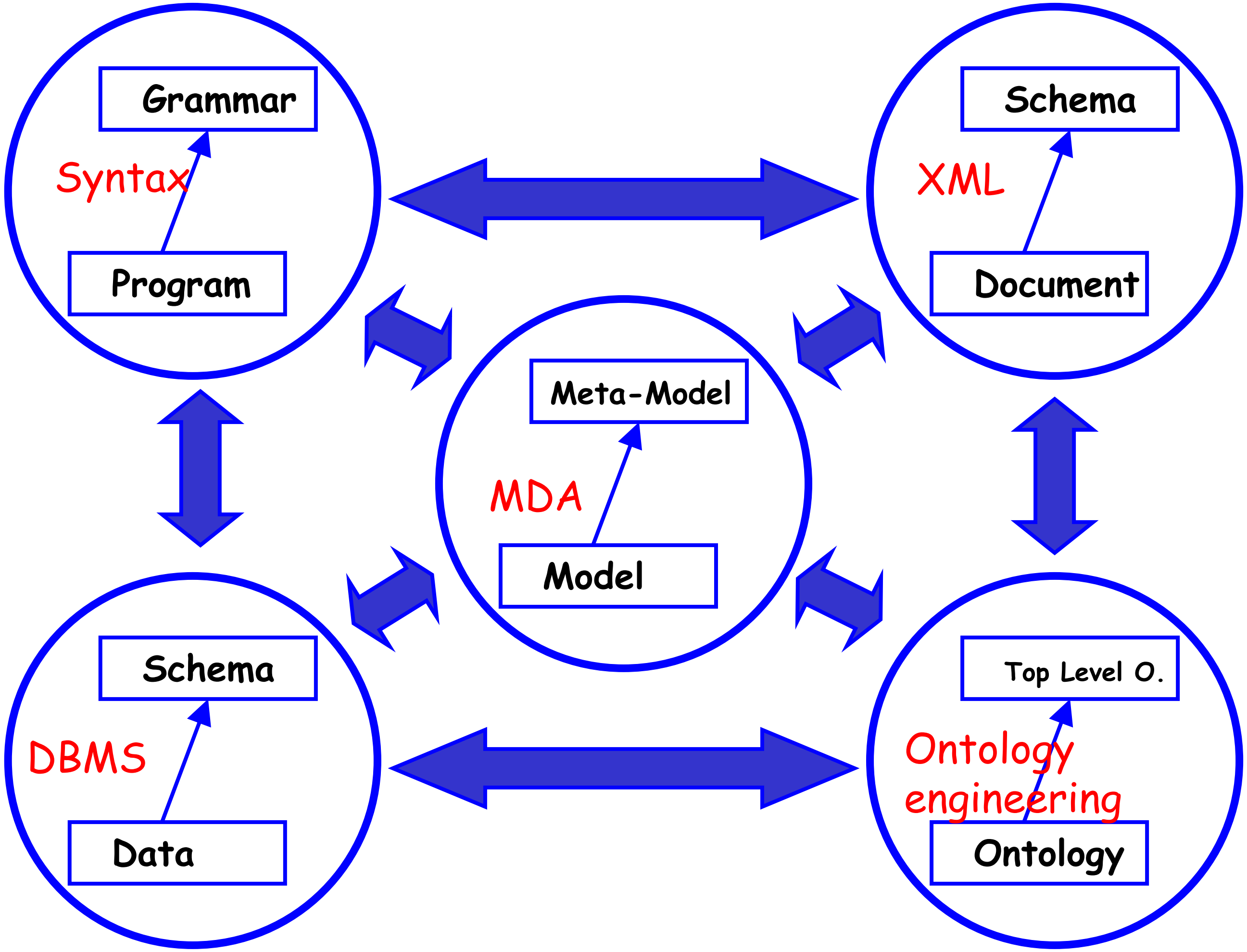
# Representation issues

- **Each TSpace is rooted on a specific representation ontology (RO)**
- This representation ontology is sometimes called a **metametamodel**
- For MDA people this is called the MOF
- Protegé for example is based on a specific RO
- XML is based on a specific RO describing a special kind of trees (well formed XML documents)
- Abstract syntax trees have a different RO
- MOF is a RO for a special kind of graphs
- sNets or Vanilla are other alternatives
- Some ROs use hypergraphs

**Fragment of the XML representation ontology**



## A technology space is organized around a set of concepts
## Spaces may be connected via bridges
## Spaces are often similarly organized



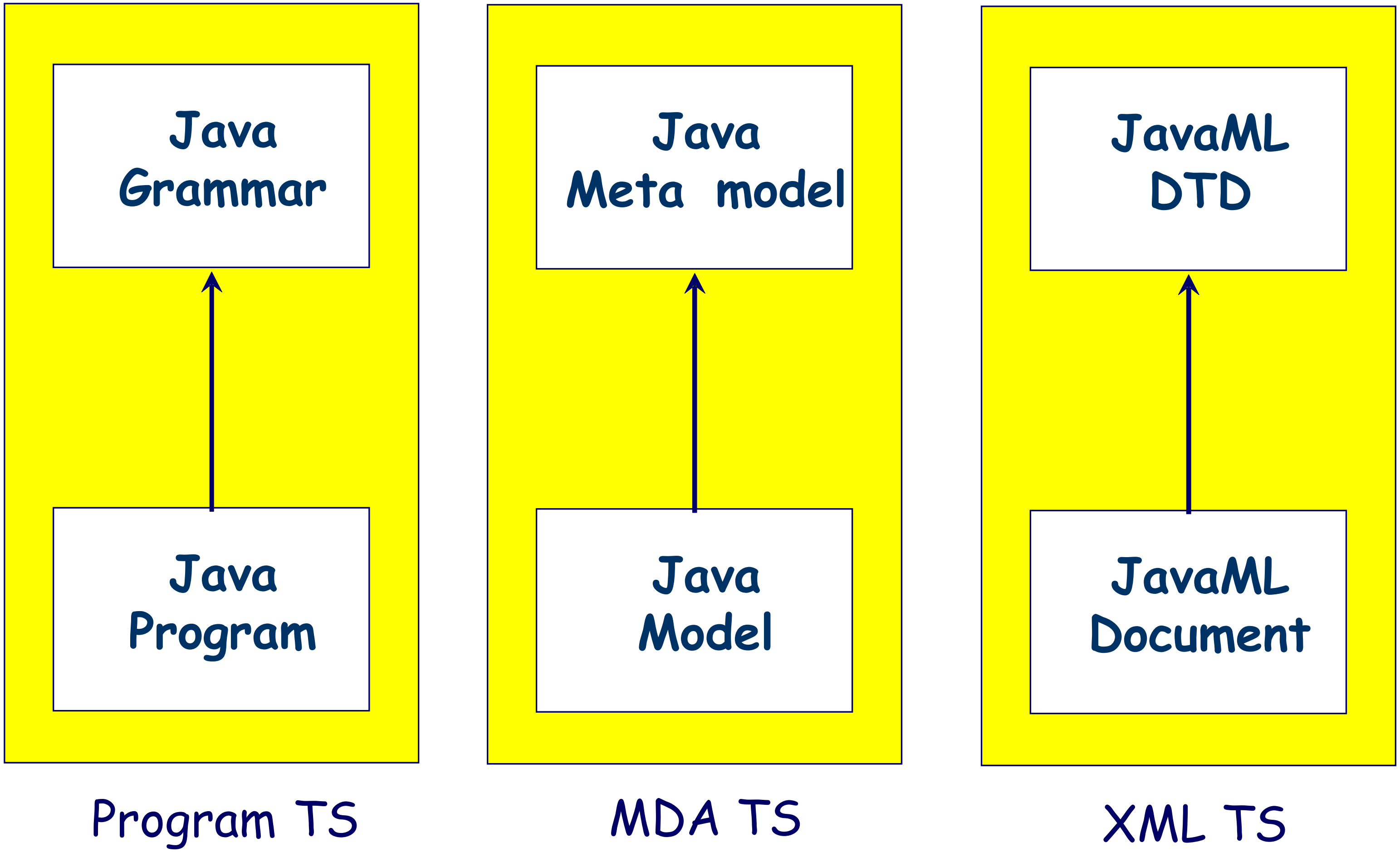
# Comparing spaces

+ stability in time

| | XML | MDA | Grammarware | Ontologies |
|---|---|---|---|---|
| **Executability** | Poor | Poor | Excellent | Poor |
| **Aspects** | Good | Excellent | Poor | Fair |
| **Formalization** | Poor | Poor | Excellent | Fair |
| **Specialization** | Fair | Good | Poor | Fair |
| **Modularity** | Good | Good | Poor | Poor |
| **Traceability** | Good | Fair | Poor | Excellent |
| **Transformability** | Excellent | Fair | Fair | Fair |

(NB: marks are indicative)

# Three representations for the same program



# Three representations for the same program

```
import java.applet.*;
import java.awt.*;

public class FirstApplet extends Applet {
  public void paint(Graphics g) {
    g.drawString("FirstApplet", 25, 50);
  }
}
```

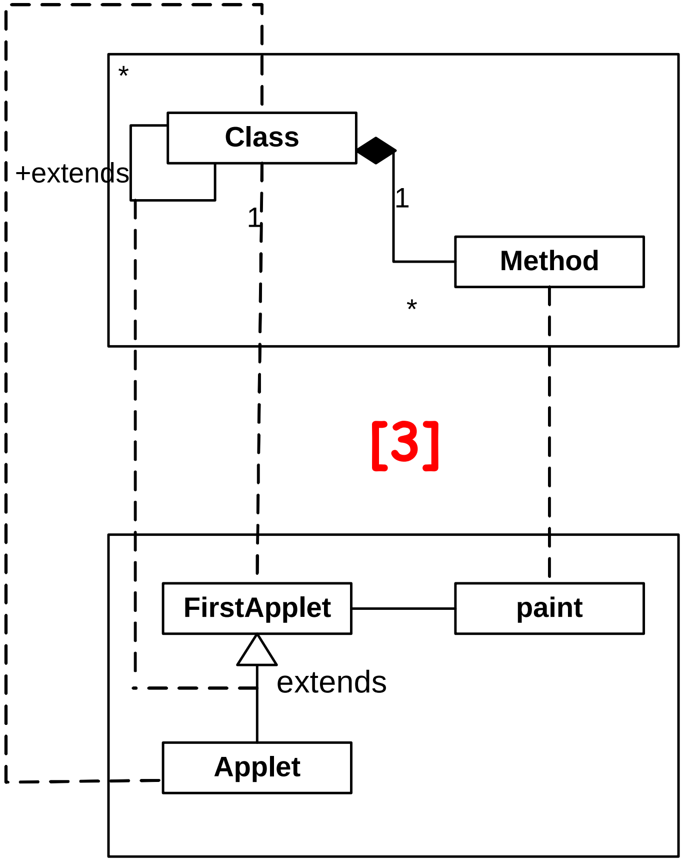
[1]

Java source code

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE java-source-program SYSTEM "java-ml.dtd">
<java-source-program name="FirstApplet.java">
  <import module="java.applet.*"/>
  <import module="java.awt.*"/>
  <class name="FirstApplet" visibility="public">
          <superclass class="Applet"/>
          <method name="paint" visibility="public" id="meth-15">
                    <type name="void" primitive="true"/>
                    <formal-arguments>
                              <formal-argument name="g" id="frmarg-13">
                              <type name="Graphics"/></formal-argument>
                    </formal-arguments> …….
```

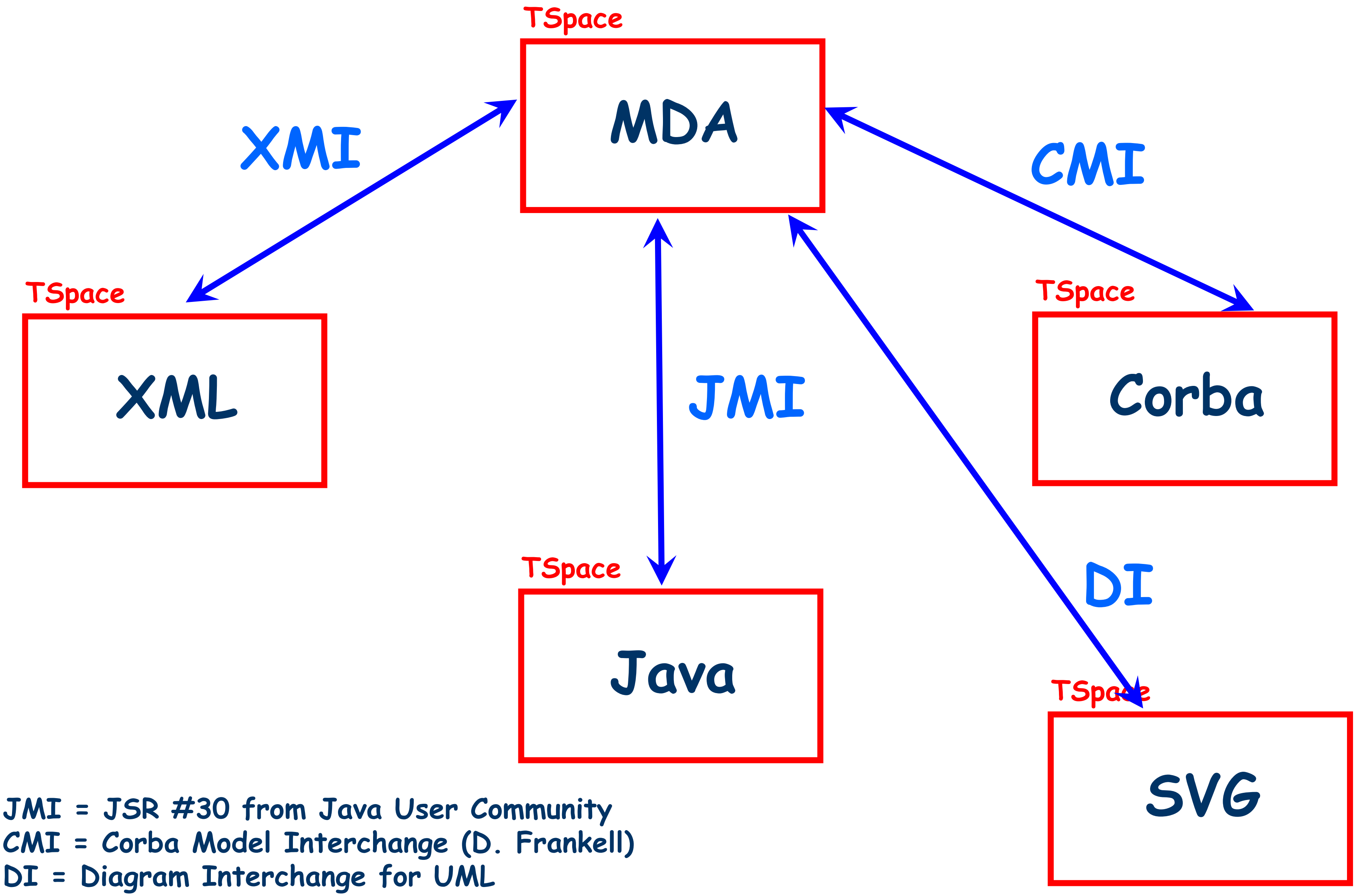
[2]

JavaML document



Each of these representations may be more convenient to perform some operation on the program.

# Collaborations between TSpaces: Projections



JMI = JSR #30 from Java User Community
CMI = Corba Model Interchange (D. Frankell)
DI = Diagram Interchange for UML

# Models revisited

- Everything is a model
  - A $\lambda$-model
  - $\lambda$ meaning the specific TS
  - An XML document is an XML-model
  - A Java source program is a Java-model
  - An UML model is a MDA-model
  - etc.
- Each TS is rooted in a metametamodel defining its representation scheme
- Distinguish between intra-space and inter-space operations

## Example: Databases TS

> “By model we mean a complex structure that represents a design artifact, such as a relational schema, object-oriented interface, UML model, XML DTD, web-site schema, semantic network, complex document, or software configuration. Many uses of models involve managing changes in models and transformations of data from one model into another. These uses require an explicit representation of mappings between models. We propose to make database systems easier to use for these applications by making model and model mapping first-class objects with special operations that simplify their use. We call this capacity model management.”
>
> P.A. Bernstein, A.L. Levy & R.A. Pottinger MSR-TR-2000-53

Agenda

# What we learn from OT and why it is not the solution

# Software factories

"…

The software industry remains reliant on the craftsmanship of skilled individuals engaged in labor intensive manual tasks. However, growing pressure to reduce cost and time to market and to improve software quality may catalyze a transition to more automated methods. We look at how the software industry may be industrialized, and we describe technologies that might be used to support this vision.

…

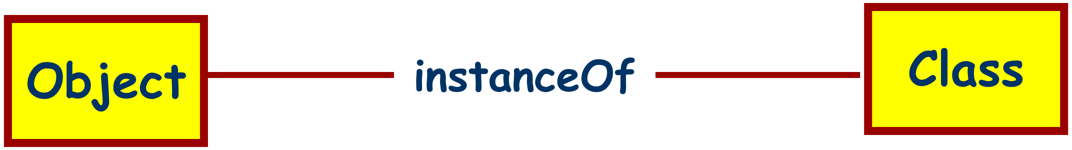
We suggest that the current software development paradigm, based on object orientation, may have reached the point of exhaustion, and we propose a model for its successor.

…"

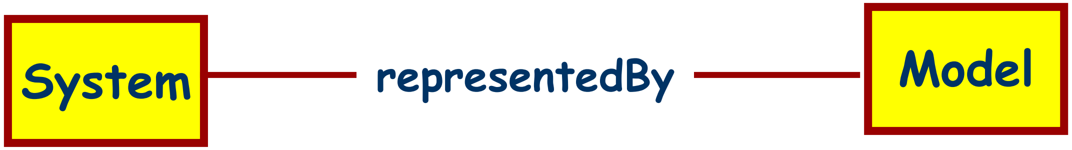
J. Greenfield and K. Short

# From objects to models

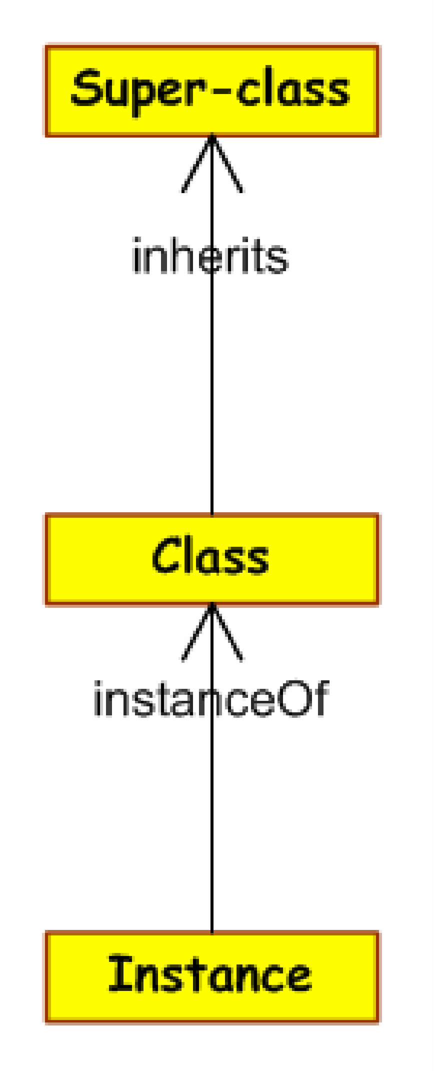
- "Everything is an object" was one of the strongest technology improvement driving principles in the last twenty years
  - As long as this principle was followed, steady progresses were achieved

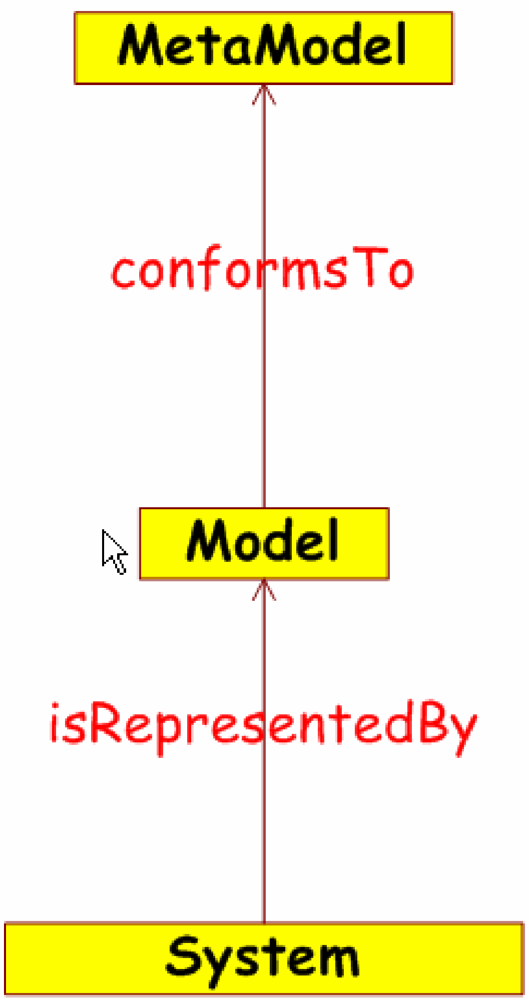


- "Everything is a model" is  a current driving principle for the MDE/MDA
  - As long as we follow this principle, steady progresses may be achieved



# Paradigm Change

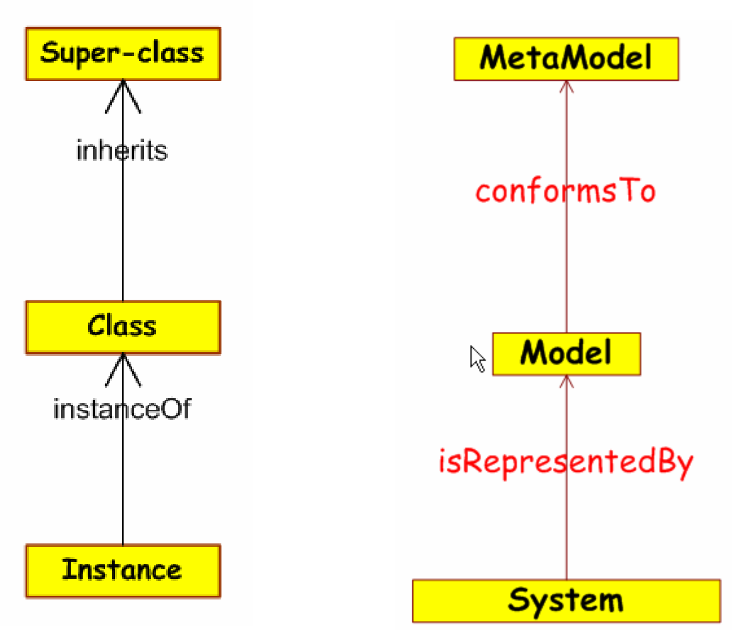




# More precisely

- The set of central relations in model-driven engineering and object-oriented technology are basically different.

The two central relations in object-oriented technology



The two central relations in model-driven engineering

- Caution: Trying to use one set of relations, as a central interpretation, in the wrong context, may lead to serious problems :
  - Example 1: A model being "an instance of" a metamodel
  - Example 2: A UML profile "inheriting from" the UML metamodel
  - etc.

# Unification principle

- In the 1980's: Everything is an object
- In the 2000's: Everything is a model

Both assertions are "basic engineering postulates".

The world is not constituted of objects,
but this helps considerably when building our systems,
that are partial images of the real or imaginary world,
if we understand clearly what is an object
from an engineering point of view.

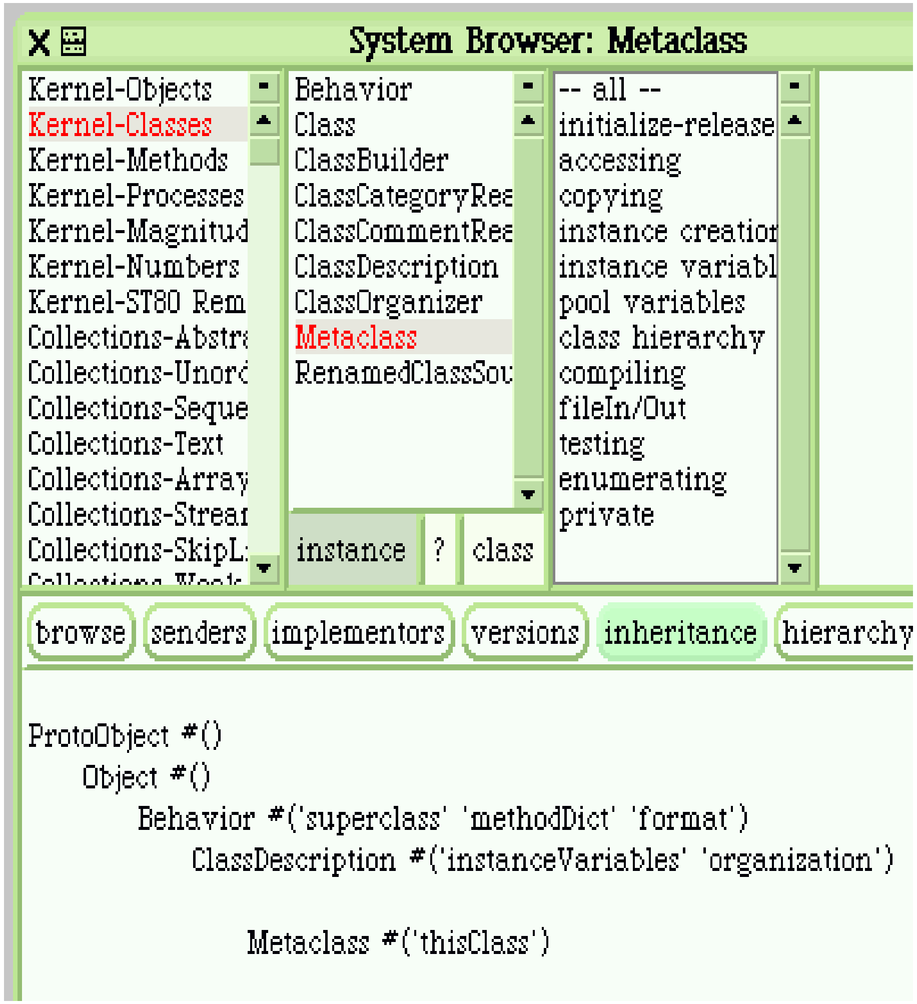
The same applies now to models.

# Everything is an object?

- Everything is an object was a goal statement that was made in the 80's
- Object technology stopped making progress as soon as this objective was given up
- Three examples in Smalltalk:
    - Classes as objects
    - Messages as objects
    - Integers as objects

# Classes as objects

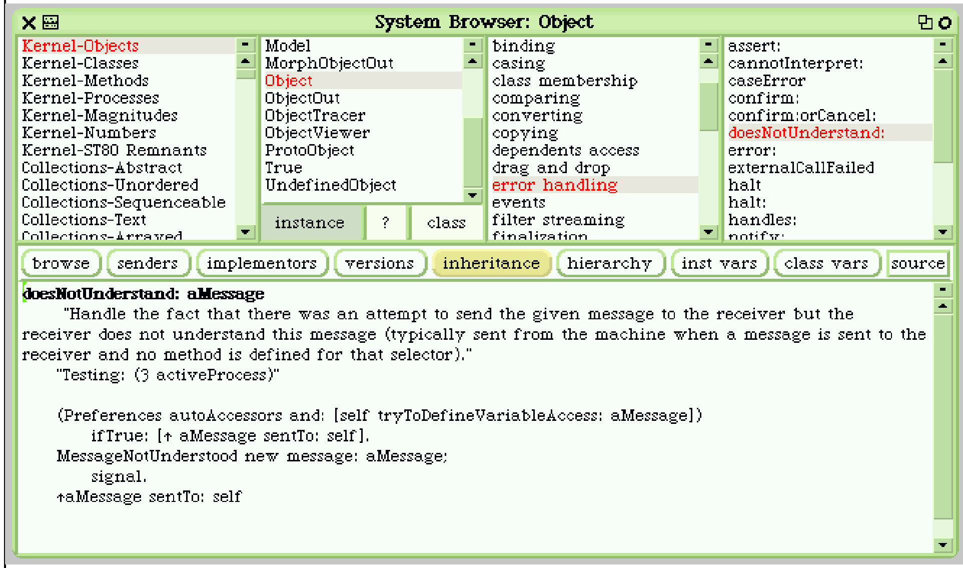
- Definition: an object is an instance of a class
- Step #1: a class is an object
- Question: what is the class of a class?
- Step #2: a class of a class is a meta-class
- Question: what is the class of a meta-class?
- Step #3: several answers
  - Smalltalk
  - CLOS
  - etc.



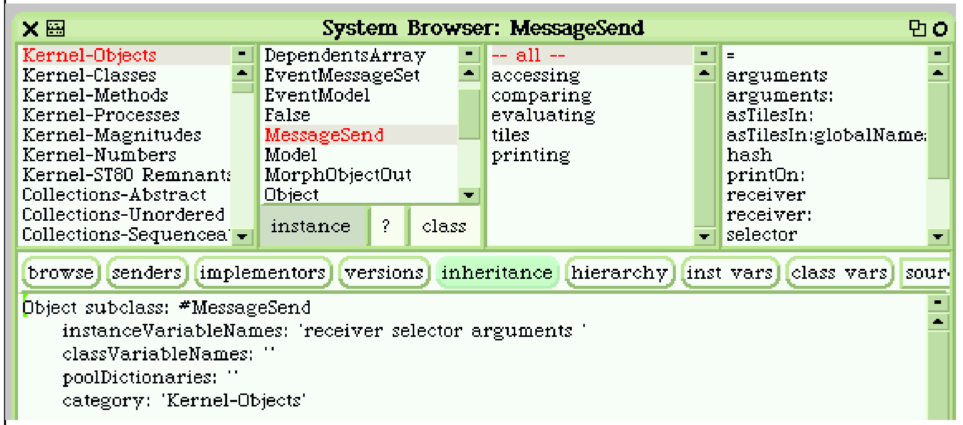
```
ProtoObject #()
    Object #()
        Behavior #('superclass' 'methodDict' 'format')
            ClassDescription #('instanceVariables' 'organization')

                Metaclass #('thisClass')
```

# Messages as objects

- Definition: an object communicates with other objects by sending messages
- Step #1: a message is an object
- Question: huge performance problems?
- Step #2: in certain situations, a message may be "reified" as an object
  - When a message *m* is sent to object *x*, and is not recognized by object *x*, then the system automatically sends a *#doesNotUnderstand:* message to object *x* with the original message *m* as an argument.
  - Such a reified message is an instance of the *MessageSend* class



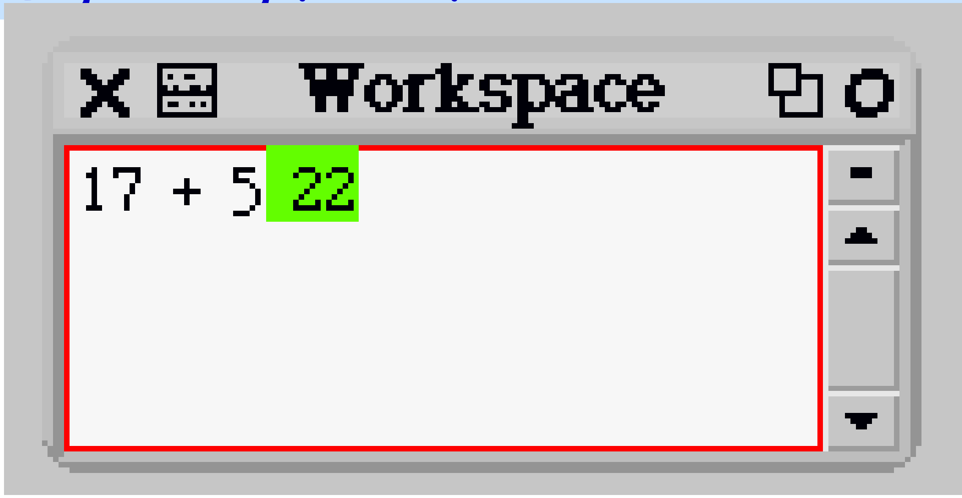
```
doesNotUnderstand: aMessage
	"Handle the fact that there was an attempt to send the given message to the receiver but the
receiver does not understand this message (typically sent from the machine when a message is sent to the
receiver and no method is defined for that selector)."
	"Testing: (3 activeProcess)"

	(Preferences autoAccessors and: [self tryToDefineVariableAccess: aMessage])
		ifTrue: [↑ aMessage sentTo: self].
	MessageNotUnderstood new message: aMessage;
		signal.
	↑aMessage sentTo: self
```



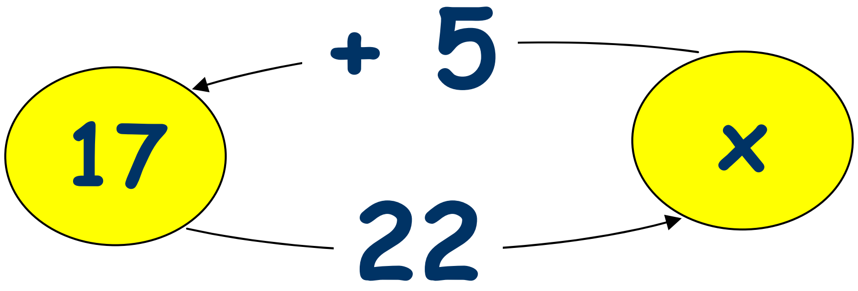
```
Object subclass: #MessageSend
	instanceVariableNames: 'receiver selector arguments '
	classVariableNames: ''
	poolDictionaries: ''
	category: 'Kernel-Objects'
```

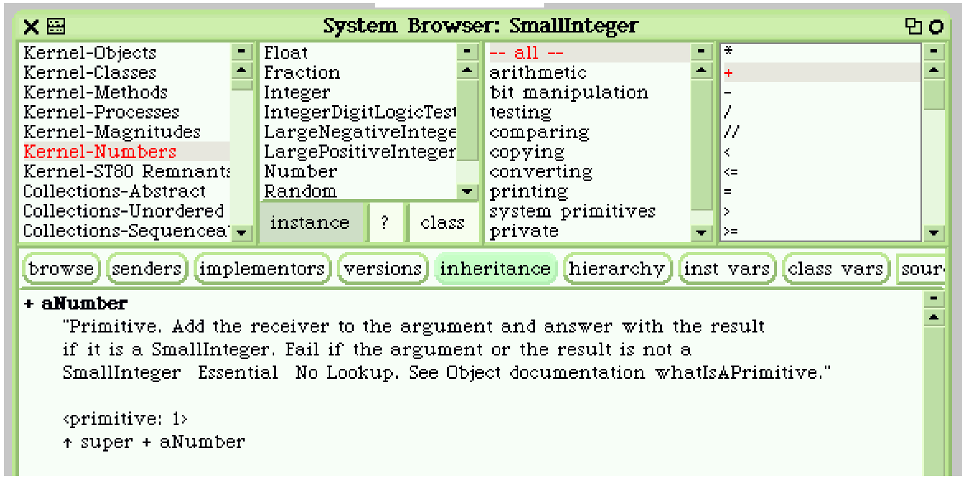
# Integers as objects

- Definition: an Integer (or a String, or a Boolean, etc.) is an object
- Question: How to implement efficiently the following situation?
  - **when object 17, instance of the *SmallInteger* class, receives from object x, the message of selector #+ with argument 5 it sends backs object 22 to object x.**
- Solution: method + in *SmallInteger* class will be marked as a special primitive method with direct short-circuit implementation.



```
17 + 5 22
```





Kernel-Objects
Kernel-Classes
Kernel-Methods
Kernel-Processes
Kernel-Magnitudes
Kernel-Numbers
Kernel-ST80 Remnants
Collections-Abstract
Collections-Unordered
Collections-Sequencea

Float
Fraction
Integer
IntegerDigitLogicTest
LargeNegativeIntege
LargePositiveInteger
Number
Random

-- all --
arithmetic
bit manipulation
testing
comparing
copying
converting
printing
system primitives
private

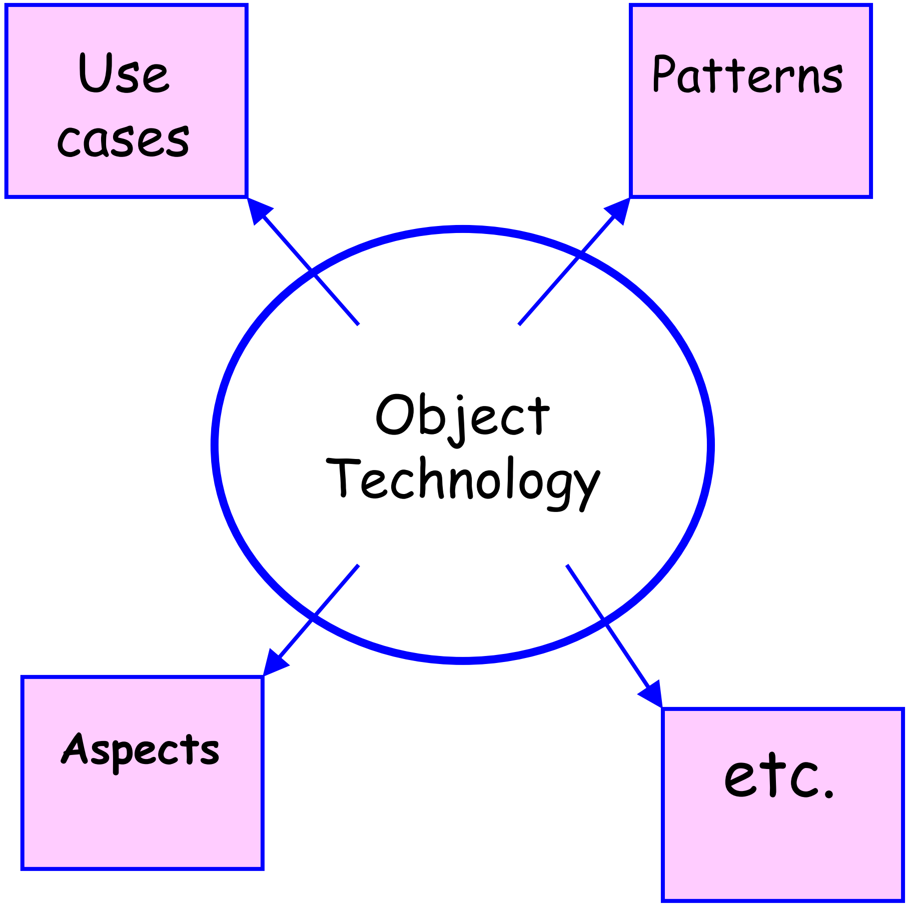
```
*
+
-
/
//
<
<=
=
>
>=
```

```
+ aNumber
    "Primitive. Add the receiver to the argument and answer with the result
    if it is a SmallInteger. Fail if the argument or the result is not a
    SmallInteger  Essential  No Lookup. See Object documentation whatIsAPrimitive."

    <primitive: 1>
    ↑ super + aNumber
```

# Limits of object technology

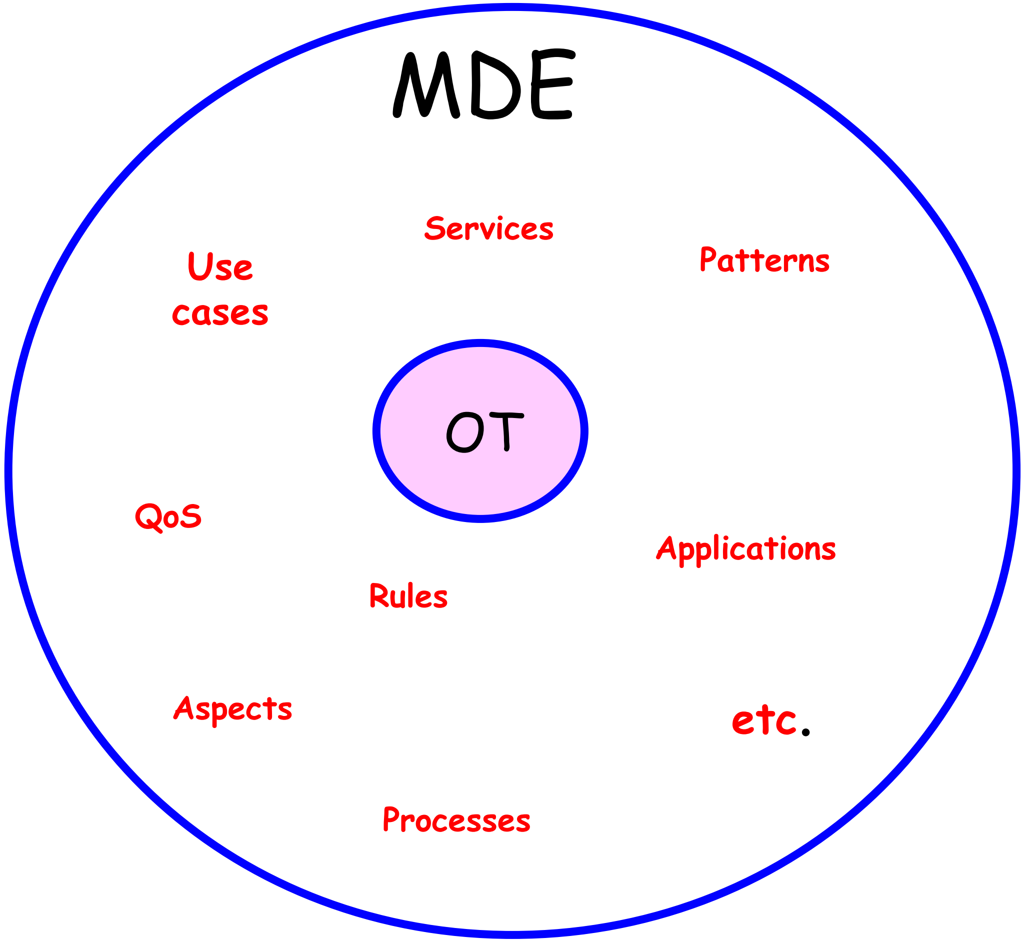
- Many problems with object technology
  - Design patterns are not objects
  - Use cases are not objects
  - Aspects are not objects
  - "Plug-ins" are not objects
  - Methods are not objects
  - etc.
- The missed marriages:
  - Objects and databases
  - Objects and processes
  - Objects and transactions
  - Objects and the Web
  - etc.

# Paradigm evolution



- OT (Object Technology) has got to answer many difficult problems since its initial industrial discovery in the 80's.
- Generally OT did not succeed very well in finding good <u>internal solutions</u> to these challenges.
- *As* a consequence, the proposed solutions are often ad-hoc, informal, and even baroque. They are difficult to generalize and hard to combine. They often don't scale up.

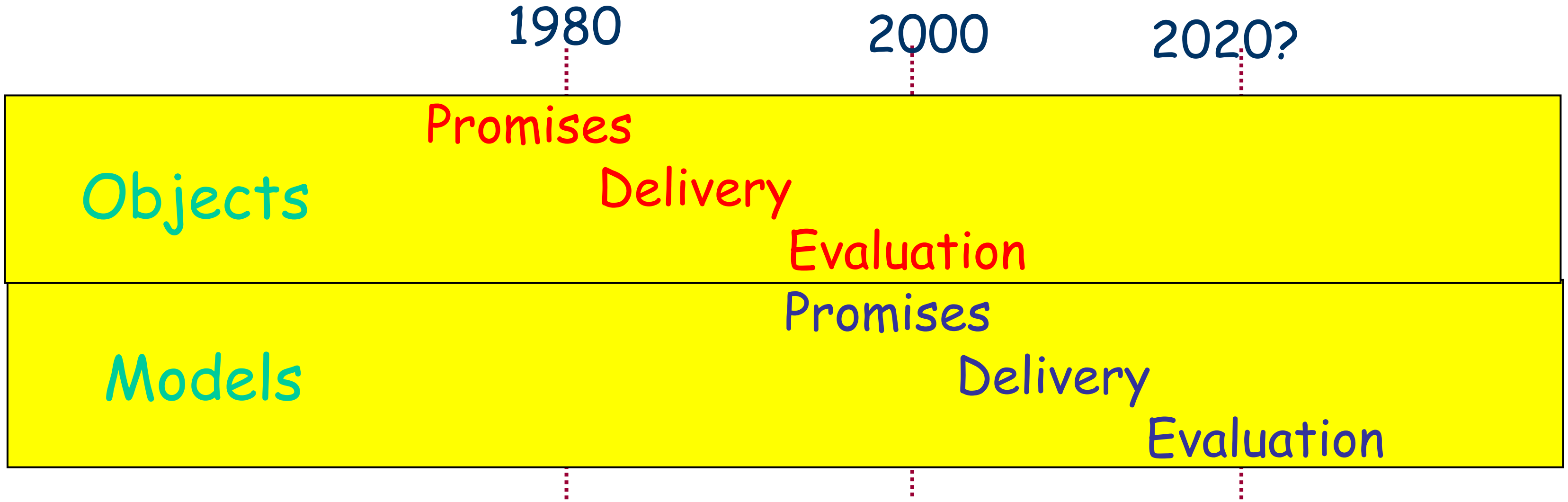
# Paradigm evolution



- MDE (Model Driven Engineering) seems to be in a position to meet some of these challenges to which OT was not able to find an internal solution.
- Among these we may quote:
  - Aspect separation
  - Homogeneous handling of functional and non-functional attributes
  - Integration of different paradigms (rules, services, processes, architecture, etc.)
- MDE seems able to subsume OT and to offer a realistic migration path from present object and component solutions to more ambitious regular, evolutive and scalable organizations.

# Summary

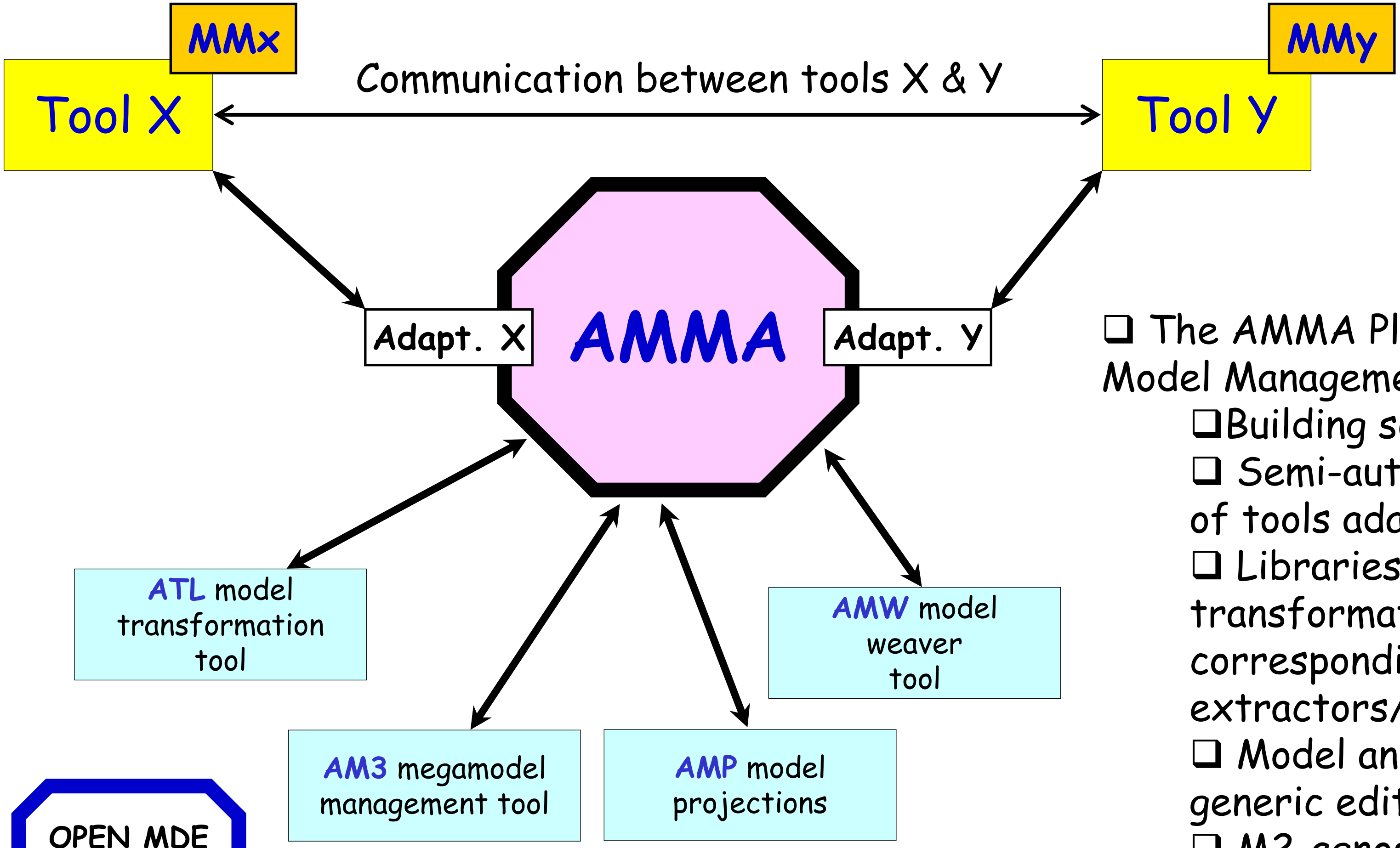
- Object technology realized some promises but failed to achieve others
  - Stopping the search for generality by unification may be one of the causes for this
- Model engineering is making many promises today
  - Will it be able to deliver correspondingly?
  - Sticking with the principle that "everything is a model" seems a good way to make progresses

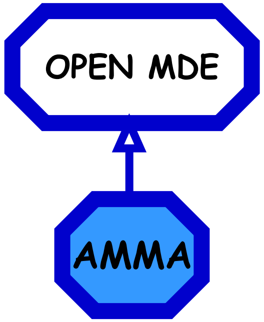


Agenda

# Towards an open MDE platform?

# Many groups building Open MDE platforms

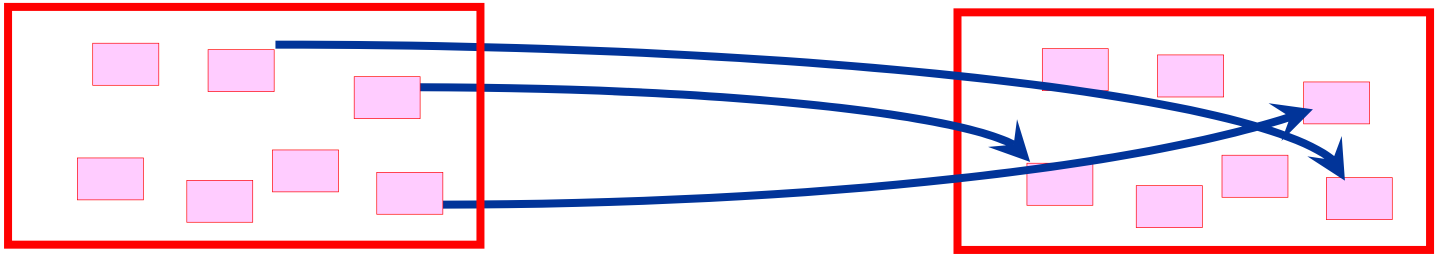

MMx
Tool X
Communication between tools X & Y
MMy
Tool Y
Adapt. X
AMMA
Adapt. Y
ATL model transformation tool
AMW model weaver tool
AM3 megamodel management tool
AMP model projections



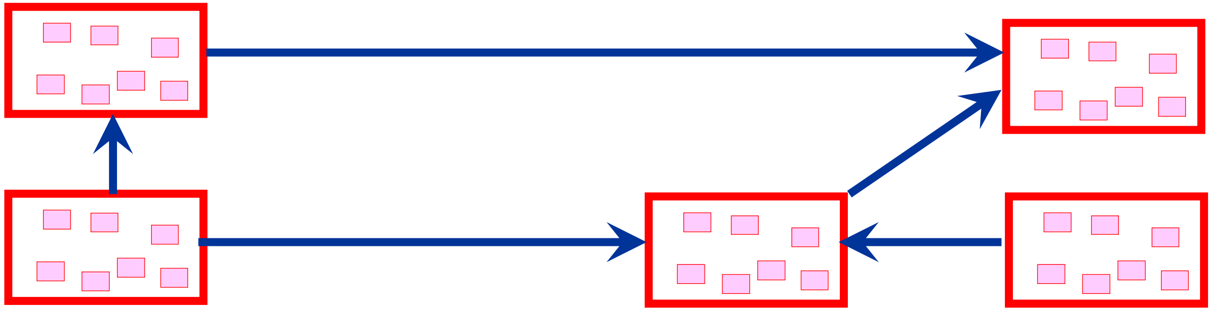
- ❑ The AMMA Platform (ATLAS Model Management Architecture)
  - ❑ Building semantic bridges
  - ❑ Semi-automatic generation of tools adapters
  - ❑ Libraries of transformation and corresponding extractors/injectors
  - ❑ Model and metamodel generic editors
  - ❑ M2-agnostic and M3-agnostic tools
  - ❑ Application to data-intensive systems and software modernization

# Atlas Model Management Architecture

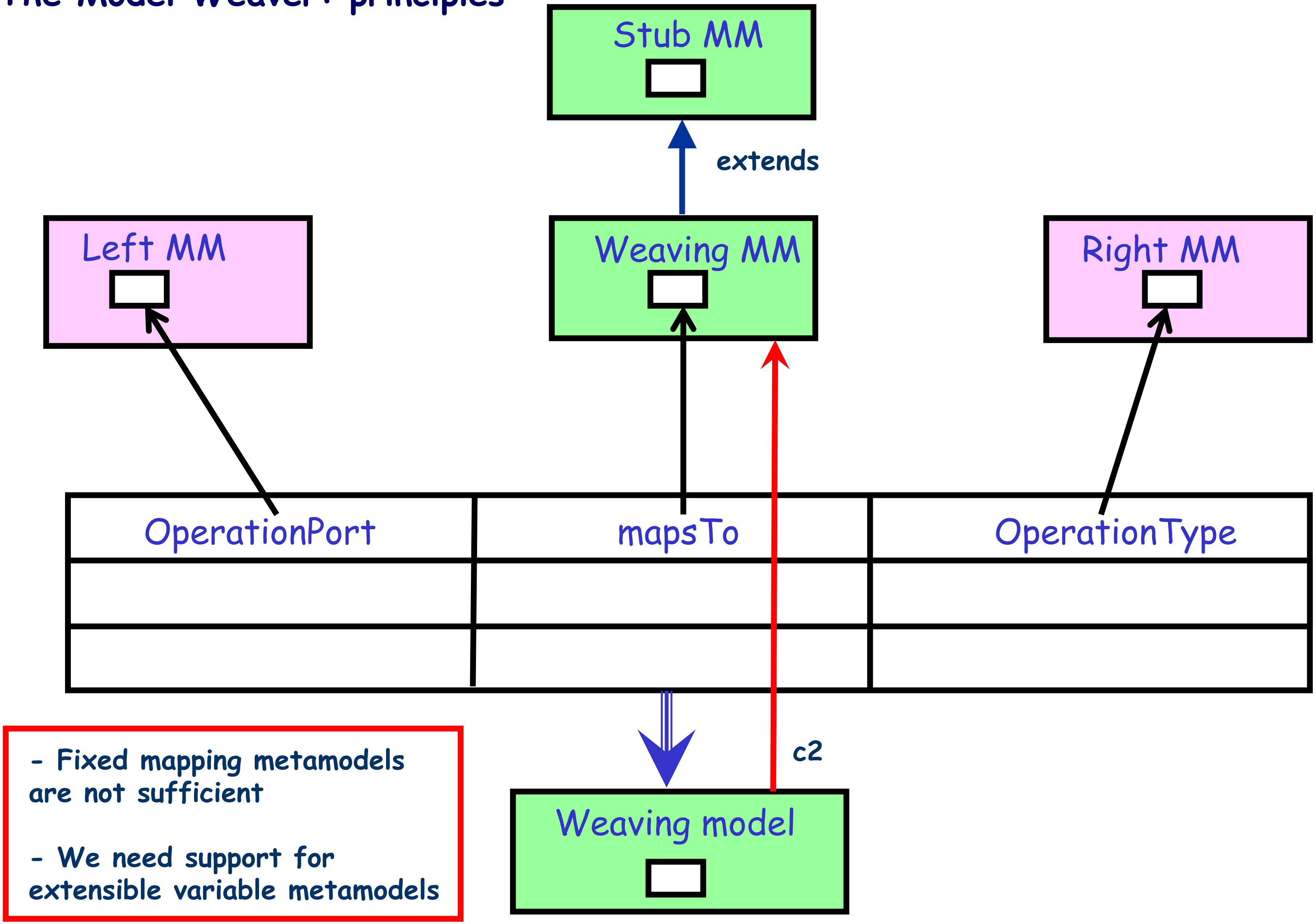
- Modeling in the small
  - Working at the level of model and metamodel elements



- Modeling in the large
  - Working with models and metamodels as global entities, for what they represent, and their mutual relations, independently of their content
  - A megamodel is a model which elements represents models, metamodels and other global entities (ako model registry with metadata on models and metamodels). A megamodel has a metamodel.



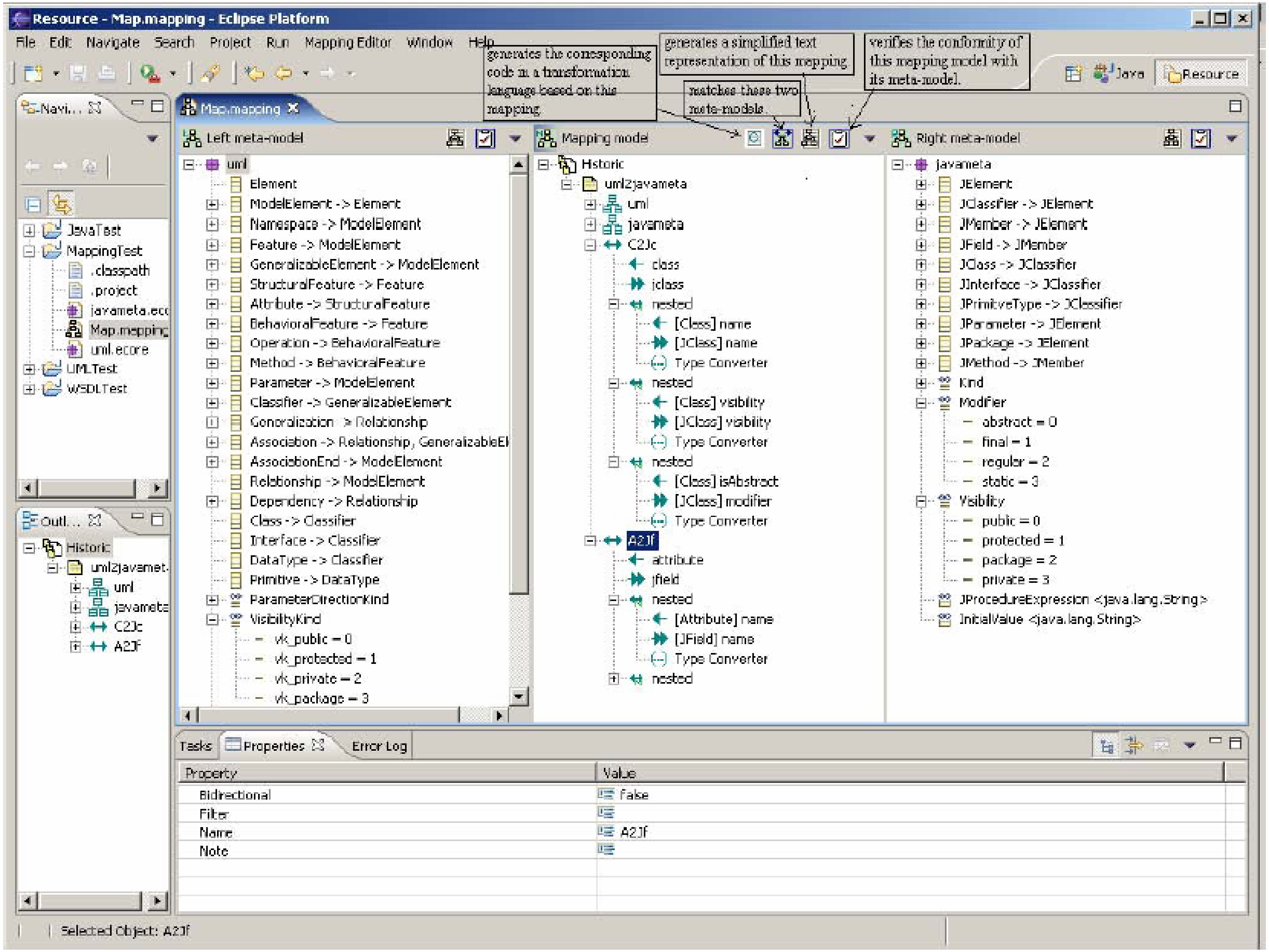
# The Model Weaver: principles



- Fixed mapping metamodels are not sufficient

- We need support for extensible variable metamodels

# The Model Weaver: first prototype


Resource - Map.mapping - Eclipse Platform
File Edit Navigate Search Project Run Mapping Editor Window Help
Java
Resource
Navi...
Map.mapping
Left meta-model
Mapping model
Right meta-model
Outl...
Tasks
Properties
Error Log

generates the corresponding code in a transformation language based on this mapping

generates a simplified text representation of this mapping

matches these two meta-models

verifies the conformity of this mapping model with its meta-model.

```
JavaTest
MappingTest
    .classpath
    .project
    javameta.ecc
    Map.mapping
    uml.ecore
UMLTest
WSDLTest
```

```
uml
    Element
    ModelElement -> Element
    Namespace -> ModelElement
    Feature -> ModelElement
    GeneralizableElement -> ModelElement
    StructuralFeature -> Feature
    Attribute -> StructuralFeature
    BehavioralFeature -> Feature
    Operation -> BehavioralFeature
    Method -> BehavioralFeature
    Parameter -> ModelElement
    Classifier -> GeneralizableElement
    Generalization -> Relationship
    Association -> Relationship, GeneralizableEl
    AssociationEnd -> ModelElement
    Relationship -> ModelElement
    Dependency -> Relationship
    Class -> Classifier
    Interface -> Classifier
    DataType -> Classifier
    Primitive -> DataType
    ParameterDirectionKind
    VisibilityKind
        vk_public = 0
        vk_protected = 1
        vk_private = 2
        vk_package = 3
```

```
Historic
    uml2javameta
        uml
        javameta
        C2Jc
            class
            jclass
            nested
                [Class] name
                [JClass] name
                Type Converter
            nested
                [Class] visibility
                [JClass] visibility
                Type Converter
            nested
                [Class] isAbstract
                [JClass] modifier
                Type Converter
        A2Jf
            attribute
            jfield
            nested
                [Attribute] name
                [JField] name
                Type Converter
            nested
```

```
javameta
    JElement
    JClassifier -> JElement
    JMember -> JElement
    JField -> JMember
    JClass -> JClassifier
    JInterface -> JClassifier
    JPrimitveType -> JClassifier
    JParameter -> JElement
    JPackage -> JElement
    JMethod -> JMember
    Kind
    Modifier
        abstract = 0
        final = 1
        regular = 2
        static = 3
    Visibility
        public = 0
        protected = 1
        package = 2
        private = 3
    JProcedureExpression <java.lang.String>
    InitialValue <java.lang.String>
```

```
Historic
    uml2javamet
        uml
        javameta
        C2Jc
        A2Jf
```

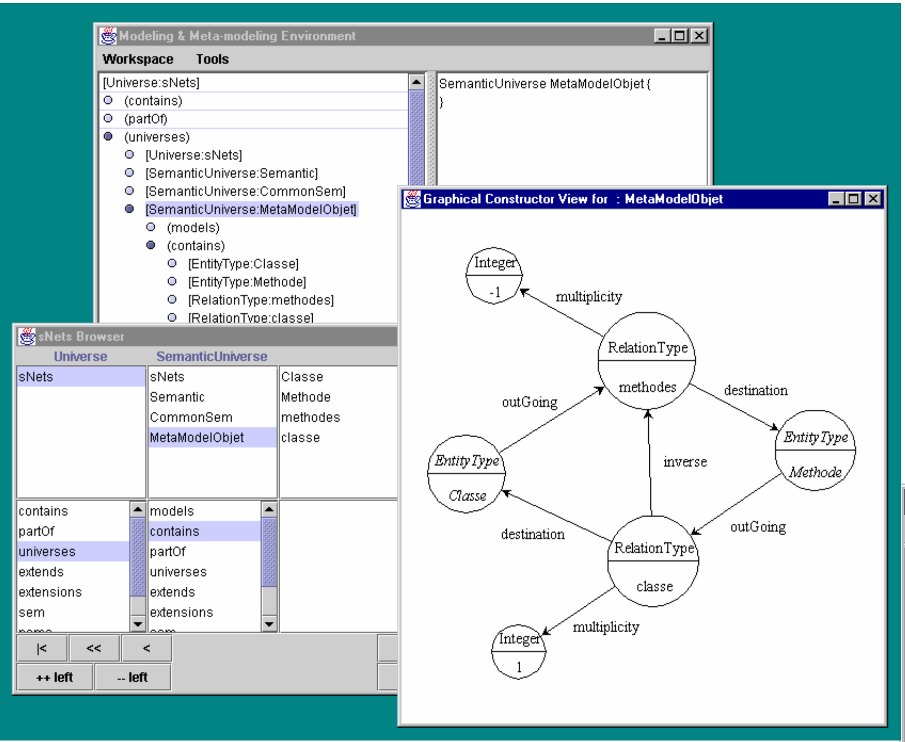
| Property | Value |
|---|---|
| Bidirectional | false |
| Filter | |
| Name | A2Jf |
| Note | |

Selected Object: A2Jf

Agenda

# Applications of MDE to software modernization

# The starting point: Semantor Explorer (©Sodifrance)



Modeling & Meta-modeling Environment

```
[Universe:sNets]
○ (contains)
○ (partOf)
● (universes)
    ○ [Universe:sNets]
    ○ [SemanticUniverse:Semantic]
    ○ [SemanticUniverse:CommonSem]
    ● [SemanticUniverse:MetaModelObjet]
        ○ (models)
        ● (contains)
            ○ [EntityType:Classe]
            ○ [EntityType:Methode]
            ○ [RelationType:methodes]
            ○ [RelationType:classe]
```

```
SemanticUniverse MetaModelObjet {
}
```

sNets Browser

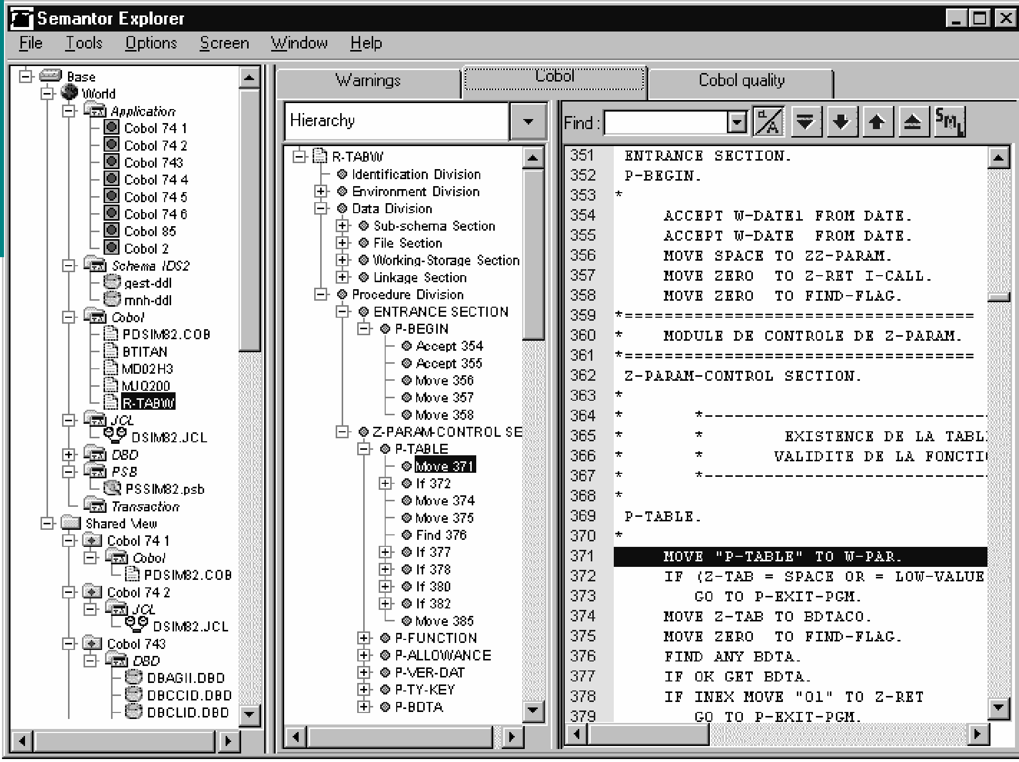
| Universe | SemanticUniverse | |
|---|---|---|
| sNets | sNets | Classe |
| | Semantic | Methode |
| | CommonSem | methodes |
| | MetaModelObjet | classe |

| | |
|---|---|
| contains | models |
| partOf | contains |
| universes | partOf |
| extends | universes |
| extensions | extends |
| sem | extensions |

Graphical Constructor View for : MetaModelObjet

Technical view (sNets)

User view (COBOL)



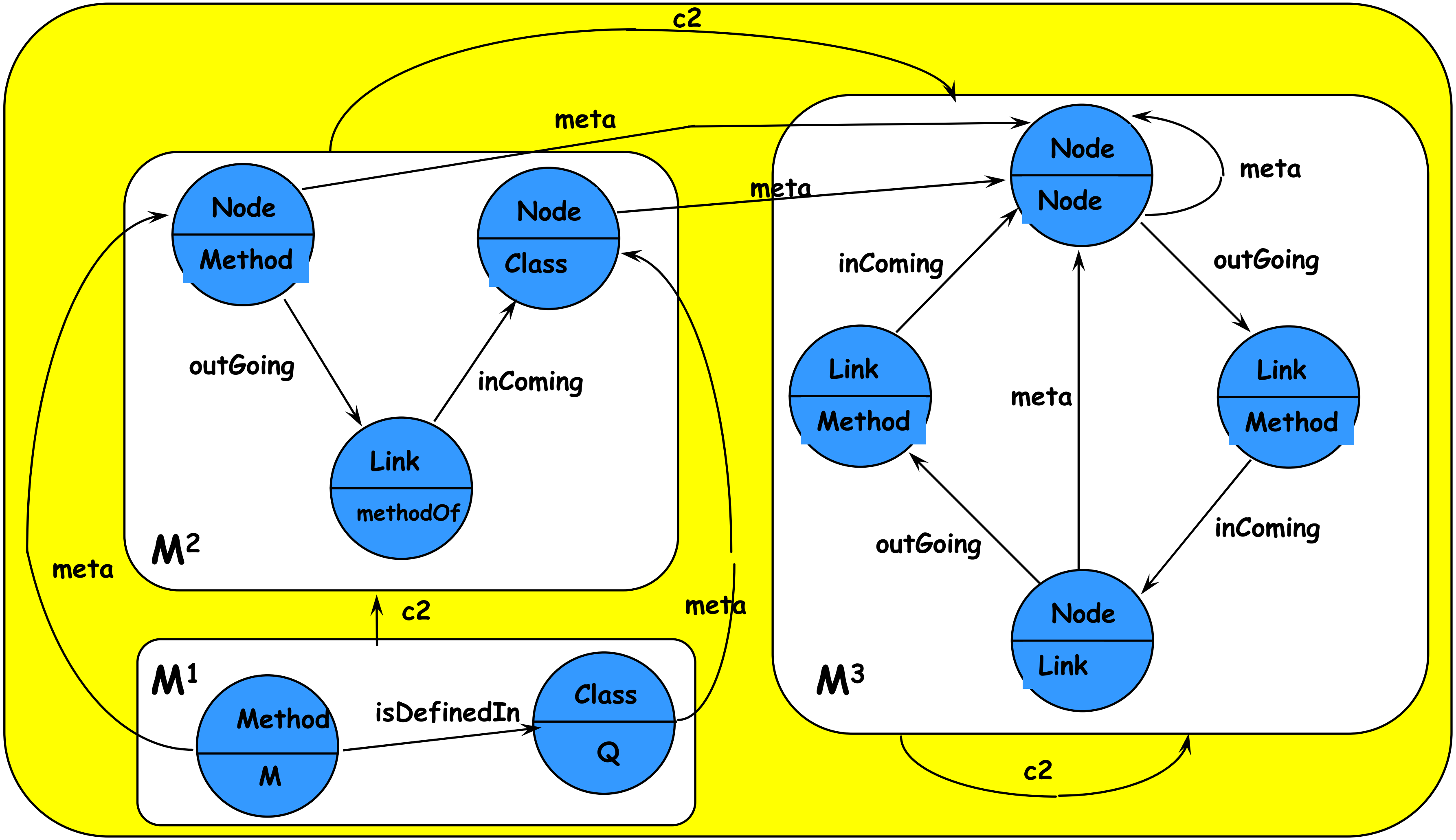
Semantor Explorer

```
Base
  World
    Application
      Cobol 74 1
      Cobol 74 2
      Cobol 743
      Cobol 74 4
      Cobol 74 5
      Cobol 74 6
      Cobol 85
      Cobol 2
    Schema IDS2
      gest-ddl
      mnh-ddl
    Cobol
      PDSIM82.COB
      BTITAN
      MD02H3
      MJQ200
      R-TABW
    JCL
      DSIM82.JCL
    DBD
    PSB
      PSSIM82.psb
    Transaction
  Shared View
    Cobol 74 1
      Cobol
        PDSIM82.COB
    Cobol 74 2
      JCL
        DSIM82.JCL
    Cobol 743
      DBD
        DBAGII.DBD
        DBCCID.DBD
        DBCLID.DBD
```

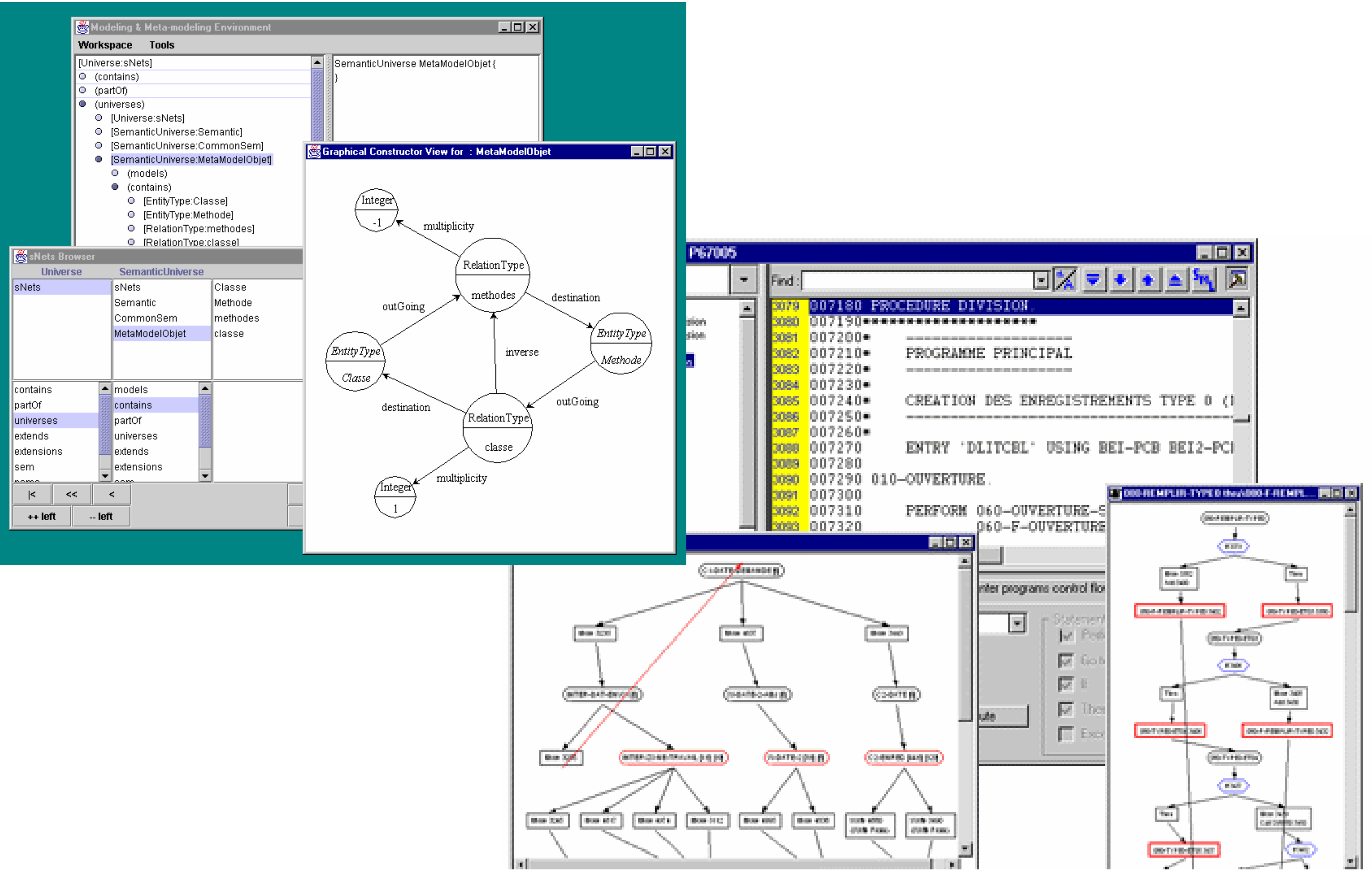
```
R-TABW
  Identification Division
  Environment Division
  Data Division
    Sub-schema Section
    File Section
    Working-Storage Section
    Linkage Section
  Procedure Division
    ENTRANCE SECTION
      P-BEGIN
        Accept 354
        Accept 355
        Move 356
        Move 357
        Move 358
    Z-PARAM-CONTROL SE
      P-TABLE
        Move 371
        If 372
        Move 374
        Move 375
        Find 376
        If 377
        If 378
        If 380
        If 382
        Move 385
      P-FUNCTION
      P-ALLOWANCE
      P-VER-DAT
      P-TY-KEY
      P-BDTA
```

```
351   ENTRANCE SECTION.
352   P-BEGIN.
353  *
354       ACCEPT W-DATE1 FROM DATE.
355       ACCEPT W-DATE  FROM DATE.
356       MOVE SPACE TO ZZ-PARAM.
357       MOVE ZERO  TO Z-RET I-CALL.
358       MOVE ZERO  TO FIND-FLAG.
359  *====================================
360  *    MODULE DE CONTROLE DE Z-PARAM.
361  *====================================
362   Z-PARAM-CONTROL SECTION.
363  *
364  *       *-----------------------------
365  *       *         EXISTENCE DE LA TABL
366  *       *        VALIDITE DE LA FONCTI
367  *       *-----------------------------
368  *
369   P-TABLE.
370  *
371       MOVE "P-TABLE" TO W-PAR.
372       IF (Z-TAB = SPACE OR = LOW-VALUE
373          GO TO P-EXIT-PGM.
374       MOVE Z-TAB TO BDTACO.
375       MOVE ZERO  TO FIND-FLAG.
376       FIND ANY BDTA.
377       IF OK GET BDTA.
378       IF INEX MOVE "01" TO Z-RET
379          GO TO P-EXIT-PGM.
```

# M1, M2 and M3 spaces in sNets



# Multiple views on the homogeneous representation system of sNets



[Universe:sNets]
○ (contains)
○ (partOf)
● (universes)
  ○ [Universe:sNets]
  ○ [SemanticUniverse:Semantic]
  ○ [SemanticUniverse:CommonSem]
  ● [SemanticUniverse:MetaModelObjet]
    ○ (models)
    ● (contains)
      ○ [EntityType:Classe]
      ○ [EntityType:Methode]
      ○ [RelationType:methodes]
      ○ [RelationType:classe]

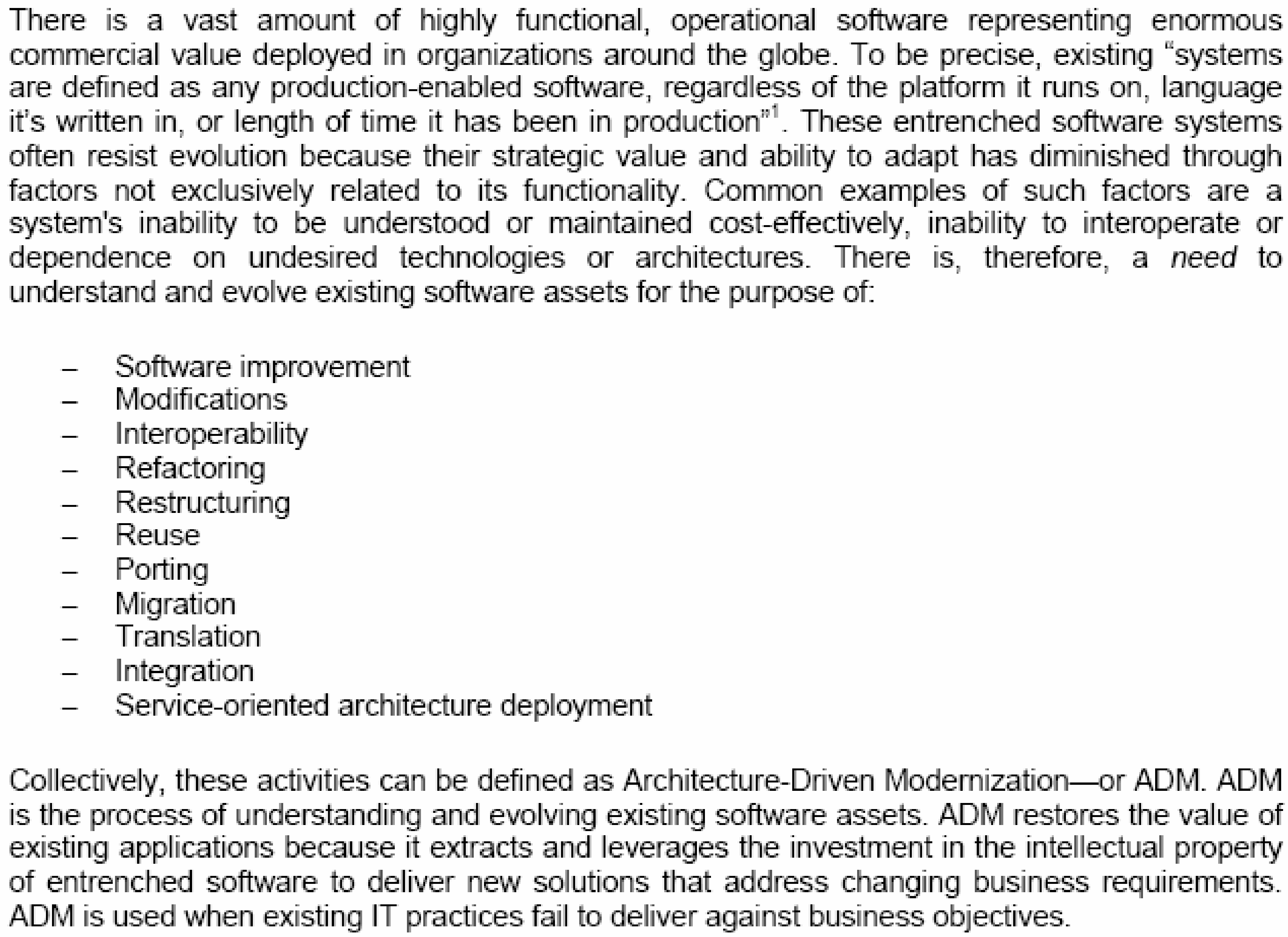
```
SemanticUniverse MetaModelObjet {
}
```

| Universe | SemanticUniverse | |
|---|---|---|
| sNets | sNets | Classe |
| | Semantic | Methode |
| | CommonSem | methodes |
| | MetaModelObjet | classe |

| | |
|---|---|
| contains | models |
| partOf | contains |
| universes | partOf |
| extends | universes |
| extensions | extends |
| sem | extensions |

```
3079 007180 PROCEDURE DIVISION.
3080 007190*********************
3081 007200*      ==================
3082 007210*      PROGRAMME PRINCIPAL
3083 007220*      ==================
3084 007230*
3085 007240*      CREATION DES ENREGISTREMENTS TYPE 0 (I
3086 007250*      -------------------------------------
3087 007260*
3088 007270       ENTRY 'DLITCBL' USING BEI-PCB BEI2-PCI
3089 007280
3090 007290 010-OUVERTURE.
3091 007300
3092 007310       PERFORM 060-OUVERTURE-S
3093 007320               060-F-OUVERTURE
```

# OMG ADM (from goal statement)

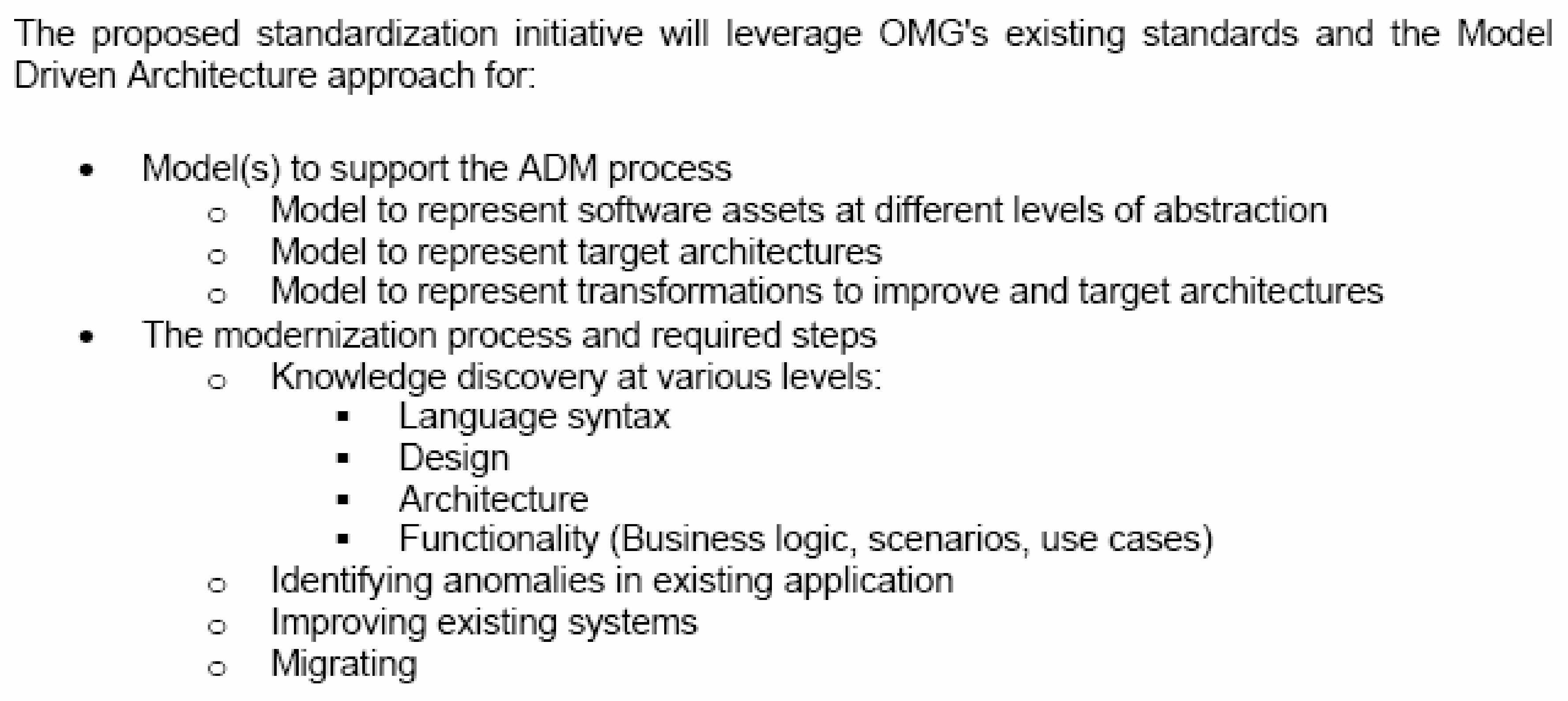


There is a vast amount of highly functional, operational software representing enormous commercial value deployed in organizations around the globe. To be precise, existing “systems are defined as any production-enabled software, regardless of the platform it runs on, language it's written in, or length of time it has been in production”[1]. These entrenched software systems often resist evolution because their strategic value and ability to adapt has diminished through factors not exclusively related to its functionality. Common examples of such factors are a system's inability to be understood or maintained cost-effectively, inability to interoperate or dependence on undesired technologies or architectures. There is, therefore, a *need* to understand and evolve existing software assets for the purpose of:

- Software improvement
- Modifications
- Interoperability
- Refactoring
- Restructuring
- Reuse
- Porting
- Migration
- Translation
- Integration
- Service-oriented architecture deployment

Collectively, these activities can be defined as Architecture-Driven Modernization—or ADM. ADM is the process of understanding and evolving existing software assets. ADM restores the value of existing applications because it extracts and leverages the investment in the intellectual property of entrenched software to deliver new solutions that address changing business requirements. ADM is used when existing IT practices fail to deliver against business objectives.

# OMG ADM



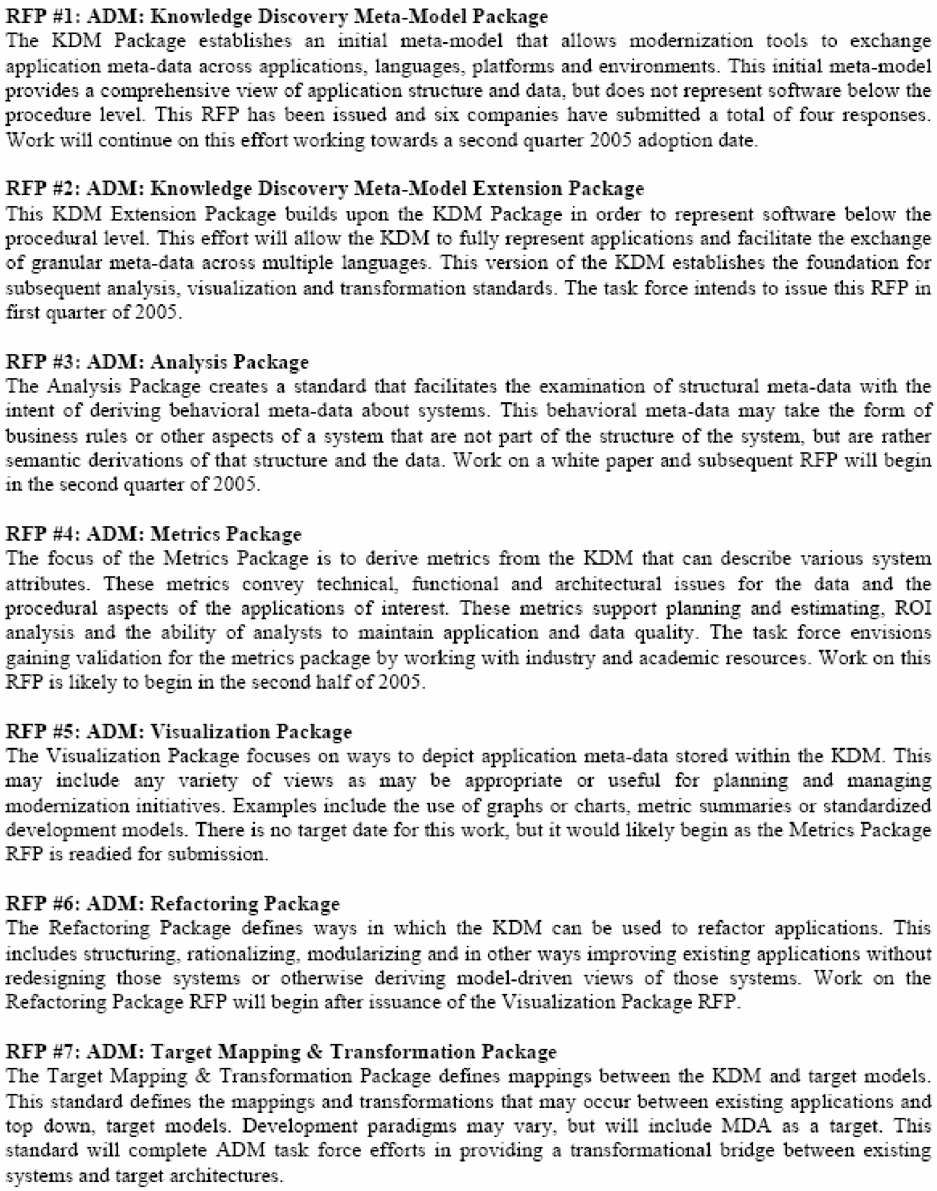
The proposed standardization initiative will leverage OMG's existing standards and the Model Driven Architecture approach for:

- Model(s) to support the ADM process
  - Model to represent software assets at different levels of abstraction
  - Model to represent target architectures
  - Model to represent transformations to improve and target architectures
- The modernization process and required steps
  - Knowledge discovery at various levels:
    - Language syntax
    - Design
    - Architecture
    - Functionality (Business logic, scenarios, use cases)
  - Identifying anomalies in existing application
  - Improving existing systems
  - Migrating

## OMG ADM

# ADM Roadmap

1. Knowledge Discovery MM
2. Knowledge Discovery MM extension
3. Analysis package
4. Metrics Package
5. Vizualization Package
6. Refactoring Package
7. Target Mapping and Transformation Package



**RFP #1: ADM: Knowledge Discovery Meta-Model Package**
The KDM Package establishes an initial meta-model that allows modernization tools to exchange application meta-data across applications, languages, platforms and environments. This initial meta-model provides a comprehensive view of application structure and data, but does not represent software below the procedure level. This RFP has been issued and six companies have submitted a total of four responses. Work will continue on this effort working towards a second quarter 2005 adoption date.

**RFP #2: ADM: Knowledge Discovery Meta-Model Extension Package**
This KDM Extension Package builds upon the KDM Package in order to represent software below the procedural level. This effort will allow the KDM to fully represent applications and facilitate the exchange of granular meta-data across multiple languages. This version of the KDM establishes the foundation for subsequent analysis, visualization and transformation standards. The task force intends to issue this RFP in first quarter of 2005.

**RFP #3: ADM: Analysis Package**
The Analysis Package creates a standard that facilitates the examination of structural meta-data with the intent of deriving behavioral meta-data about systems. This behavioral meta-data may take the form of business rules or other aspects of a system that are not part of the structure of the system, but are rather semantic derivations of that structure and the data. Work on a white paper and subsequent RFP will begin in the second quarter of 2005.

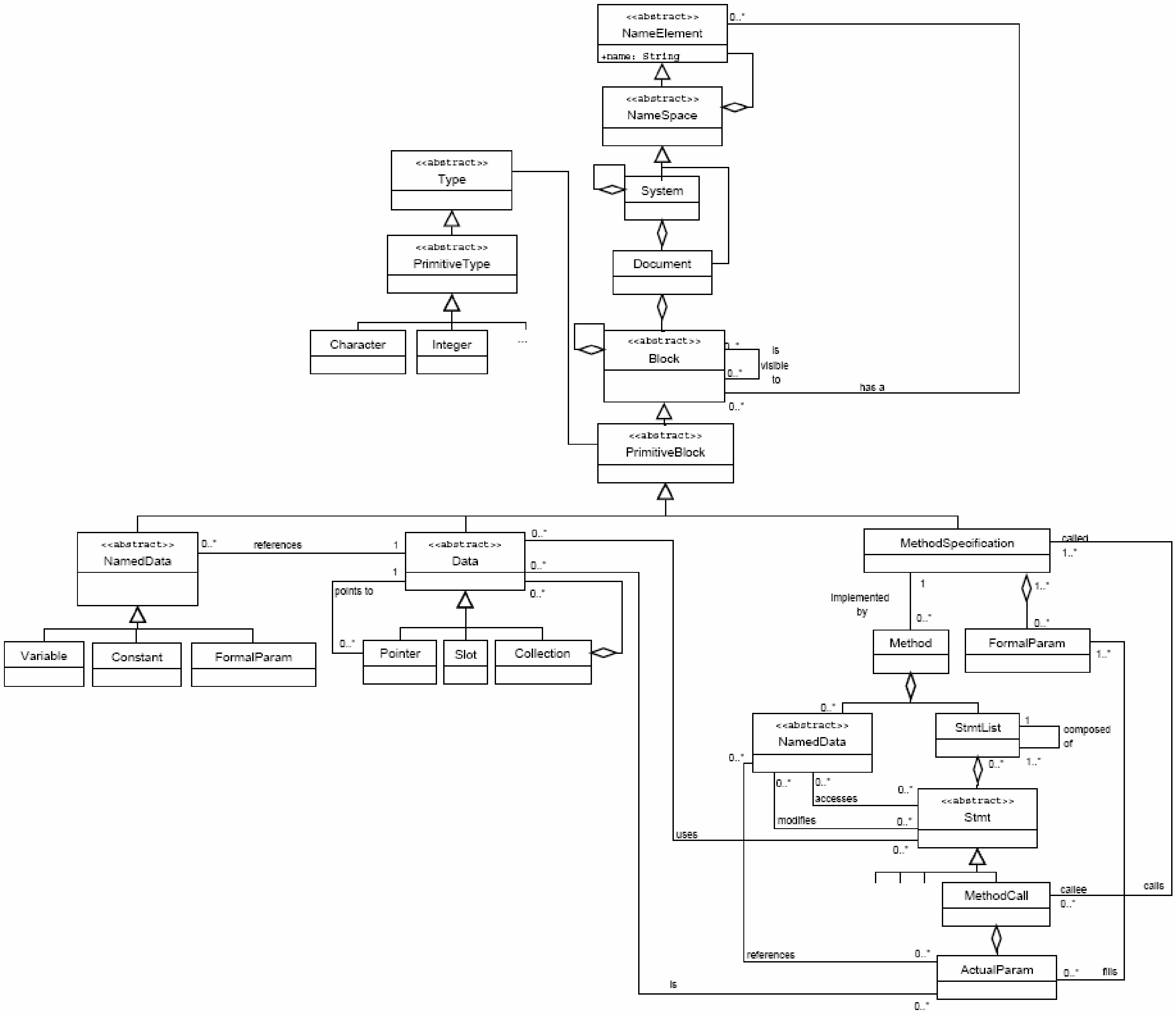
**RFP #4: ADM: Metrics Package**
The focus of the Metrics Package is to derive metrics from the KDM that can describe various system attributes. These metrics convey technical, functional and architectural issues for the data and the procedural aspects of the applications of interest. These metrics support planning and estimating, ROI analysis and the ability of analysts to maintain application and data quality. The task force envisions gaining validation for the metrics package by working with industry and academic resources. Work on this RFP is likely to begin in the second half of 2005.

**RFP #5: ADM: Visualization Package**
The Visualization Package focuses on ways to depict application meta-data stored within the KDM. This may include any variety of views as may be appropriate or useful for planning and managing modernization initiatives. Examples include the use of graphs or charts, metric summaries or standardized development models. There is no target date for this work, but it would likely begin as the Metrics Package RFP is readied for submission.

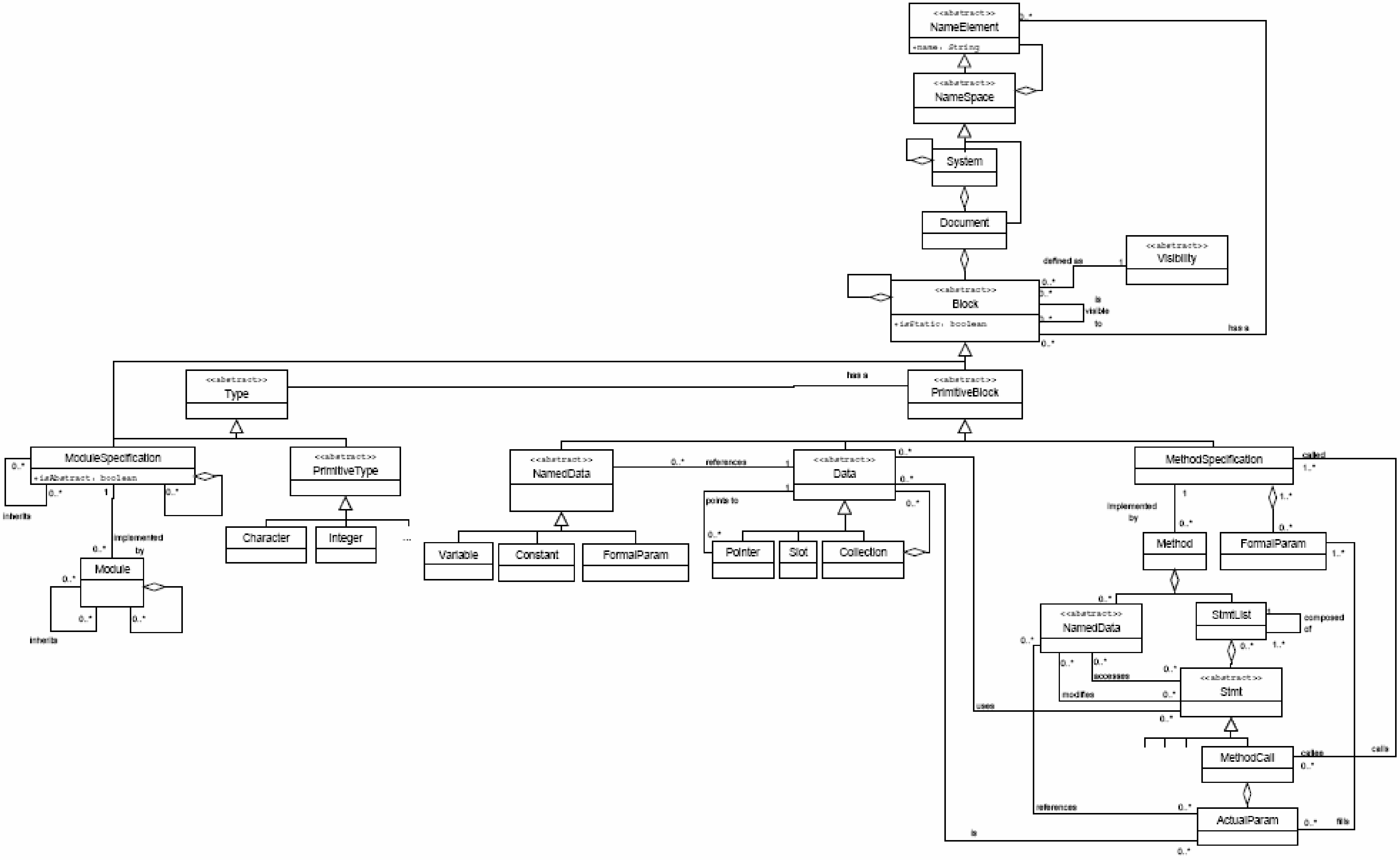
**RFP #6: ADM: Refactoring Package**
The Refactoring Package defines ways in which the KDM can be used to refactor applications. This includes structuring, rationalizing, modularizing and in other ways improving existing applications without redesigning those systems or otherwise deriving model-driven views of those systems. Work on the Refactoring Package RFP will begin after issuance of the Visualization Package RFP.

**RFP #7: ADM: Target Mapping & Transformation Package**
The Target Mapping & Transformation Package defines mappings between the KDM and target models. This standard defines the mappings and transformations that may occur between existing applications and top down, target models. Development paradigms may vary, but will include MDA as a target. This standard will complete ADM task force efforts in providing a transformational bridge between existing systems and target architectures.

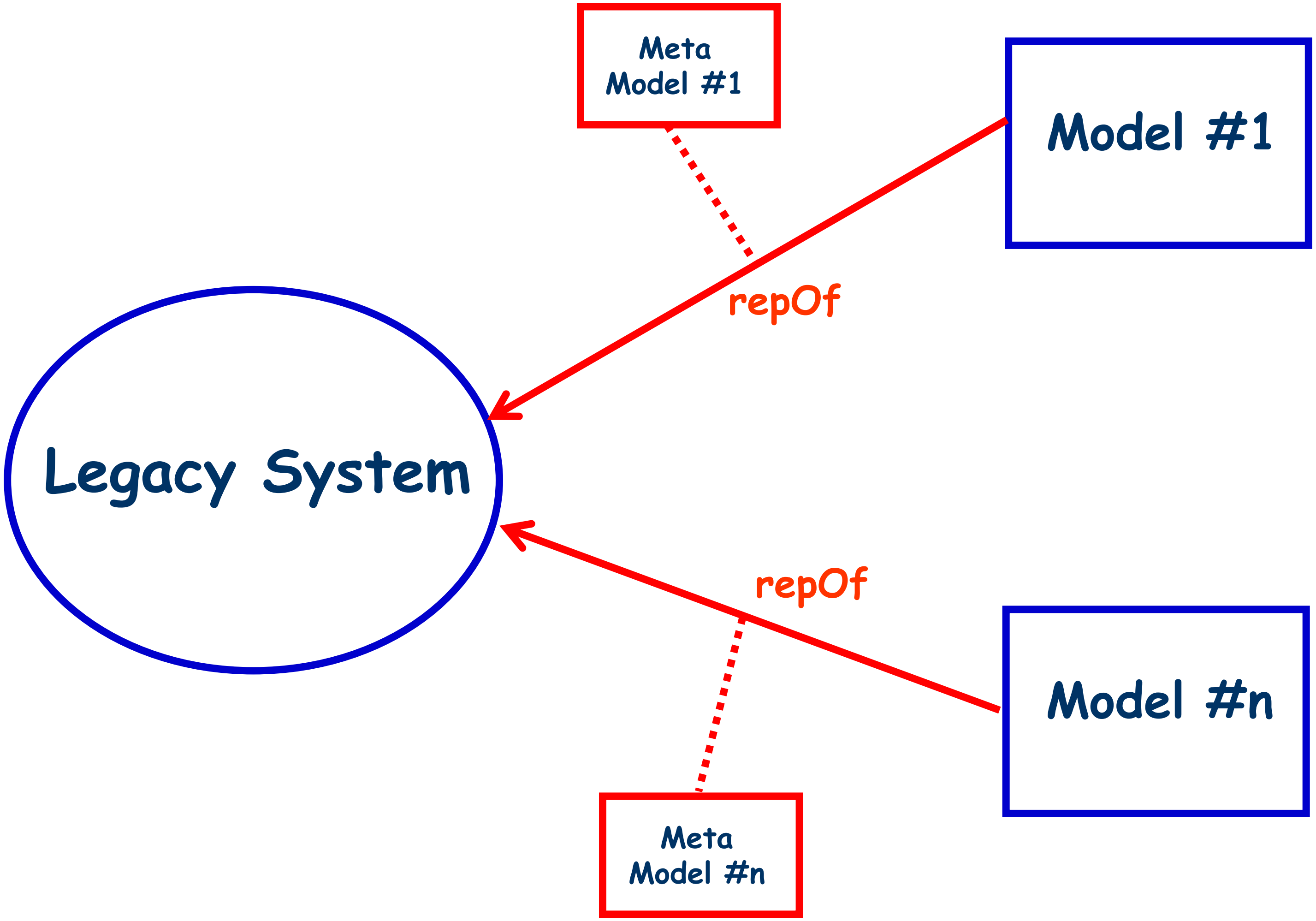
# General Static Metamodel (DSTC proposal)



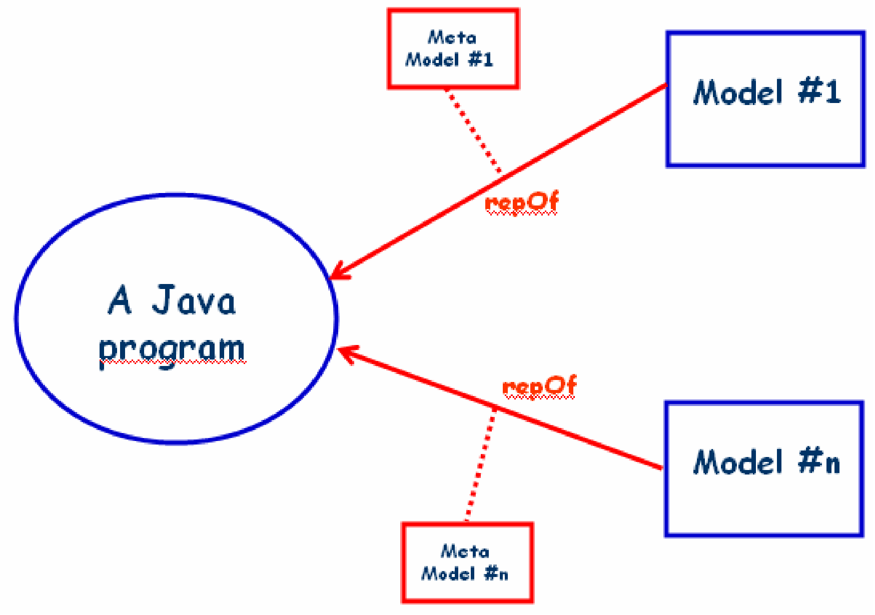
# OO Static Metamodel (DSTC proposal)



# Metamodel-Driven Aspect extraction

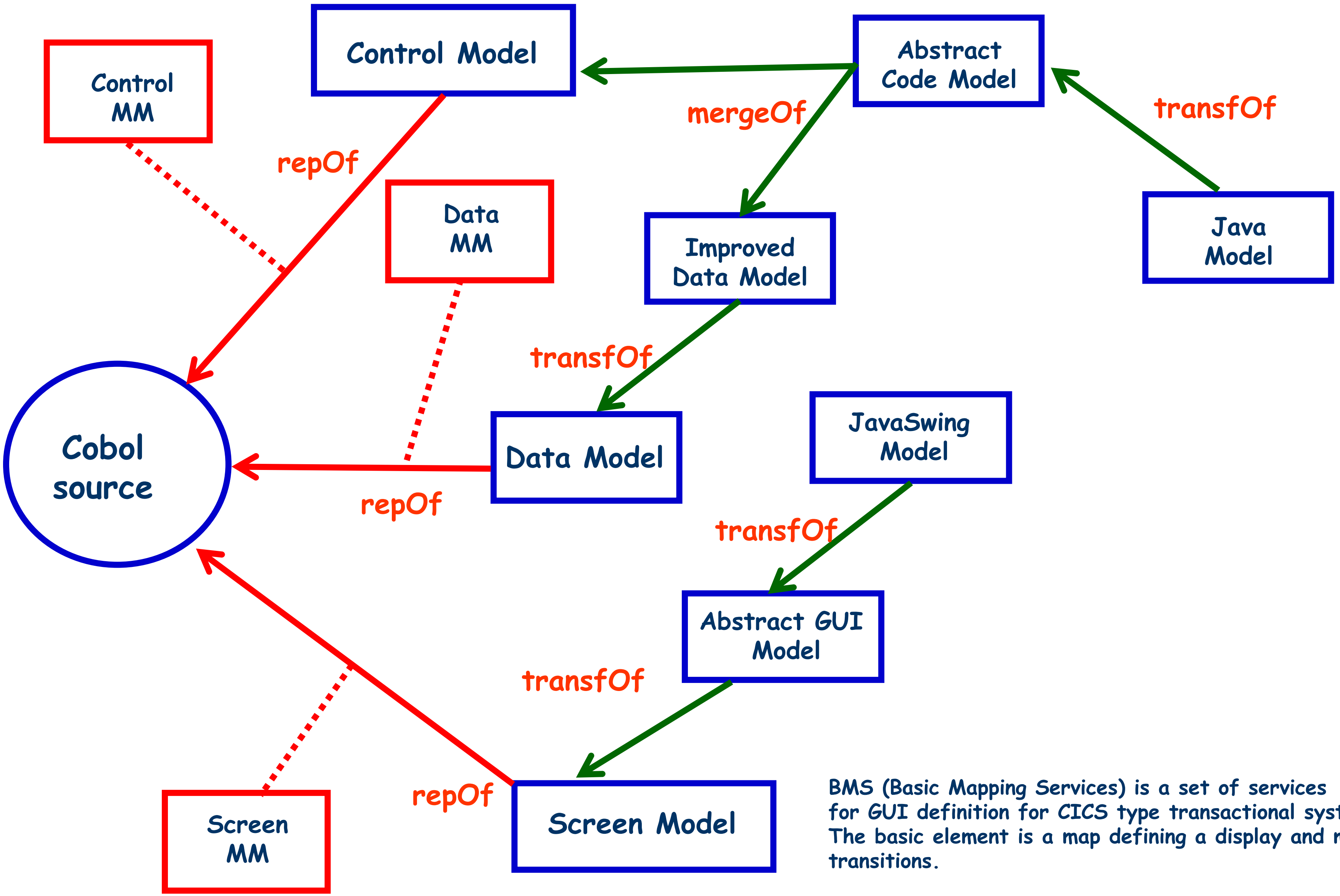


# Metamodel-Driven aspect extraction from source code



- There is no unique extraction metamodel
- Example of model extraction from a source Java or Cobol program.
  - 100% syntactic extraction if MM = Grammar (reversible)
  - <100% if MM<grammar
  - >100% if MM>Grammar
    - need heuristics
    - e.g. pattern discovery

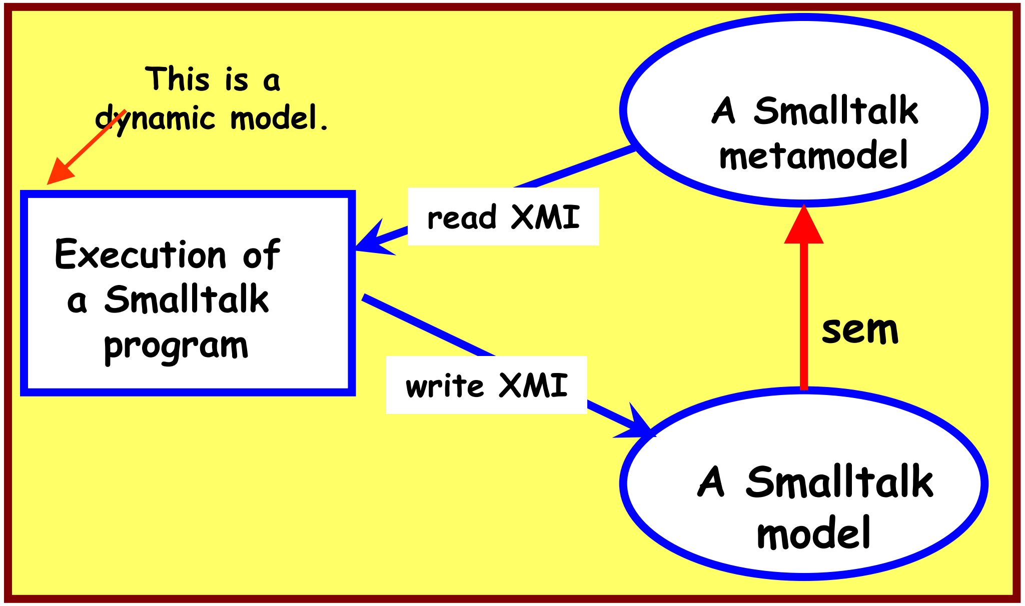
# Cobol to Java extraction: Principle transformation chain

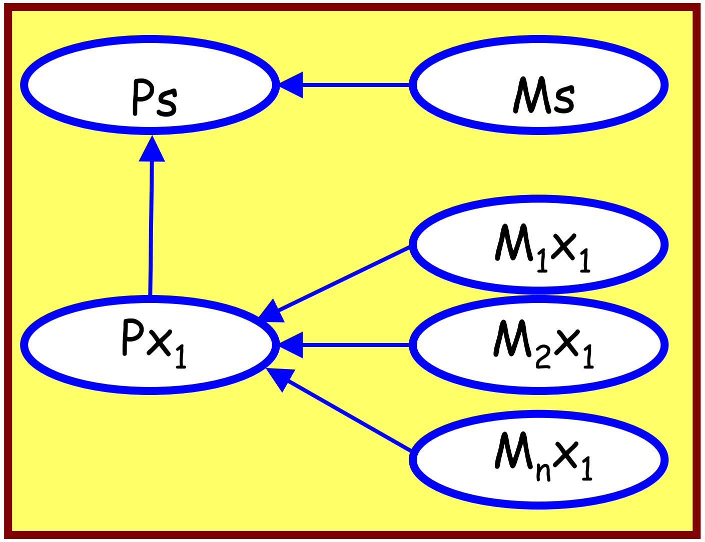


BMS (Basic Mapping Services) is a set of services for GUI definition for CICS type transactional systems. The basic element is a map defining a display and map transitions.

# Just in time model production (an execution trace is a model)

- *A program execution may produce at any time a model of itself, based on a given metamodel.*
- This model may correspond to an instantaneous state of its execution, including instances.
- It may also correspond to a history of its execution states: a trace is a model.
- Model production may be periodical, or on demand, or on specific events.
- The produced model may be either directly used or serve to produce in terms other models, possibly by model combination.
- All MDA models and metamodels conforms to the XMI format.
- From internal to external reflection.
- Variable metamodel handling is possible by decorating the metamodel with reflective API discovery code (ako model weaving operation).





Agenda

# Conclusions and perspectives

# Conclusions

- **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 metamodels are beginning to be considered alike in the 2000's.
  - Libraries of hundreds of domain specific models and metamodels of high abstraction are beginning to appear. Each such metamodel contains a number of concepts and relations.
  - Tools will be needed to work with these vast libraries of models and metamodels. These tools will be much different of present CASE tools and class browsers. They will interoperate on open platforms (EMF or VisualStudio based). Many of them will be partially automatically generated from metamodels.

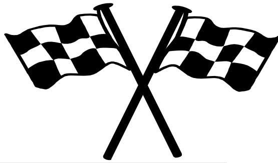
# Conclusions

- The problems of understanding what is a good metamodel for a PIM, a PSM or a PDM may take longer to settle than initially planned. The concept of PSM itself may be questionable and is probably still ill-defined.
- The model transformation operation is beginning to be understood (QVT++). Other operations like model weaving still need research work. The global relations between models should be considered.

# Conclusions

- What could kill the MDE? Only two things:
  - Lack of modesty
    - Overselling
    - Hiding difficult open problems
    - Claiming that everything is simple and under control
    - etc.
  - Lack of ambition
    - Ad-hoc solutions
    - No research or insufficient research.
    - "UML case tool vendor" restricted
    - If the MDE does not fly, then other technology spaces will harbor similar ideas

# Main messages of the presentation



- Model Driven Engineering (i.e. the consideration of models as first class entities) and its various variants are changing the landscape of software engineering and data engineering.
- Software modernization is more demanding than plain forward engineering, but could benefit as well from the application of model engineering. Many well established software modernization practices may be revisited in the unifying context of MDE.
- However the more urgent need is to define the foundations for model engineering. MDE is presently industry-driven and still lacks solid conceptual basis.
- The relations of conformance and representation lies at the hearth of these conceptual foundations. Model engineering is presently in the state of object technology in the 80's.

## Thanks

- Questions?
- Comments?

**http://www.sciences.univ-nantes.fr/lina/atl/**

**Jean Bézivin**

Jean.Bezivin{noSpamAt}lina.univ-nantes.fr

Equipe ATLAS, INRIA & LINA, Nantes, France