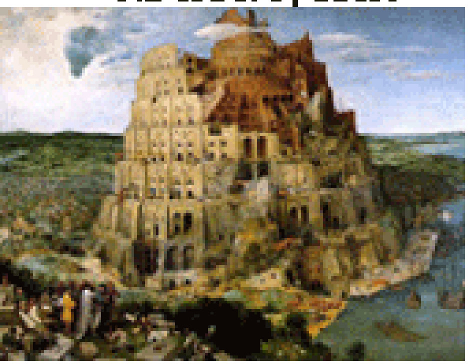
If Model Driven Engineering is the Solution, then What is the Problem?

(or on the broadening scope of software modeling techniques)

The tower of Babel



Bruegel the Elder 1563

Jean Bézivin

mail Jean.Bezivin@inria.fr
twitter@JBezivin

AtlanMod Team (INRIA & EMN),
Nantes, France

http://www.emn.fr/x-info/atlanmod/

SLE 2nd International Conference on Software Language Engineering Denver, Colorado, October 5-6, 2009

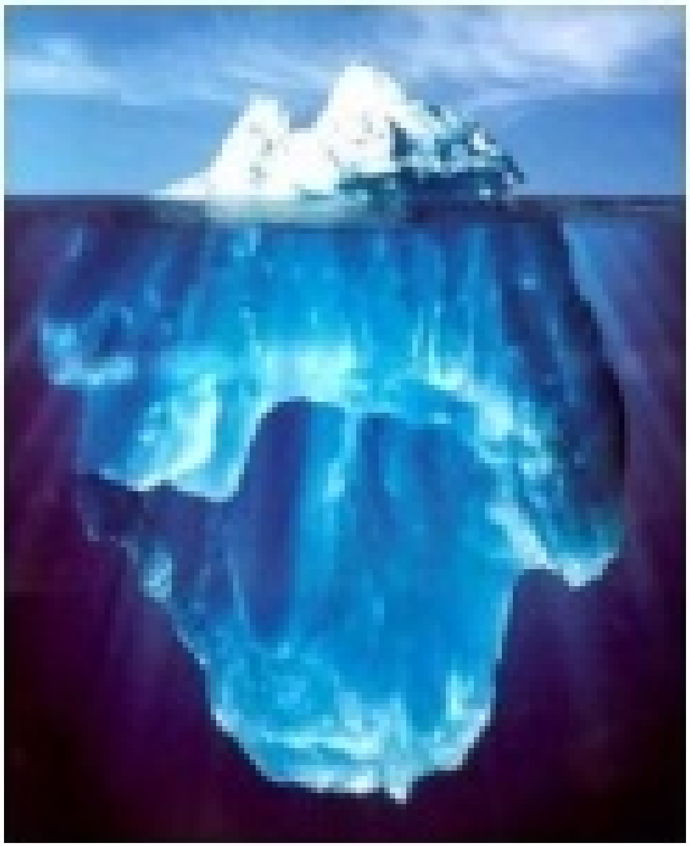
INRIA ECOLE DES MINES DE NANTES

INTRODUCTION

Some observations on a rapidly evolving situation

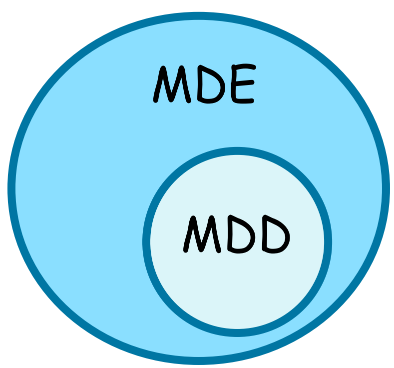
Main Message of the presentation

We have not yet seen the full application deployment of MDE

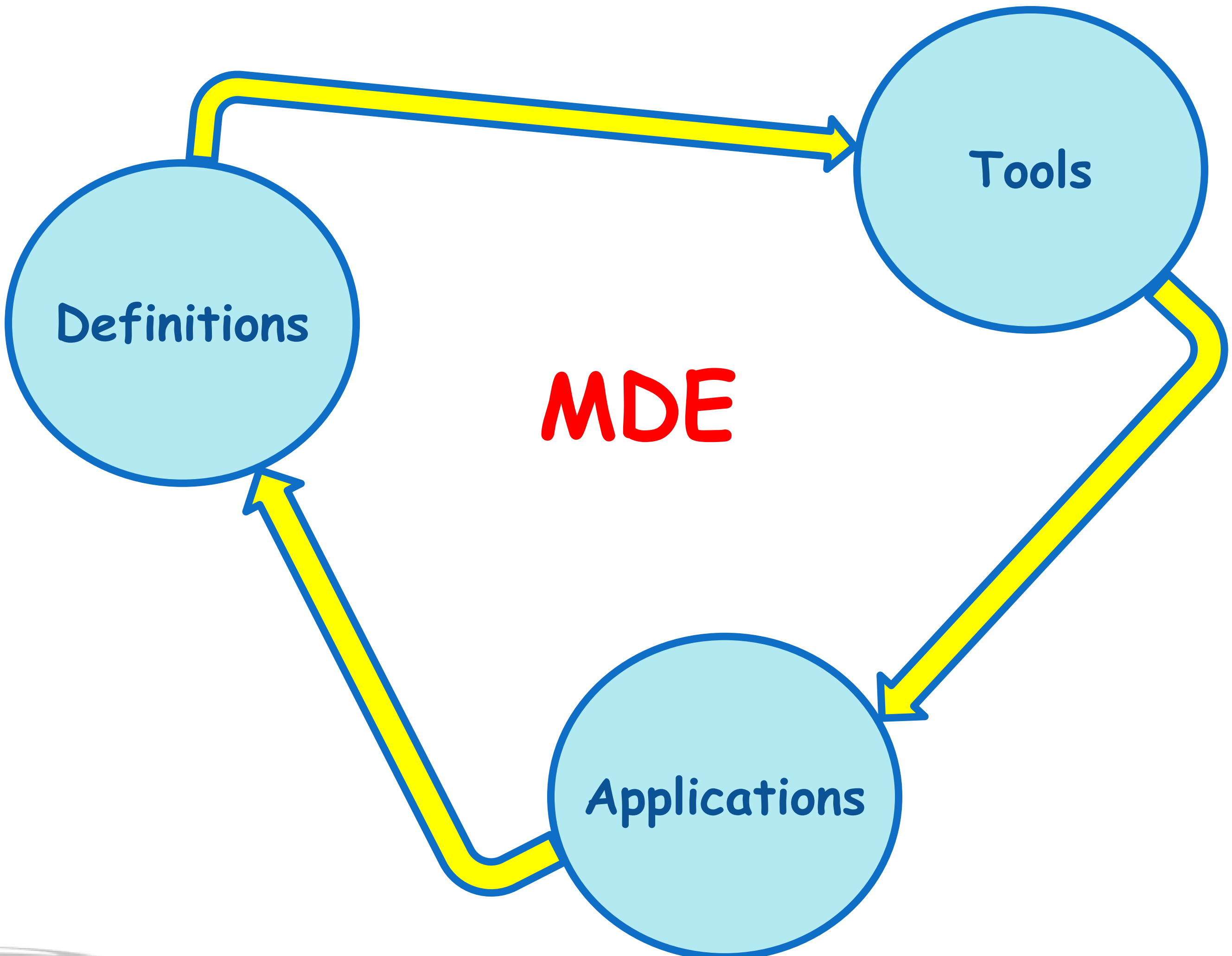


MDD

MDE



Progressive Improvements



Questions about models

- New MDE applications challenge the tools and the basic definitions, push to their improvement and contribute in turn to broadening the applicability scope
- Three main recurring questions
  - ✓ What is a model?
  - ✓ Where are models coming from?
  - ✓ What may models be useful for?
- Answers to these three questions often depend on the application type
- MDE is progressively finding its right place in cooperation with other technical spaces

CONTEXT

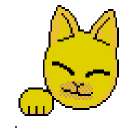
Experimental observations coming from several projects

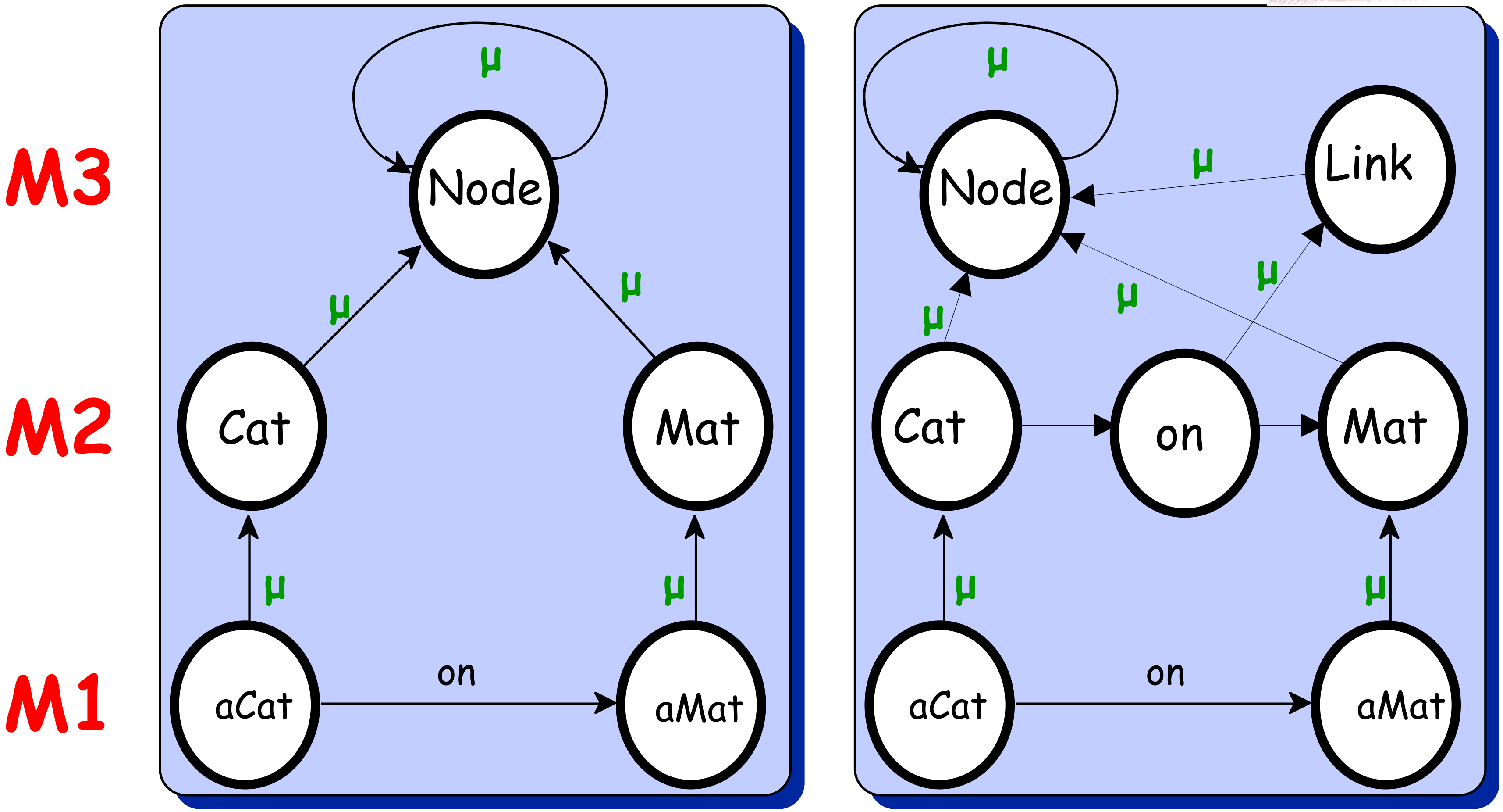
Context

- Modeling projects
  - ✓ sNets (1990-2000) w. Smalltalk
  - ✓ AmmA (2000-now) w. Eclipse
- Normative Organizations (1995-2005)
  - ✓ OMG
  - ✓ UML, OCL, MDA, QVT, ADM, KDM, etc.
- Open source meets Modeling
  - ✓ Demand-Driven Evolution
  - ✓ Best observation point for technology evolution

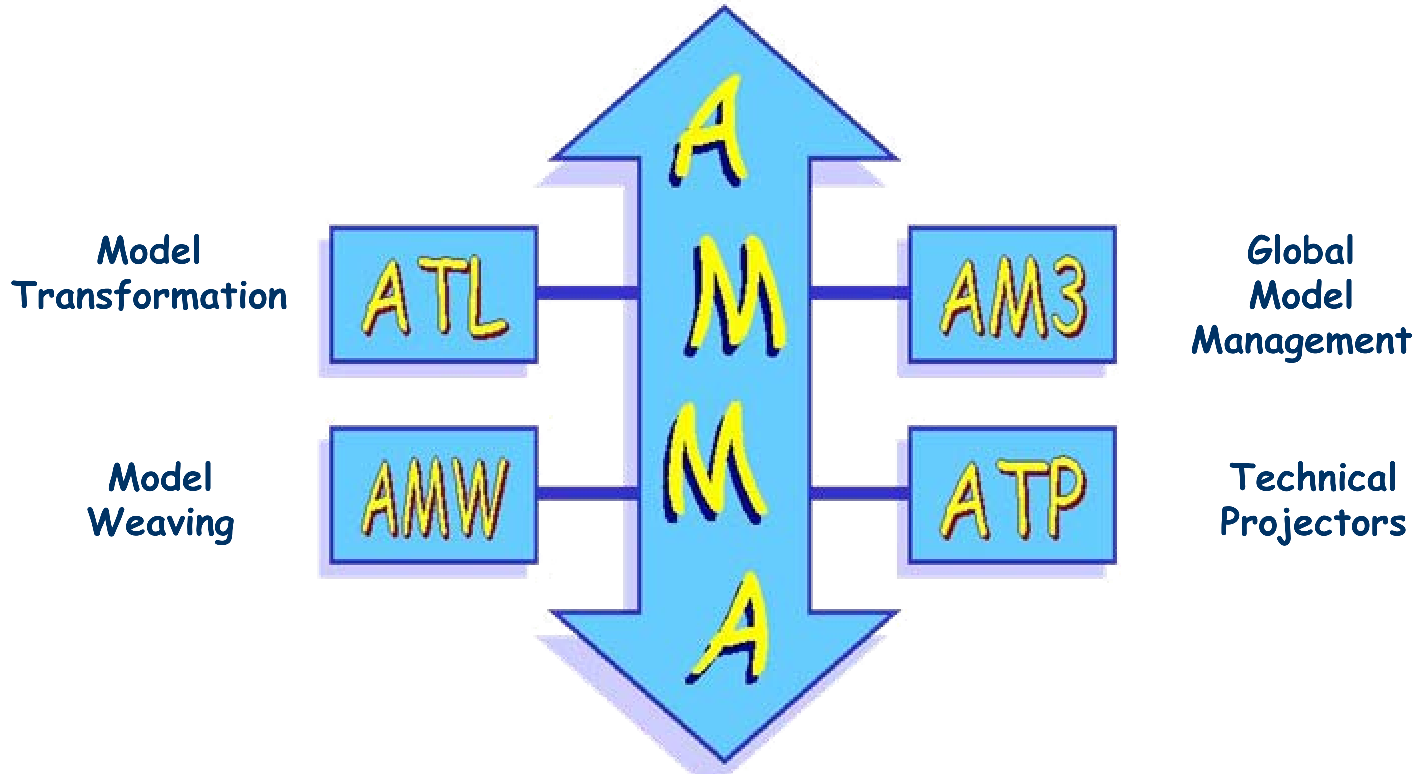
sNets :Constrained Semantic Networks

$\exists\, x, \exists\, y : Cat\ (x) \wedge Mat\ (y) \wedge on\ (x,y)$





AtlanMod model management Architecture

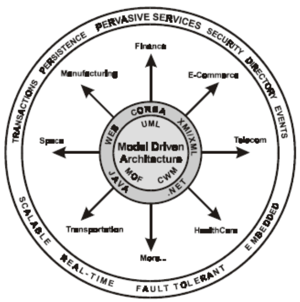


AmmA as a "DSL Framework", i.e. a framework built from a set of DSLs (KM3, ATL, AMW, AM3, TCS, XCS, BCS, etc.) and intended to build new DSLs (CPL, SPL, etc.)

A POINT OF HISTORY

The sources of modern software modeling

Starting point : OMG definition of MDA™



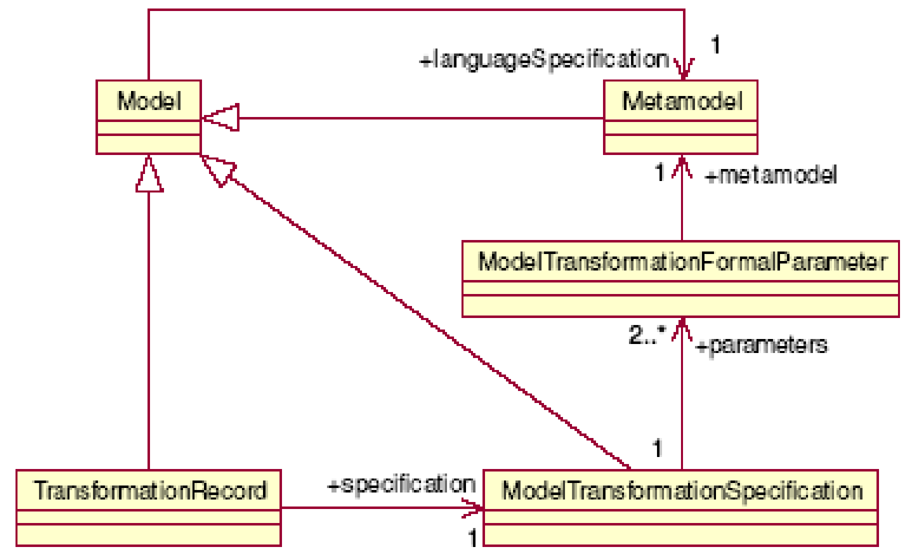
Richard Soley and the OMG staff,
MDA Whitepaper Draft 3.2
November 27, 2000

A Proposal for an MDA Foundation Model

An ORMSC White Paper

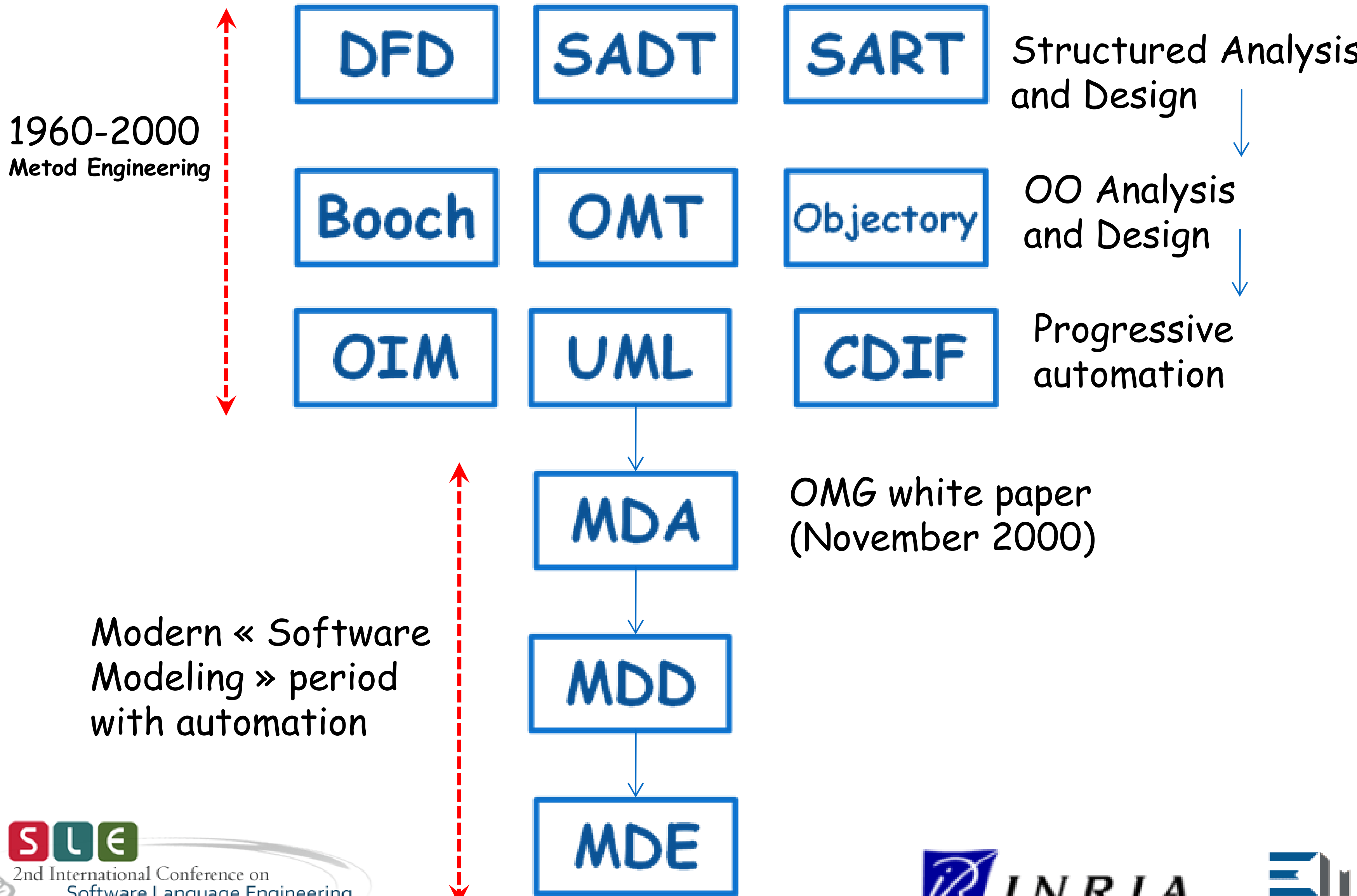
V00-02

ormsc/05-04-11

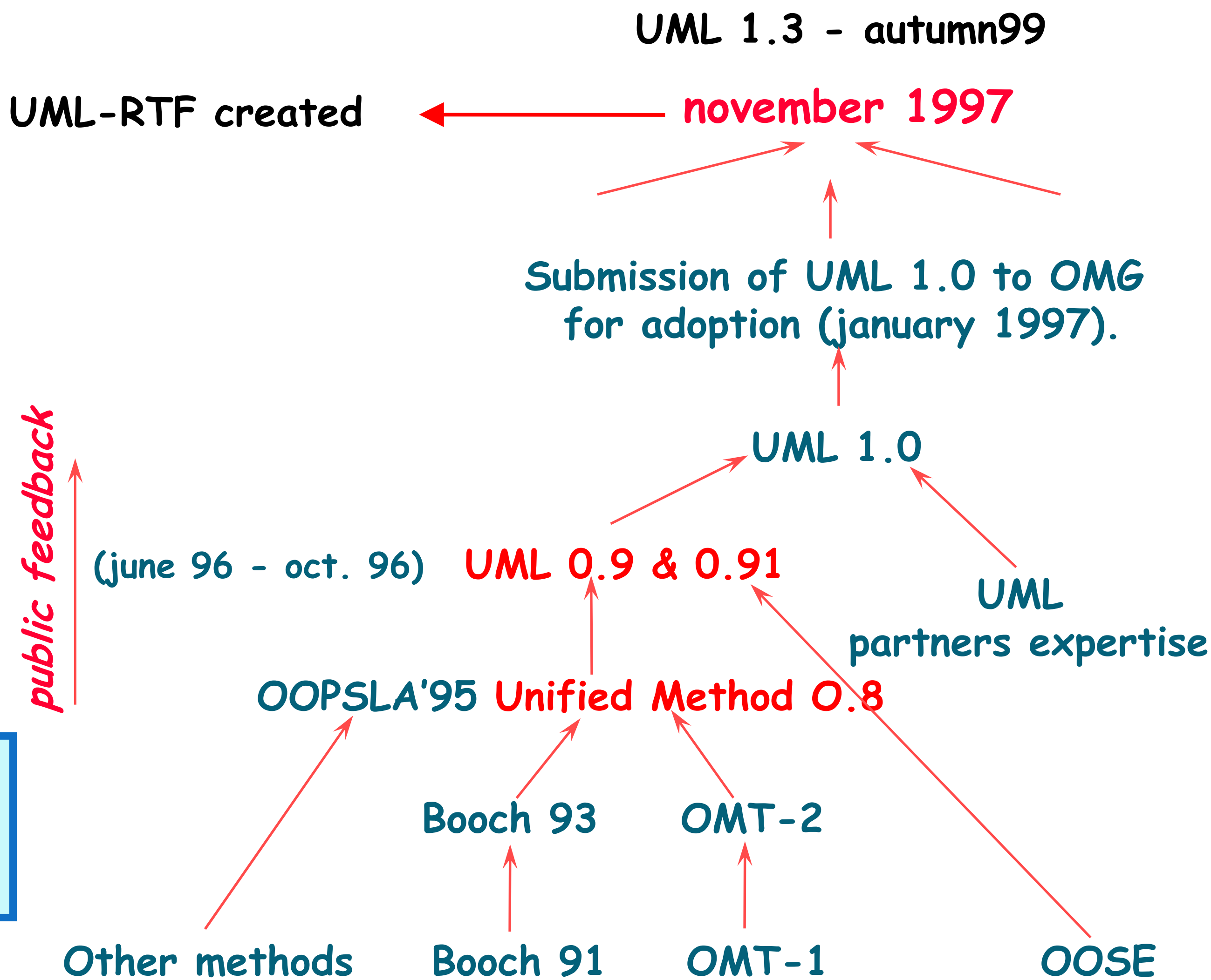


… At the core of MDA are the concepts of models, of metamodels defining the abstract languages in which the models are captured, and of transformations that take one or more models and produce one or more other models from them. …

Method Engineering and Model Engineering

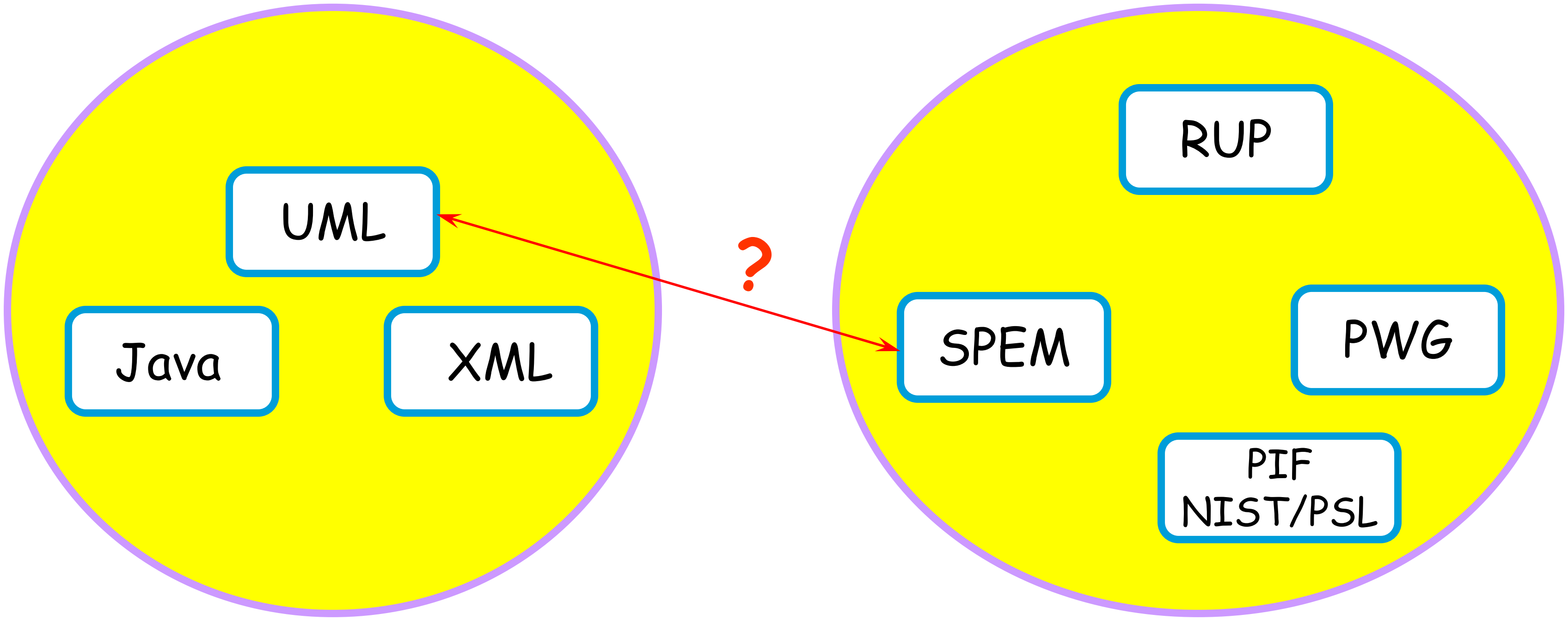


UML contribution : separation of concerns



From Unified Method
To Unified Language

But how to weave the product and process models?



Language separation achieved
Language coordination yet to be worked out

18 ± 3 Software Technology Maturation

- « *The magic number Eighteen Plus or Minus Three* », William E. Riddle, ACM Sigsoft, April 1984
- 15 to 20 years to mature a technology to the point that it can be popularized and disseminated to the technical community at large

Cost Models
0-1966- appearance of first collection of cost-related data
1-1976- appearance of a usable system (Price S)
2-1978- alternative systems are available (for example, COCOMO)
3-1981- publication of Boehm's text

Smalltalk-80
0-1965- Kay's thesis defines concept of a personal computerized notebook
1-1972- preliminary version of Smalltalk is available
2-1976- major new version of Smalltalk appears
3-1981- other companies start porting the Smalltalk-80 system to their computers
4-1983- Smalltalk-80 available as a commercial product

SREM
0-1968- ISDOS system demonstrates applicability of attribute-value-relation approach to pre-implementation activities
1-1973-74- first concrete definition of the SREM system appears
2-1977- first release of the SREM system
3-1981- Vax version available

Unix
0-1967- appearance of the Multics system
1-1971- initial versions of Unix available
2-1973- Unix system debuts at Sigops conference
3-1976- collection of papers appears and system begins to be widely used in academic community
4-1981- announcement of Unix System III

0 1 2 3 4
4 V
3 V Substantial Evidence of Value and Applicability
2 V Shift to Usage Outside of Development Group
1 V Usable Capabilities Available
0 V Definition Via Seminal Paper or Demonstration System
Emergence of Key Idea

Knowledge-based Systems
0-1965- appearance of artificial intelligence systems providing intelligent assistance (for example, Dendral)
1-1973- appearance of systems containing a knowledge base (for example, Hearsay)
2-1970-00- appearance of knowledge-based systems that can be routinely used for problem-solving tasks (for example, R1)

Software Engineering
0-1960- inadequacy of existing techniques for large-scale software development noted in several projects (for example SAGE)
1-1968- concept of software engineering is articulated at Workshop on Software Engineering at Garmisch Partenkirchen
2-1973-74- general collections of papers appear and policy guidelines are established in various communities
3-1978-79- texts and generally usable systems supporting software engineering appear (for example, the SREM system)
4-1983- use of software engineering shifts to a larger community through actions such as the DeLauer directive and the definition of a Software Engineering Institute

Verification
0-1966- Floyd's paper on program correctness analysis
1-1971- King's demonstration system appears
2-1975- multiple systems are available
3-1979- usage of some systems shifts to application groups

Compiler Construction
0-1961- Iron's paper on compiler generation
1-1967- review paper by Feldman and Gries
2-1970- usable systems appear (such as the XPL system at Stanford)
3- - (cannot be determined)
4-1980- appearance of production-quality compiler-compilers

Metrics
0-1972- publication of book on Halstead metrics
1-1977- results of trying to measure various empirical and analytic measures appear

Abstract Data Types
0-1968- initial report on information hiding
1-1973- appearance of some languages using idea of abstract data types (for example, TOPD design language)
2-1977- major publication on the subject and frequent appearance of the concept in new programming languages (for example, CLU)
3-1980- use of abstract data types in other technologies (such as in the Affirm program verification system)

Structured Programming
0-1965- Dijkstra's paper on programming as a human activity
1-1969- paper on structured programming by Dijkstra at the First NATO-sponsored Workshop on Software Engineering
2-1972-73- concept is widely discussed and presented in papers
3- - (cannot be determined)
4-1976- publication of first introductory text based on structured programming

SCR Methodology
0-1968- appearance of concepts such as information hiding and communicating sequential processes
1-1976- completion of feasibility demonstration by NRL with positive experiences
2-1978-79- appearance of training material and models of usage
3-1982- methodology moved to a variety of other organizations

DOD-STD-SDS
0-1967- initial articulation of phased approaches to software development
1-1980- contract signed for development of DOD-STD-SDS

AFR 800-14
0-1972- basic need for policy and specific guidance is documented
1-1973- initial draft policy is published
2-1974- policy guidance is published
3-1974- final draft is available
4-1975- regulation and instructions for its use are officially published

Software Technology Maturation Process (W. Riddle)

BASIC RESEARCH
investigation of ideas and concepts that later prove fundamental
general recognition of problem and discussion of its scope/nature

0

Appearance of a Key Idea Underlying the Technology or a Clear Articulation of the Problem

CONCEPT FORMULATION
informal circulation of ideas
convergence on a compatible set of ideas
general publication of solutions to parts of the problem

1

Clear Definition of Solution Approach Via a Seminal Paper or a Demonstration System

DEVELOPMENT and EXTENSION
trial, preliminary use of the technology
clarification of the underlying ideas
extension of the general approach to a broader solution

2

Usable Capabilities Become Available

ENHANCEMENT and EXPLORATION (Internal)
major extension of general approach to other problem domains
use of the technology to solve real problems
stabilization and porting of the technology
development of training materials
derivations of results indicating value

3

Shift to Usage Outside of Development Group

ENHANCEMENT and EXPLORATION (External)
Same activities as for ENHANCEMENT and EXPLORATION (Internal) but they are carried out by a broader group, including people outside the development group.

MDE?

4

Substantial Evidence of Value and Applicability

POPULARIZATION
appearance of production-quality, supported versions
commercialization and marketing of the technology
propagation of the technology throughout community of users

4a

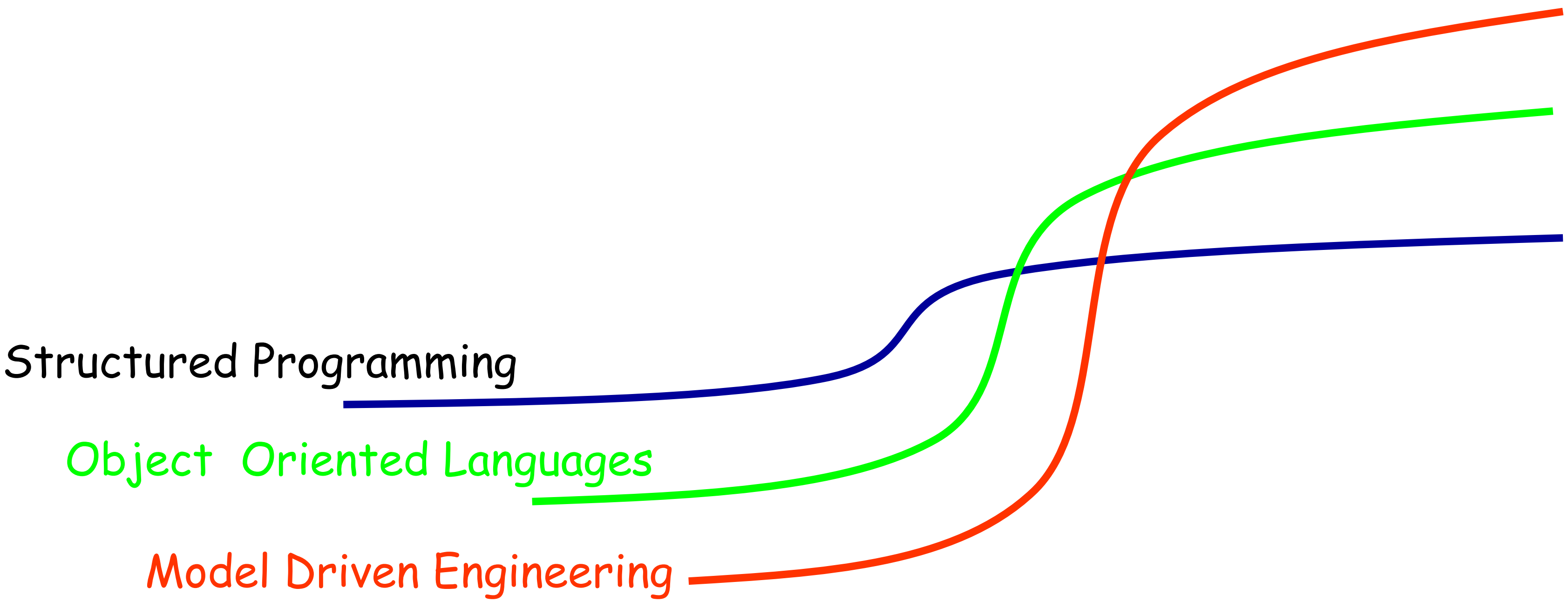
Propagation Throughout 40% of the Community

4b

Propagation Throughout 70% of the Community

Figure 1: Software Technology Maturation Phases

Technology waves

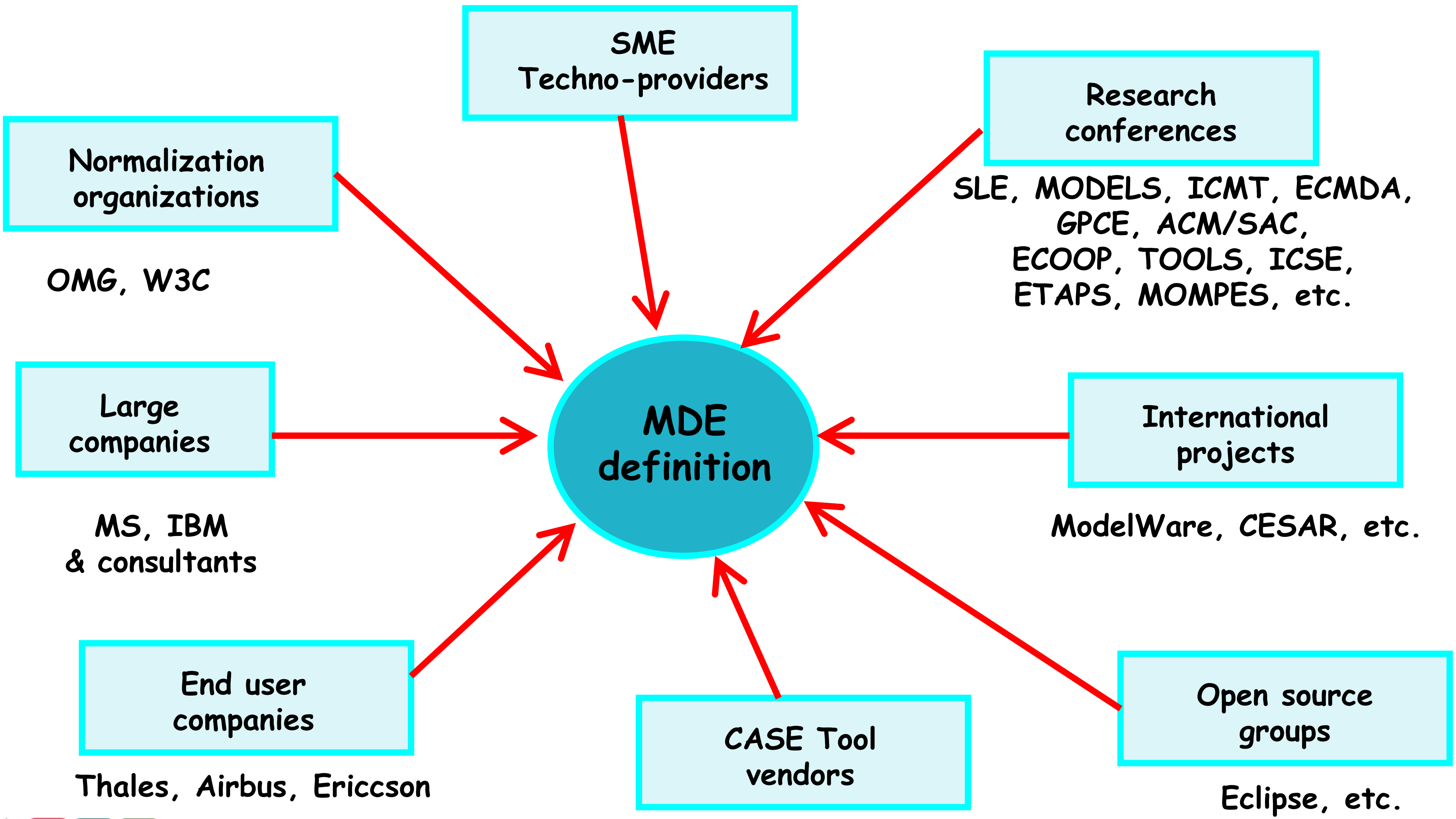


Each wave does not replace the previous one, but complements it.

MDE IN PERSPECTIVE

How is MDE evolving?

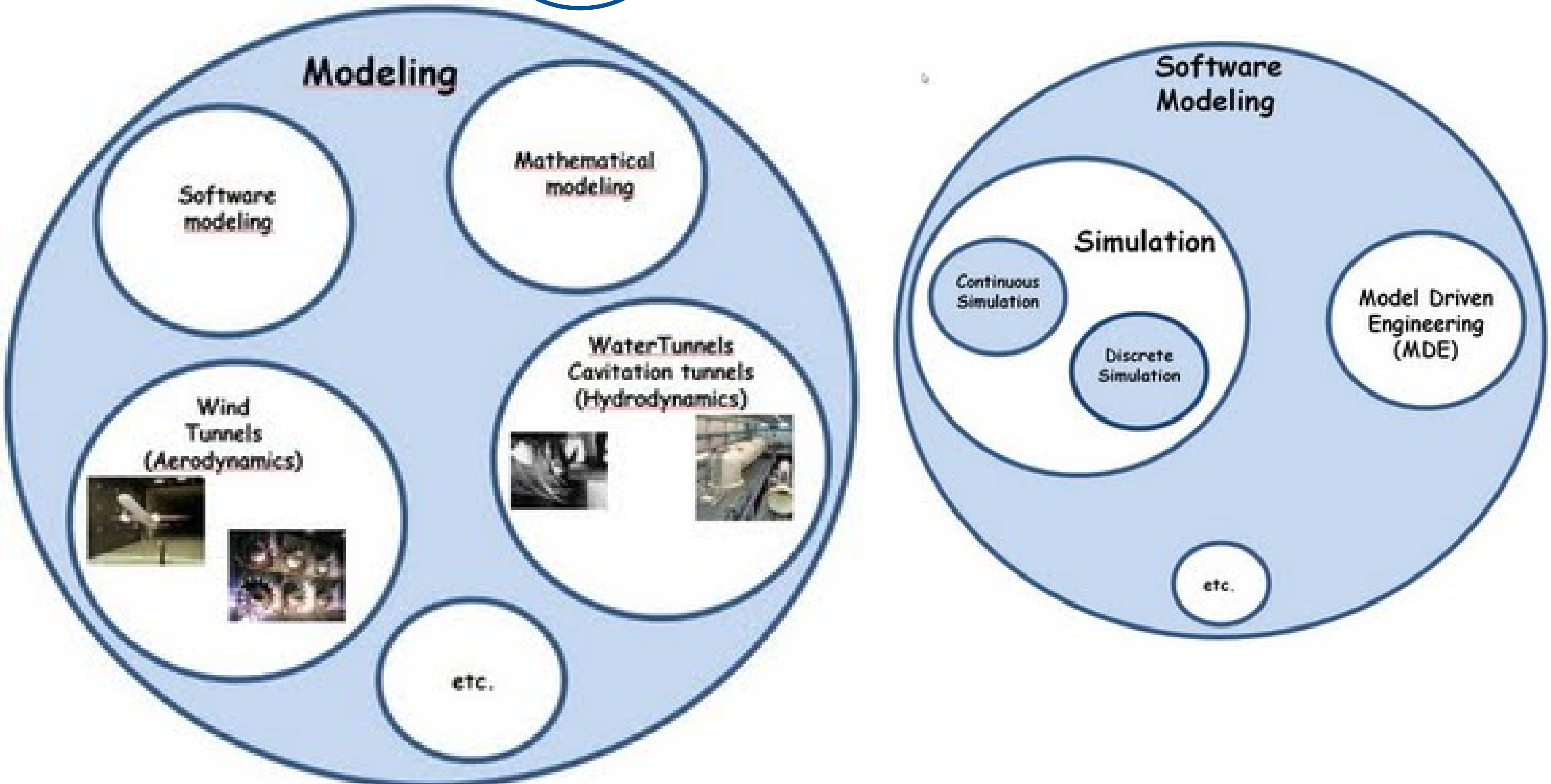
Influencing parties (some)



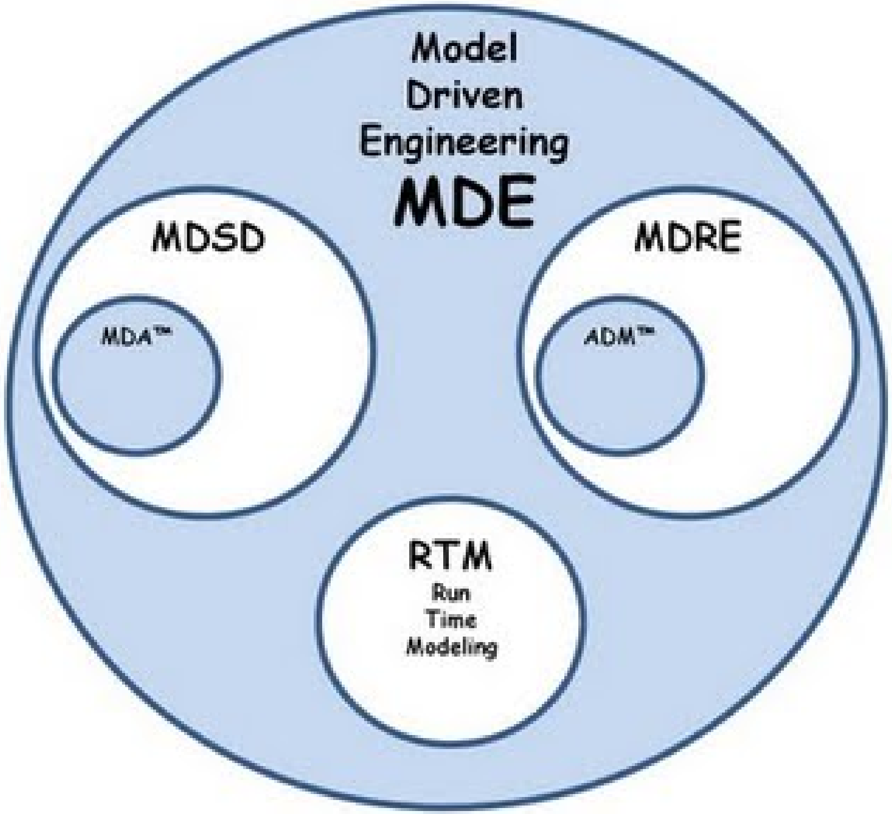
We need clear definitions

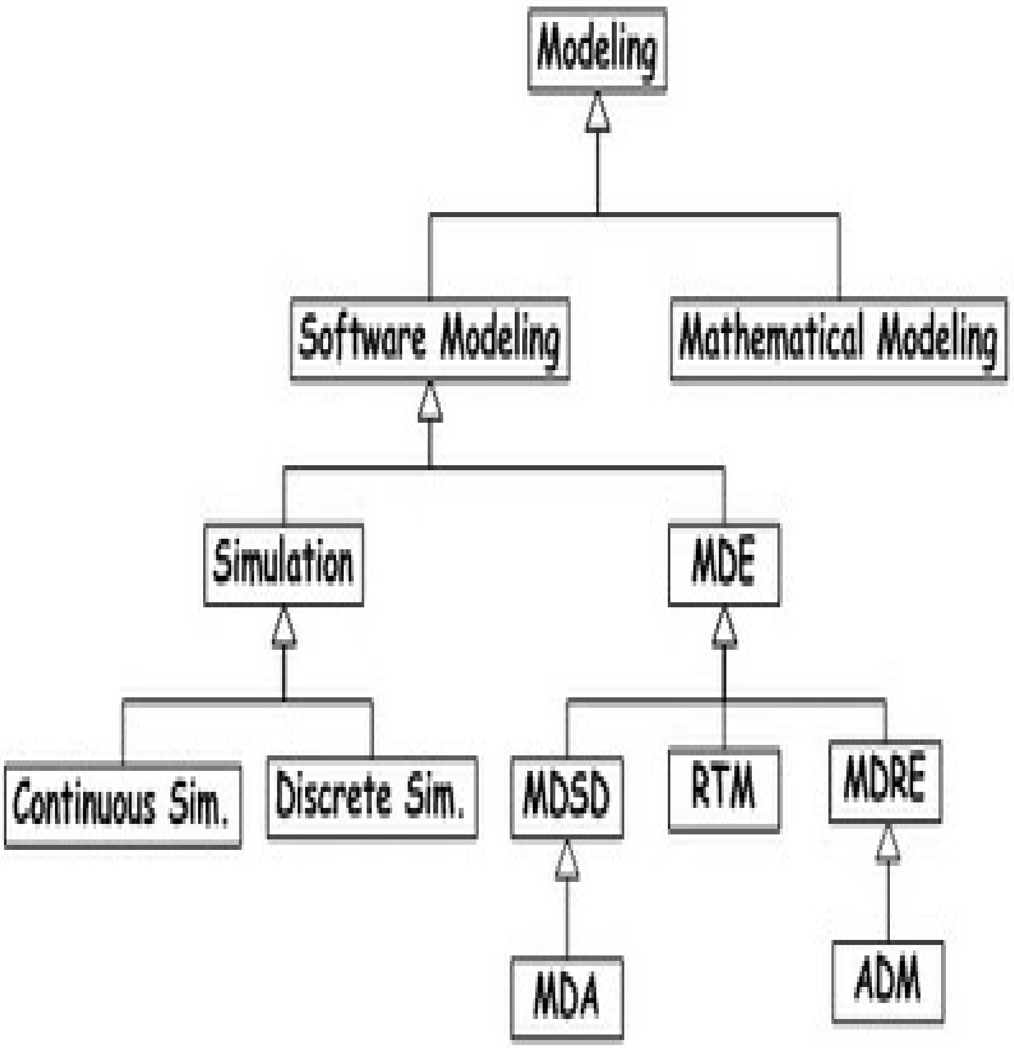
- **MDE** Model Driven Engineering
- **ME** Model Engineering
- **MDA™** Model Driven Architecture
- **MDD** Model Driven Development
- **MDSD** Model Driven Software Development
- **MDSE** Model Driven Software Engineering
- **MBD** Model Based Development
- **MM** Model Management
- **ADM** Architecture Driven Modernization
- **DSL** Domain Specific Language
- **DSM** Domain Specific Modeling
- **DDD** Domain Driven Design
- **MDRE** Model Driven Reverse Engineering
- **MD*** (Markus Voelter)
- etc.

Modeling at large



Modeling at large



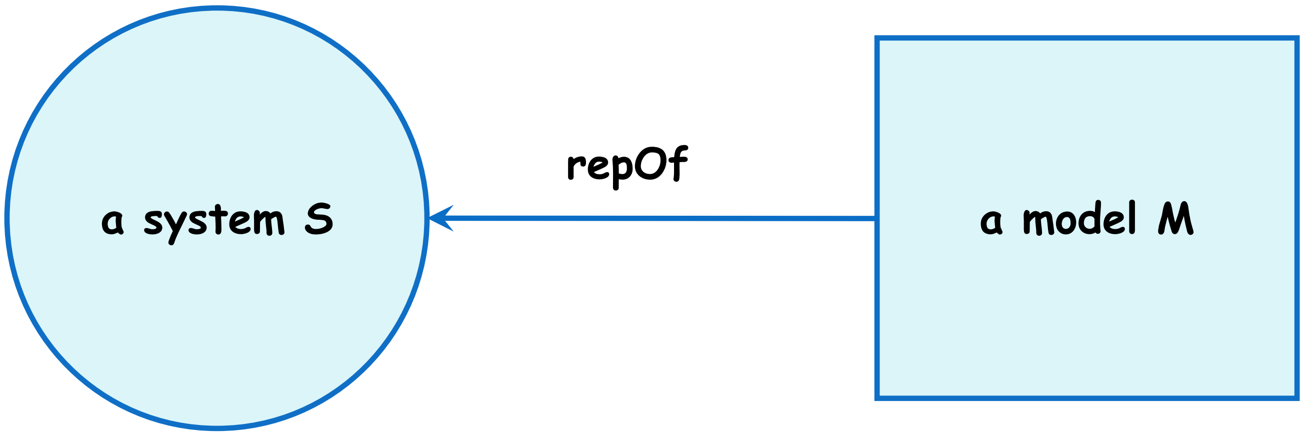


BASIC CORE MECHANISMS

We need clean definitions of basic concepts

Systems and Models

Squares and Circles



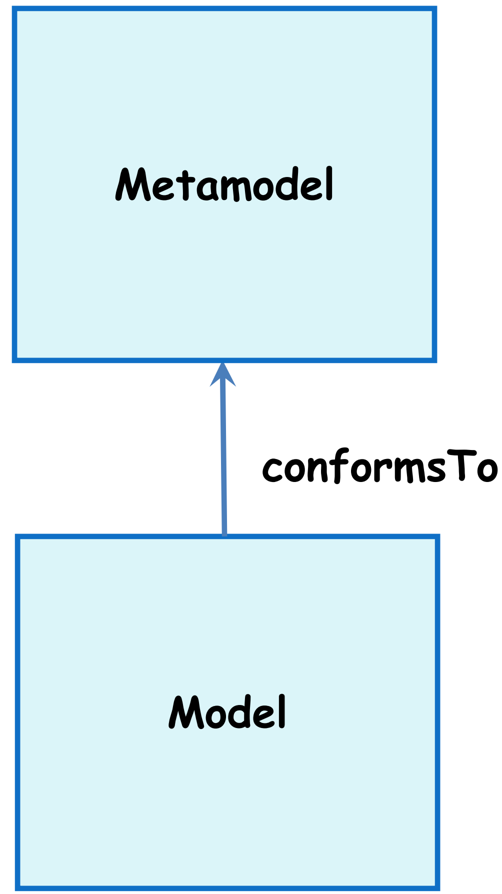
A situation or a phenomenon of
the real or imagined world.

System and Model

<u>Caution</u>: These are only plastic food models, don't eat them.

Metamodeling

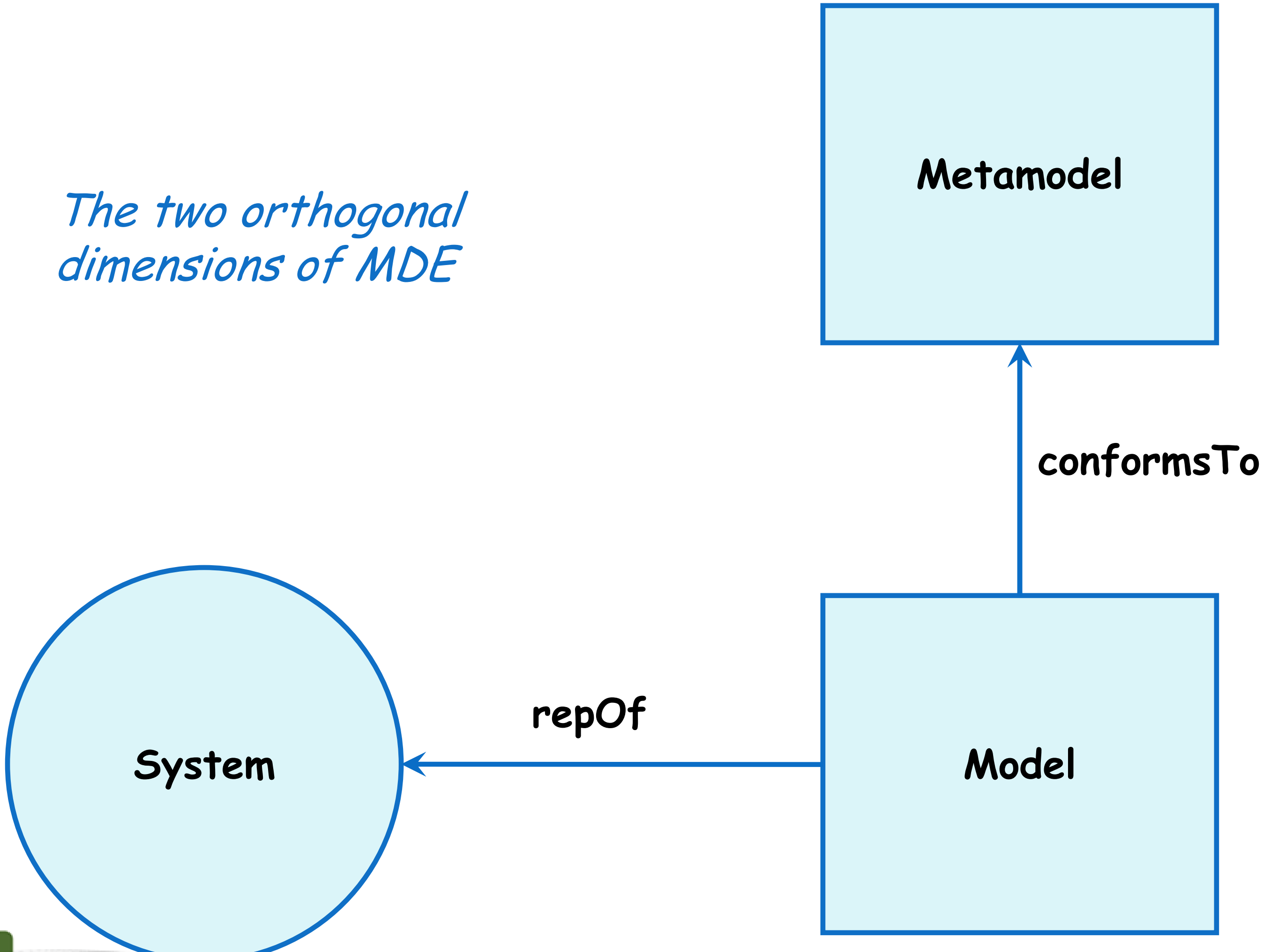
A metamodel is a simplified ontology, i.e. a set of concepts and relations between these concepts.



A model is a graph composed of elements (nodes and edges). Each such element corresponds to a concept in the metamodel.

Representation and Conformance

The two orthogonal dimensions of MDE



Structural definition of a model

- **Definition 1**. A directed multigraph $G = (N_G, E_G, \Gamma_G)$ consists of a set of distinct nodes N_G, a set of edges E_G and a mapping function $\Gamma_{G:}$: $E_G \rightarrow N_G \times N_G$

- **Definition 2**. A model $M = (G, \omega, \mu)$ is a triple where:
  - ✓ $G = (N_G, E_G, \Gamma_G)$ is a directed multigraph
  - ✓ ω is itself a model, called the reference model of M, associated to a graph $G_\omega = (N_\omega, E_\omega, \Gamma_\omega)$
  - ✓ μ: $N_G \cup E_G \rightarrow N_\omega$ is a function associating elements (nodes and edges) of G to nodes of G_ω (metaElements)

Definitions

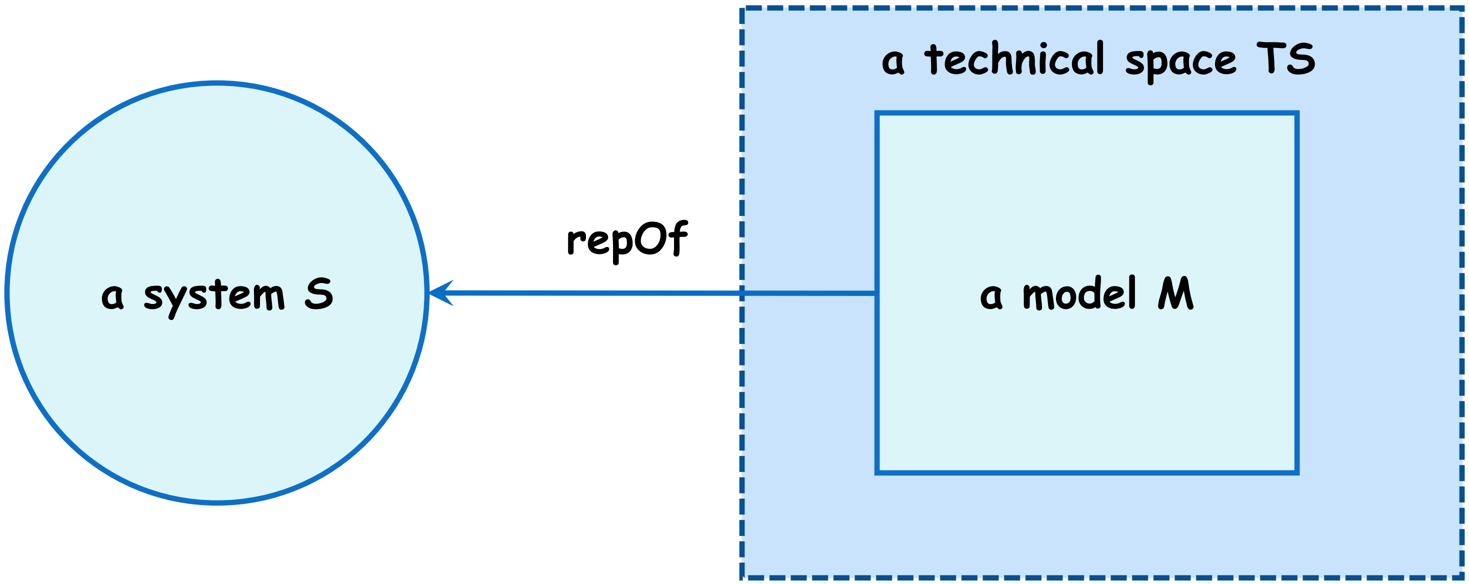
- **Definition 3**. A metametamodel is a model that is its own reference model (i.e. it conforms to itself).

- **Definition 4**. A metamodel is a model such that its reference model is a metametamodel.

- **Definition 5**. A terminal model is a model such that its reference model is a metamodel.

Technical Spaces

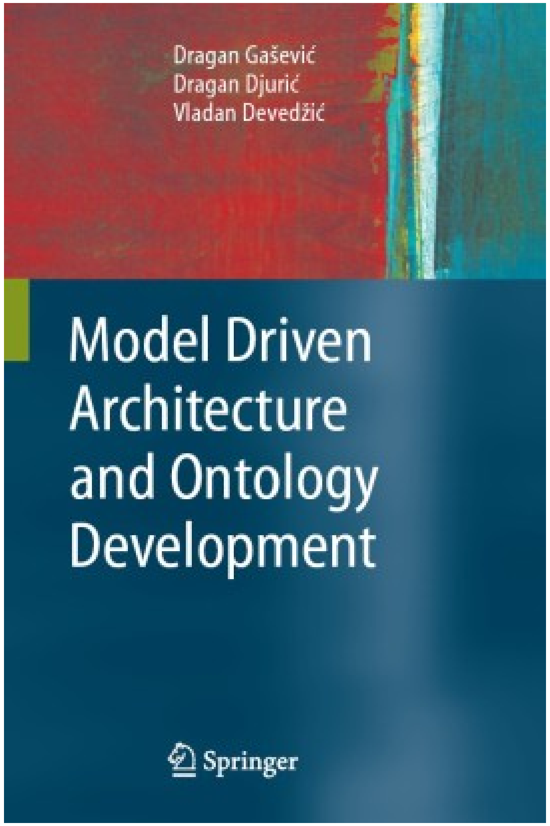
✓ Each model is expressed in some representation system, named a "technical space"
✓ Some technical spaces are based on trees, other on graphs, others on hypergraphs, etc. There are a lot of possible such representation systems.
✓MDE is sometimes called ModelWare and this is the default technical space considered here.

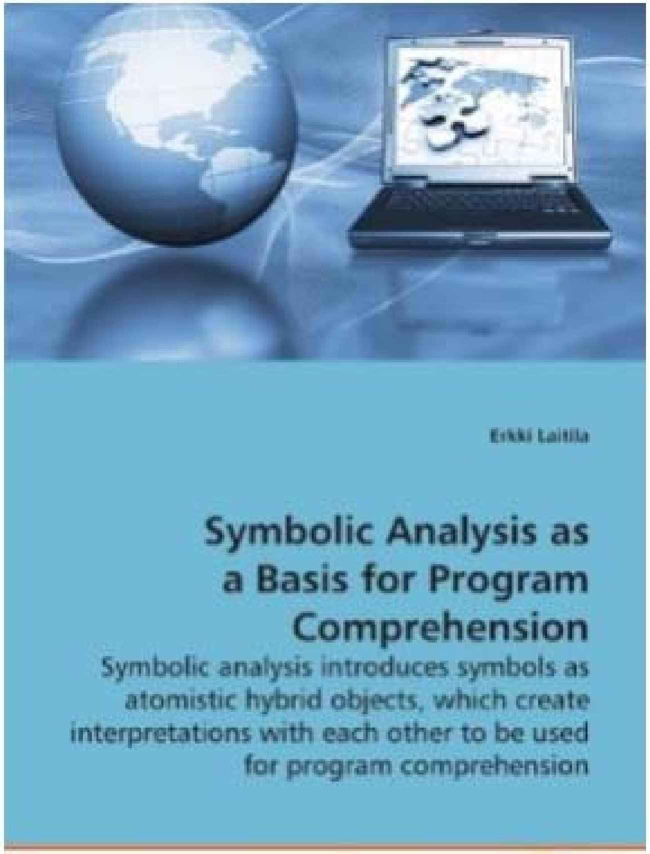


Grammarware comprises grammars and all grammar-dependent software, i.e. software artifacts that directly involve grammar knowledge.

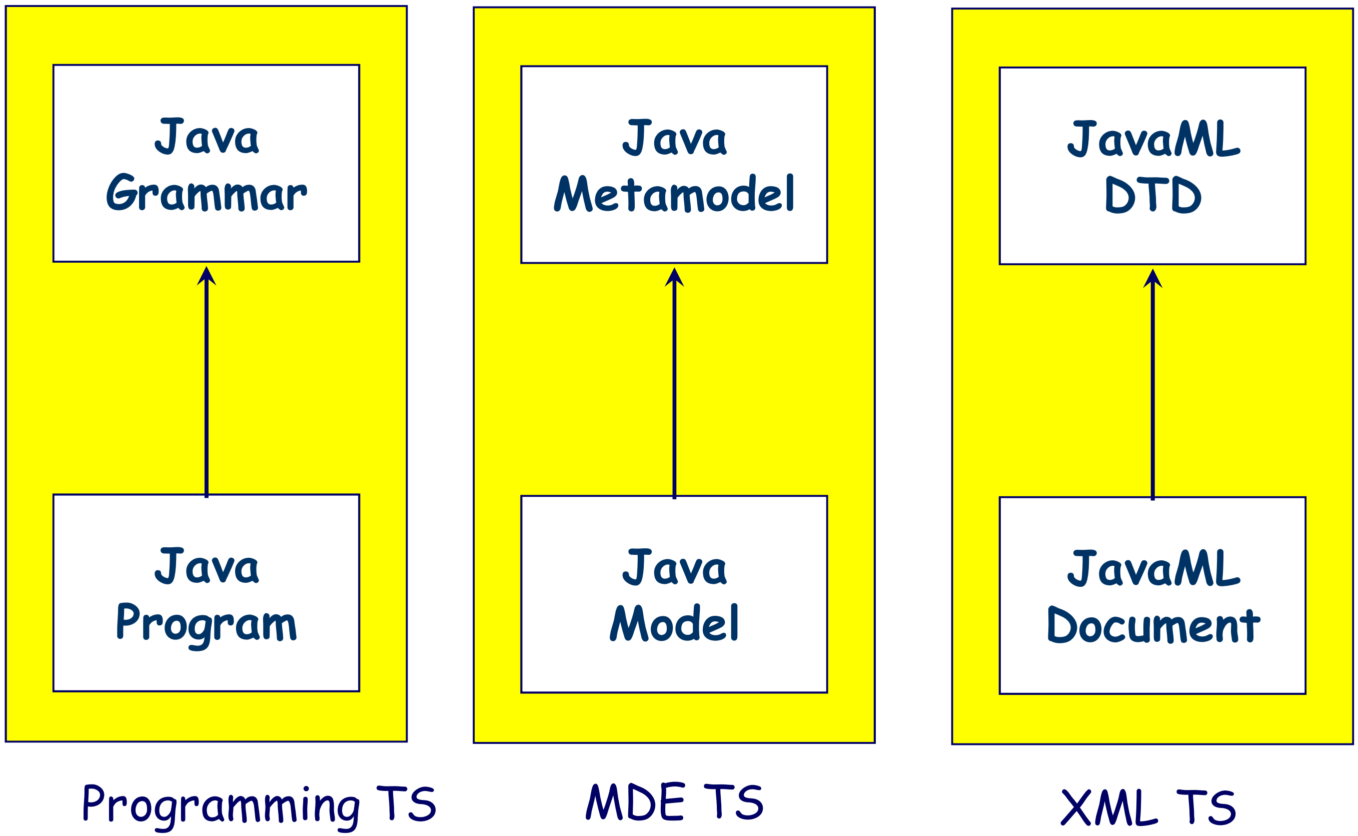
Paul Clint, Ralf Lammel, Chris Verhoef
"*Towards an engineering discipline for Grammarware*"

Several Books on Technical Spaces





Three representations for the same program



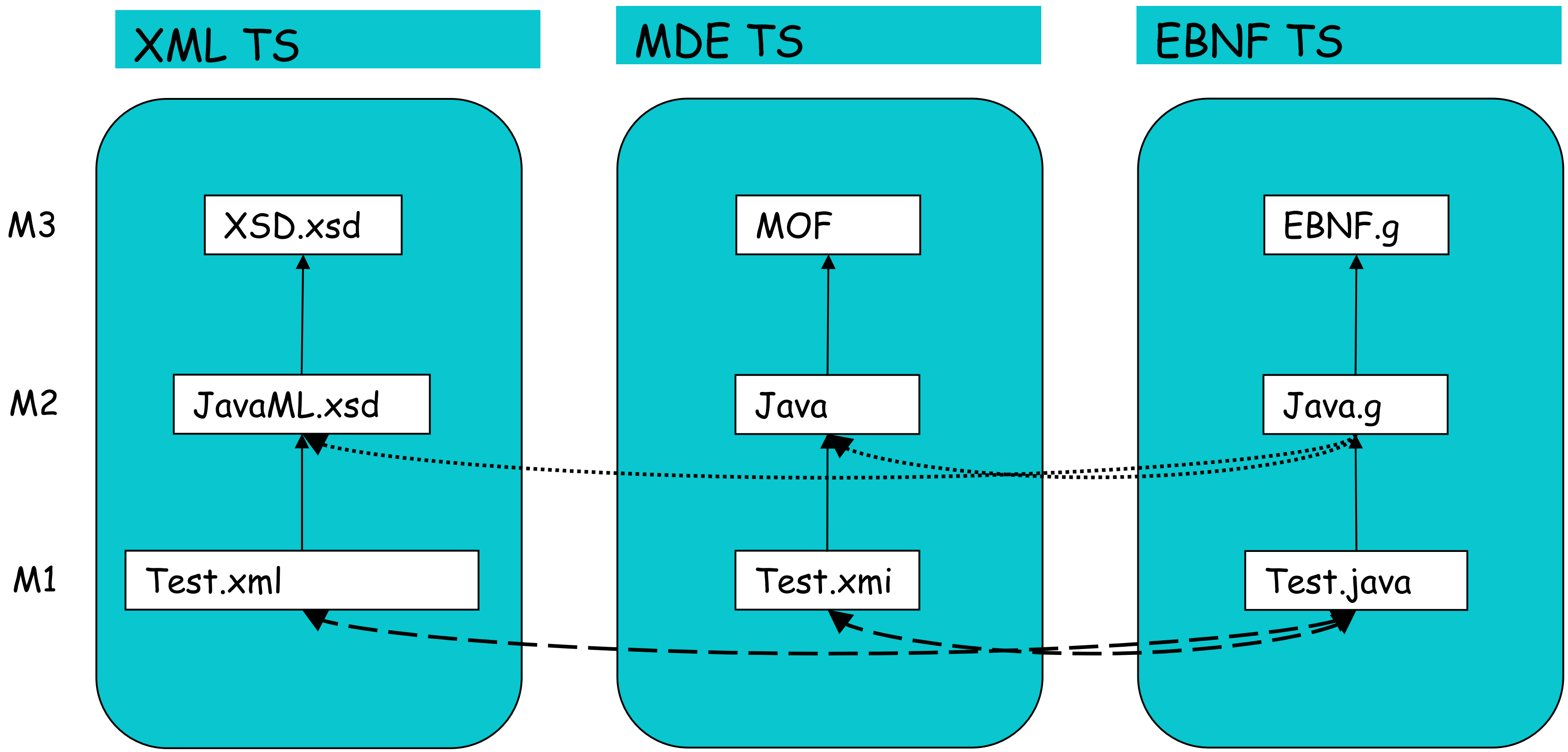
Communications between technical spaces

Any software artifact can be **injected** into a model

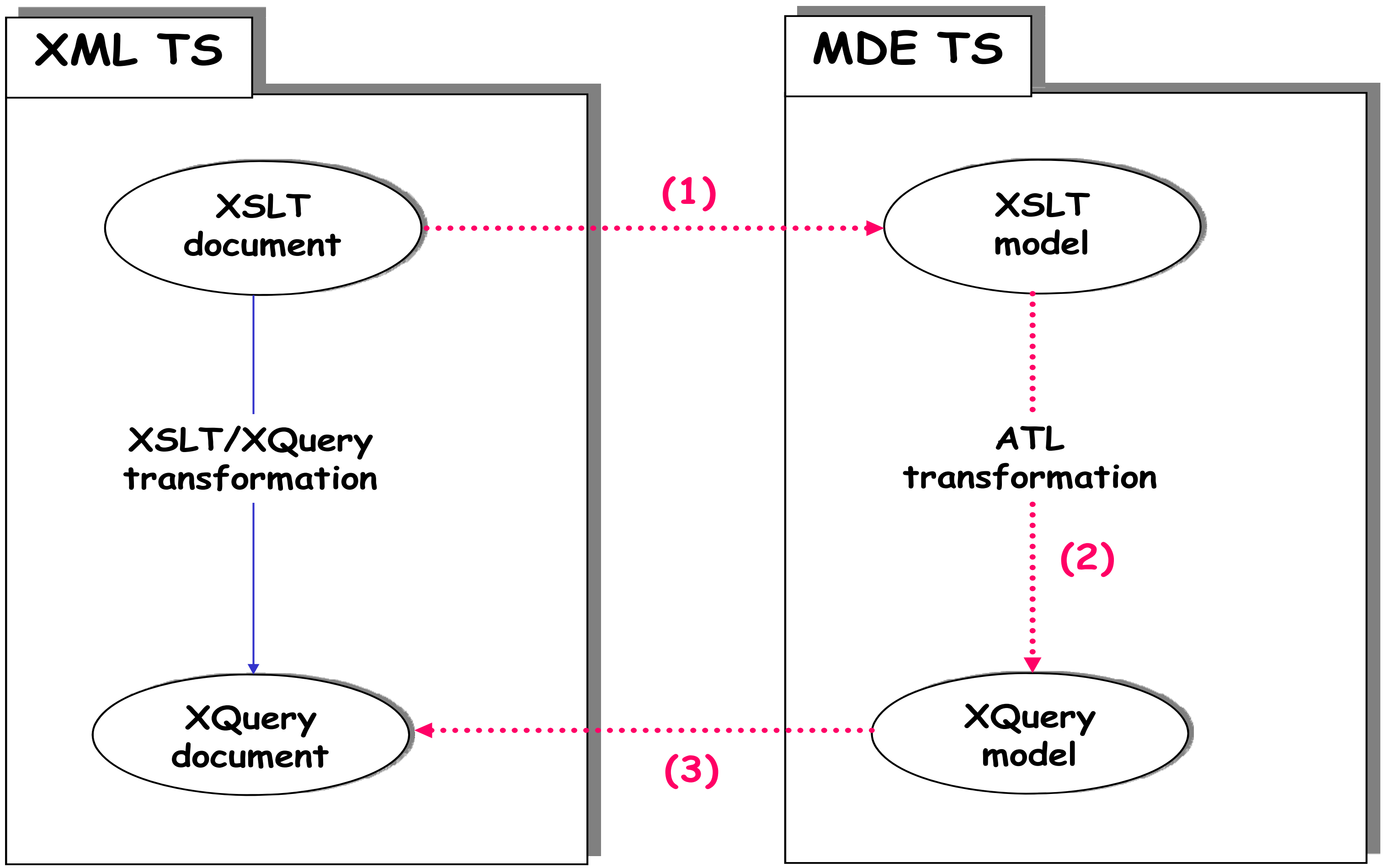
Any model can be **extracted** to a software artifact

Cost of solving a problem inside a TS vs. Exporting it to another TS, solving it and importing back the solution.

Examples of projections from EBNF to MDE and XML TSs



Economic equation



Transformation in the Prolog Technological Space

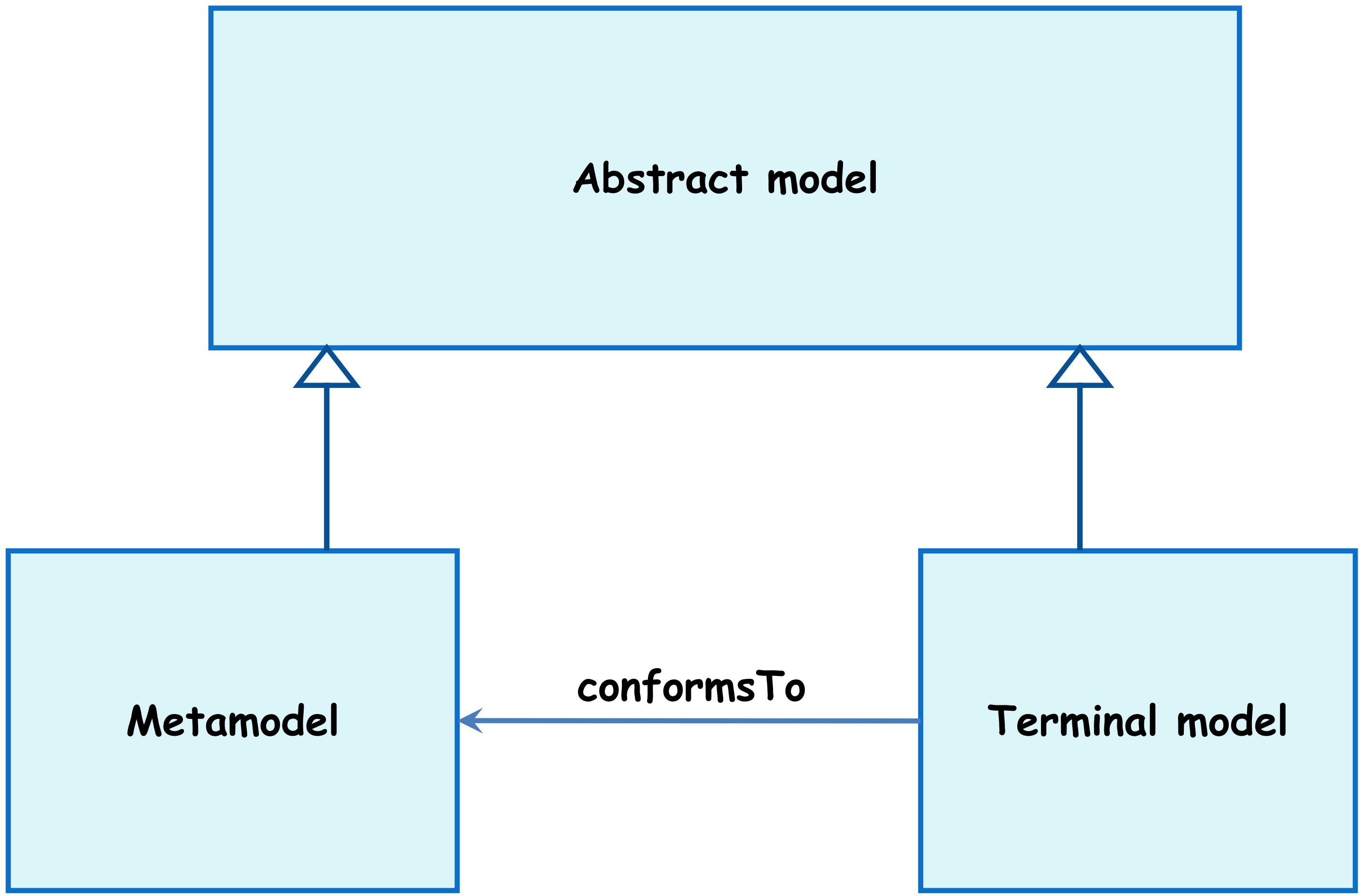
XIV → 14

```
Arab2Roman(U,W) :-
    tr(U,['?','M','D','C','L','X','V','I'], W).
    tr([], S, []).
    tr(U o [0], S o [V,I], W) :-
    dif([0] o U, U o [0]), tr(U, S, W).
    tr(U o [1], S o [V,I], W o [I]) :- tr(U, S, W).
    tr(U o [2], S o [V,I], W o [I,I]) :- tr(U, S, W).
    tr(U o [3], S o [V,I], W o [I,I,I]) :- tr(U, S, W).
    tr(U o [4], S o [V,I], W o [I,V]) :- tr(U, S, W).
    tr(U o [5], S o [V,I], W o [V]) :- tr(U, S, W).
    tr(U o [6], S o [V,I], W o [V,I]) :- tr(U, S, W).
    tr(U o [7], S o [V,I], W o [V,I,I]) :- tr(U, S, W).
    tr(U o [8], S o [V,I], W o [V,I,I,I]) :- tr(U, S, W).
    tr(U o [9], S o [X,V,I], W o [I,X]) :- tr(U, S o [X], W).

U ← [1,2,3,4]
Arab2Roman(U,W).
W = ['M','C','C','X','X','X','I','V'].
```

MDA alternative :
(1) building a source metamodel for arab numbers and a target metamodel for roman numbers
(2) building a QVT-like transformation for translating one model into another one.
(3) Building all the model2text and text2model inter-TS translations.

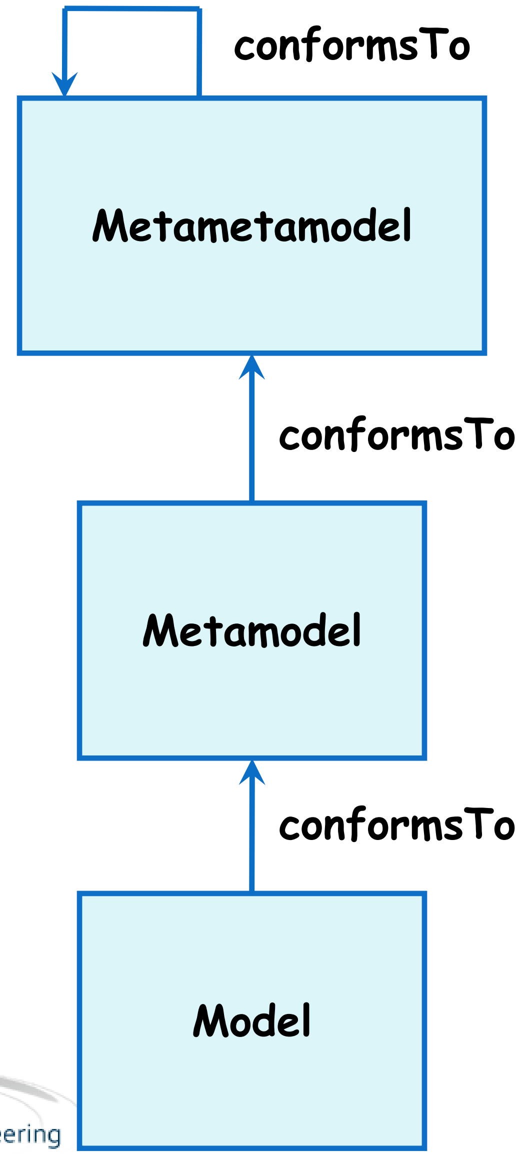
Abstract Models



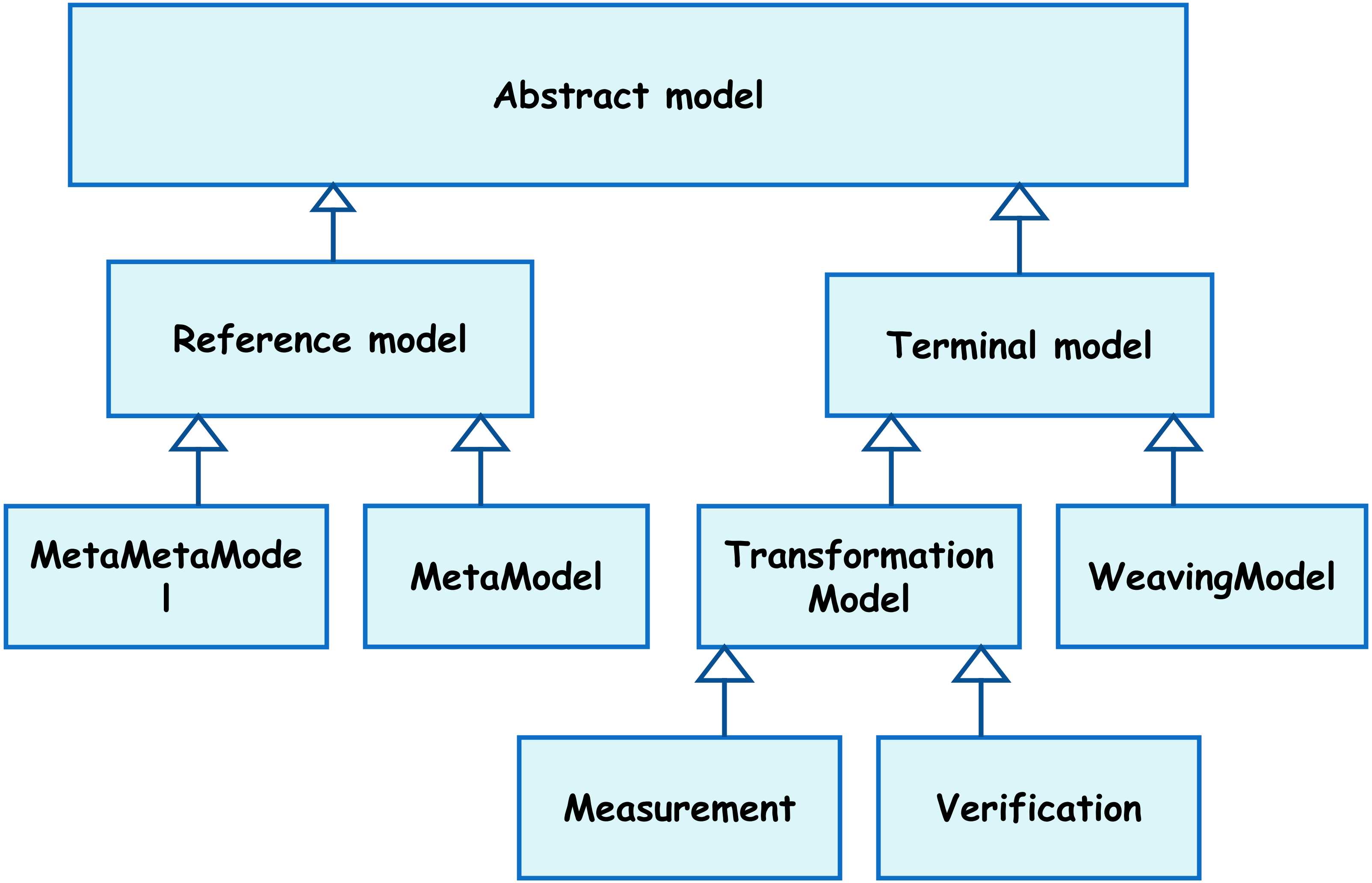
Precise Comparison of Technical Spaces

- Several operations possible in the MDE Technical Space:
  - ✓ Transforming a terminal model in a terminal model
  - ✓ Transforming a metamodel in a metamodel
  - ✓ Transforming a metamodel in a terminal model (demotion)
  - ✓ Transforming a terminal model in a metamodel (promotion)
  - ✓ Transforming a metametamodel in a metamodel (demotion)
  - ✓ etc.
- Fundamental operations in MDE (transformations but also store/retrieve, etc.)
- Some of the regularities properties of MDE TS also present in other TSs
  - ✓ Transforming a grammar into a program or a program into a grammar
  - ✓ Transforming a grammar in another grammar
  - ✓ Transforming an XML document in an XML schema
  - ✓ Converting XSD schema to RelaxNG
    - ➢ http://debeissat.nicolas.free.fr/XSDtoRNG.php
  - ✓ Compare with the Atlantic metamodel zoo (various metametamodels)
    - ➢ http://www.emn.fr/z-info/atlanmod/index.php/Zoos

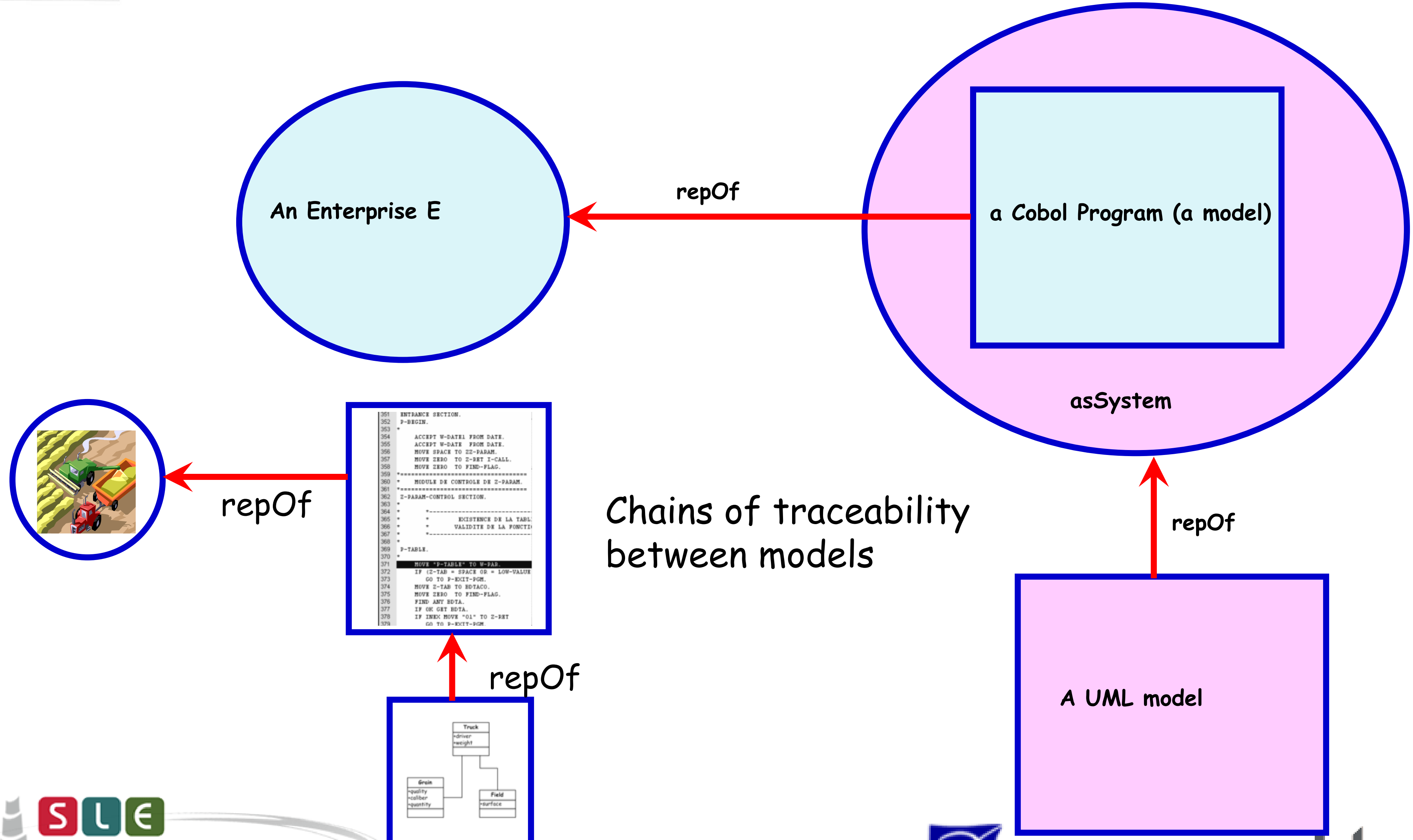
MetaMetaModels



But how about a definition of model extension?



A model of a model



The model of a model

The Correspondence Continuum

- *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.*
- *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]*

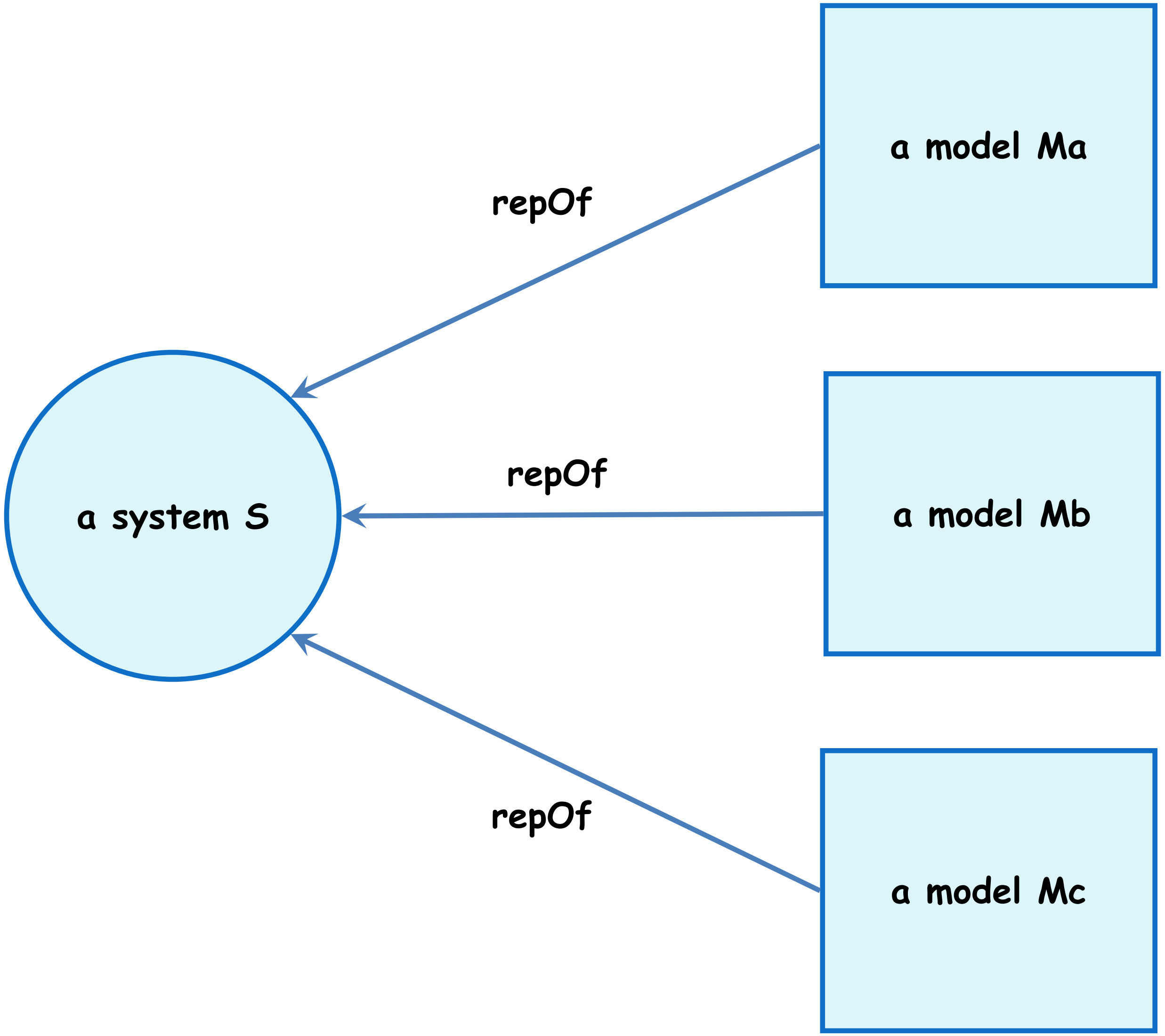
Data Semantics Revisited: Databases and the Semantic W

John Mylopoulos
University of Toronto

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

Multimodeling

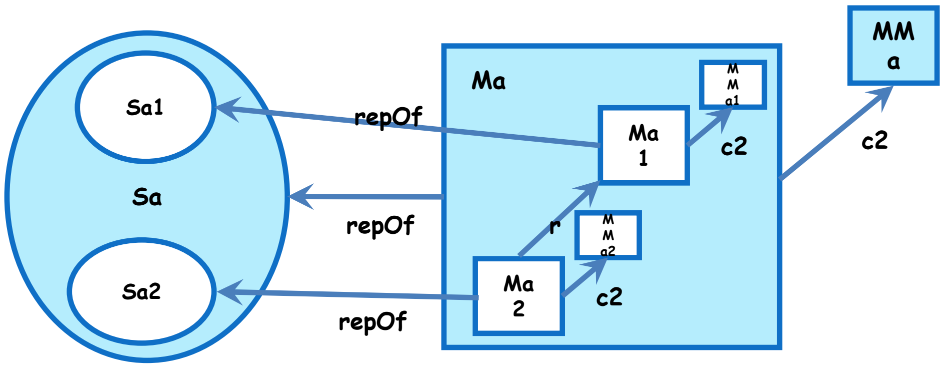
- ✓ Multimodeling is the joint exploitation of different models representing the same system.
- ✓ These models usually conform to different metamodels.
- ✓ Multimodeling suggests to manage complex systems by collaborative reasoning based on multiple models, each one encompassing a specific type of knowledge (e.g. structural, behavioral, functional) and representation.



Megamodeling

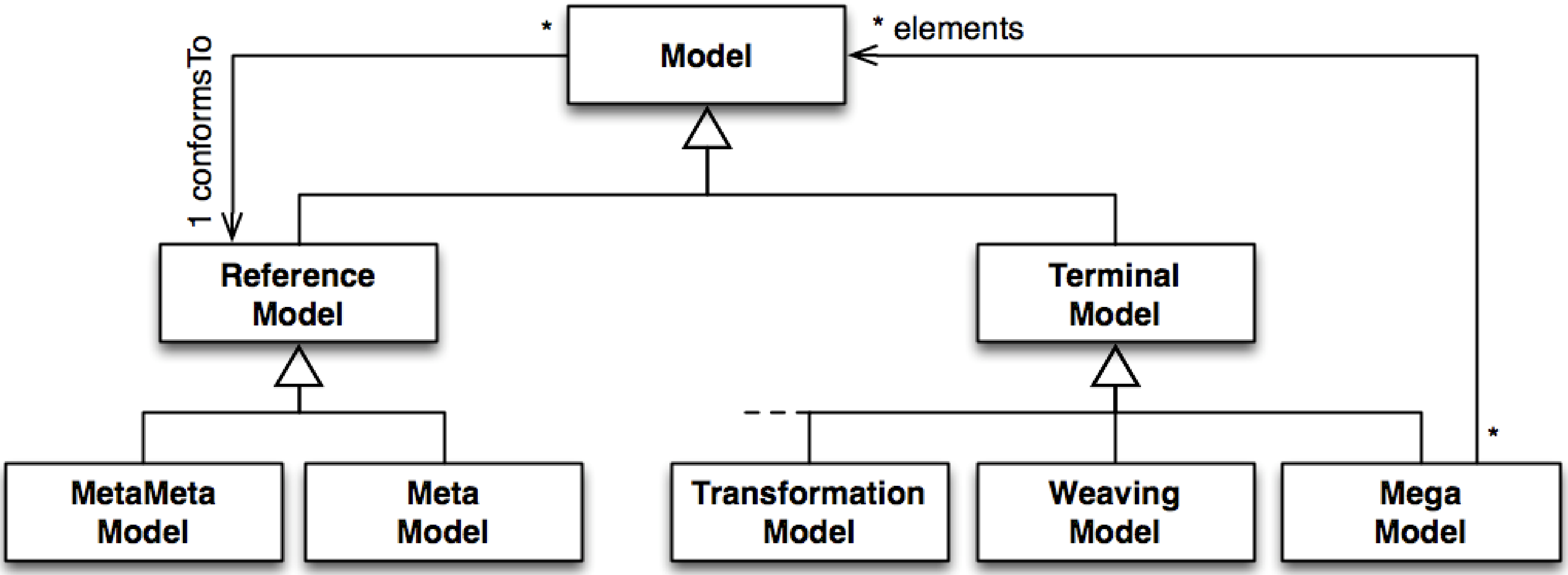
- With megamodeling, a given model may describe a set of other models, their metadata and mutual relationships between them.

- Since a megamodel is itself a model, this allows to represent deeply nested systems of systems.



Area of Interest (AoI)

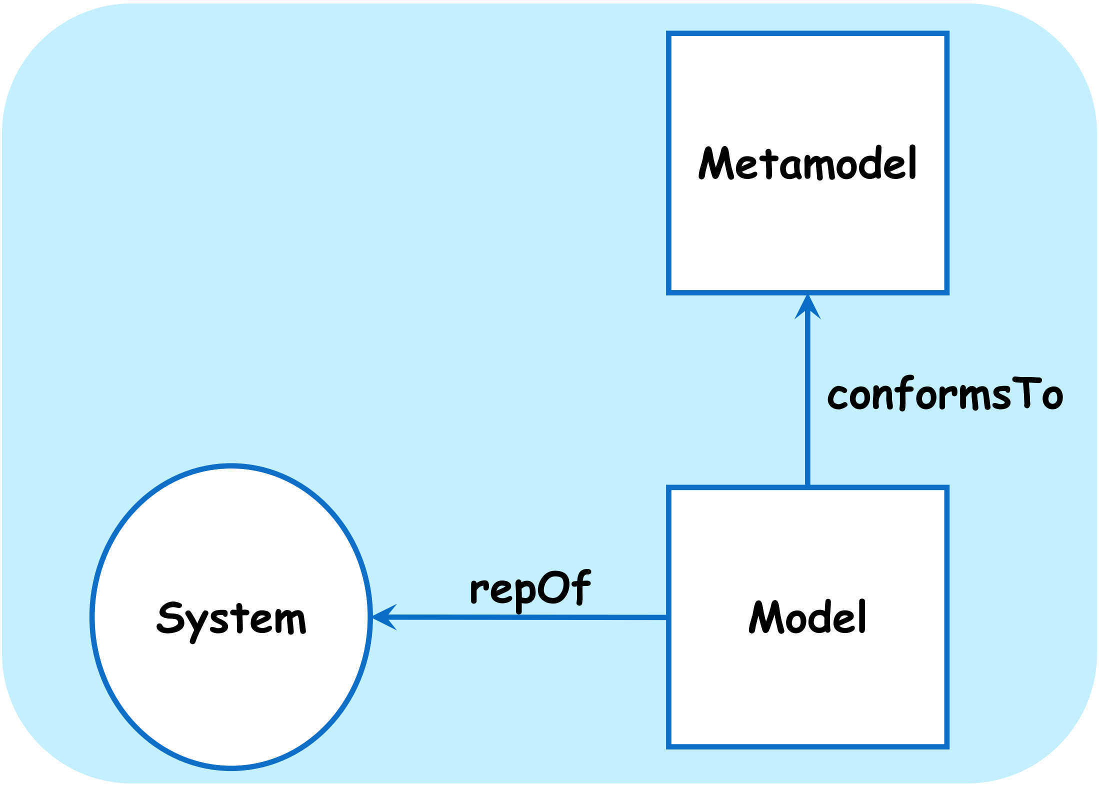
Classification

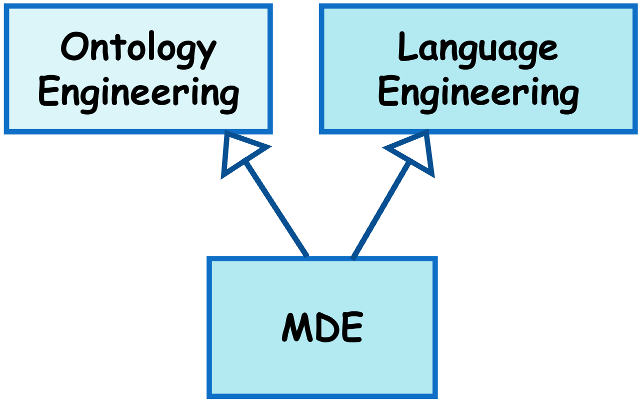



```
context MetaMetaModel inv: self.conformsTo = self
context MetaModel inv: self.conformsTo.oclIsKindOf(MetaMetaModel)
context TerminalModel inv: self.conformsTo.oclIsKindOf(MetaModel)
```

Utilization definition

Two orthogonal definitions for a model
- ✓ Structural definition (language dimension)
- ✓ Utilization definition (ontology dimension)





Utilization definition

The objective here is to define the possible usages of a model. Consequently, in all the present subsection, model will mean "terminal model".

- ✓ **Definition 6**. A system S is a delimited part of the world considered as a set of elements in interaction.
- ✓ **Definition 7**. A model M is a representation of a given system S, satisfying the substitutability principle (see below).
- ✓ **Definition 8**. (Principle of substitutability). A model M is said to be a representation of a system S for a given set of questions Q if, for each question of this set Q, the model M will provide exactly the same answer that the system S would have provided in answering the same question.

Principle of limited substitutability according to Minsky

"If a creature can answer a question about a hypothetical experiment without actually performing it, then it has demonstrated some knowledge about the world. …

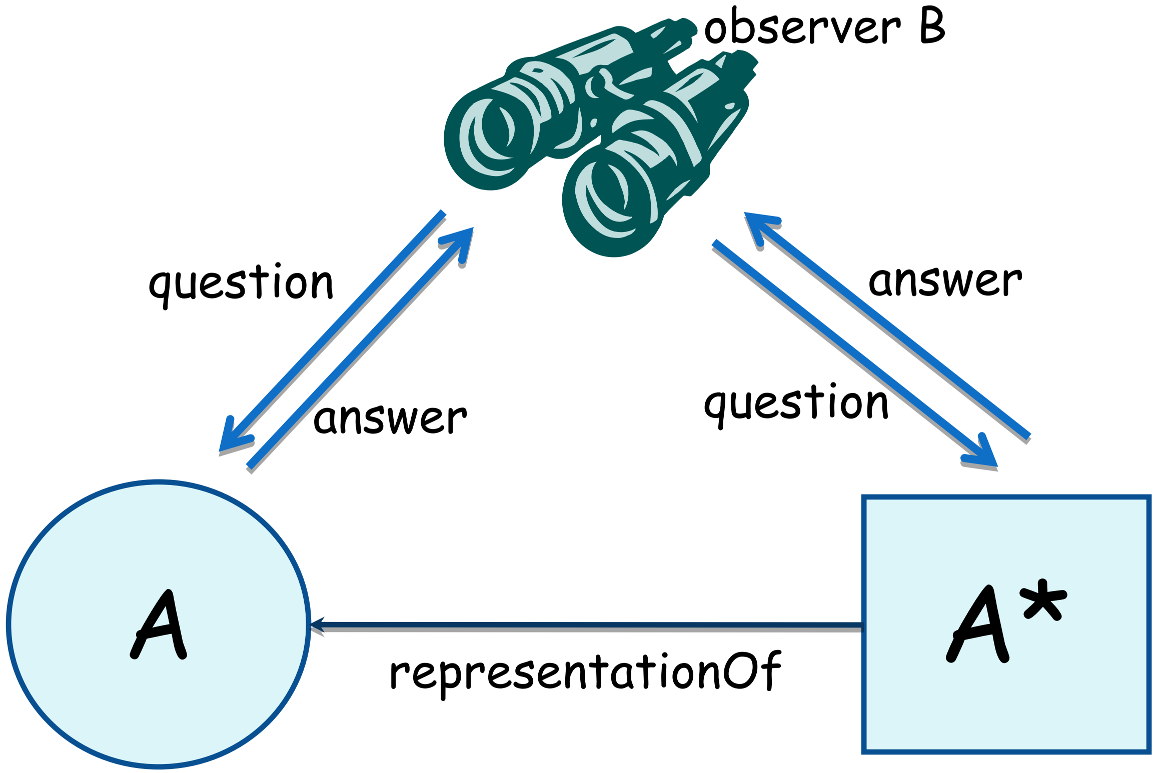
We use the term "model" in the following sense: To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A. …

It is understood that B's use of a model entails the use of encodings for input and output, both for A and A*.
If A is the world, questions for A are experiments. …
A* is a good model of A, in B's view, to the extent that A*'s answers agree with those of A's, on the whole, with respect to the questions important to B. …"

Marvin L. Minsky

Matter, Mind and Models Semantic Information Processing, MIT Press, 1968

Limited substitutability



We use the term "model" in the following sense: To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A.

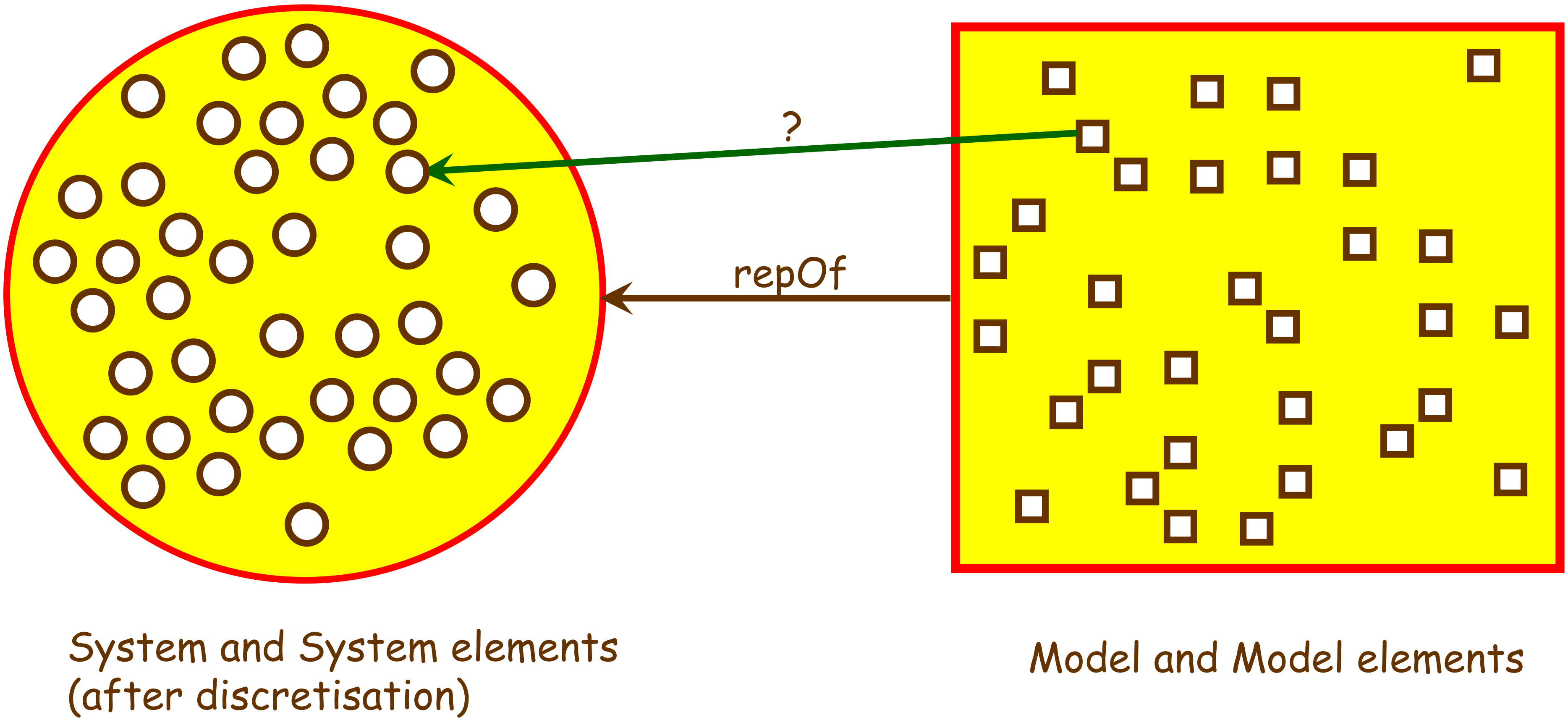
Taking the *representation* relation seriously

"*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*".

Brian Cantwell Smith **The Limits of Correctness**;
a paper prepared for the Symposium on Unintentional Nuclear War, Fifth Congress of the International Physicians for the Prevention of Nuclear War, Budapest, Hungary, June 28 – July 1, 1985.

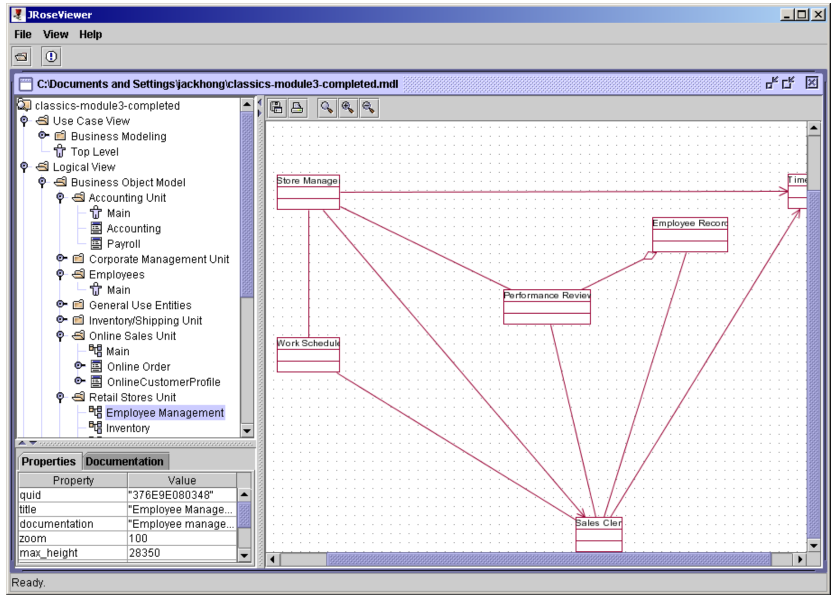
See also "***On the origin of objects***"

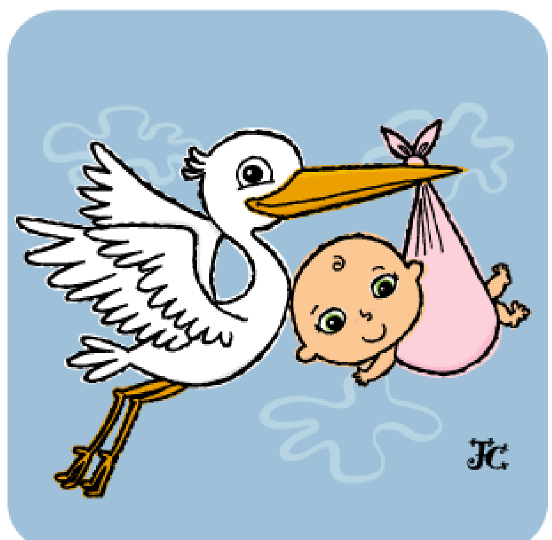
The "representation" relation




Where are models coming from?

First naïve answer



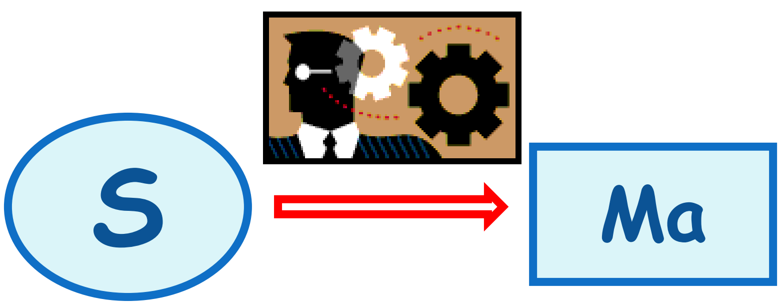


More precise answer

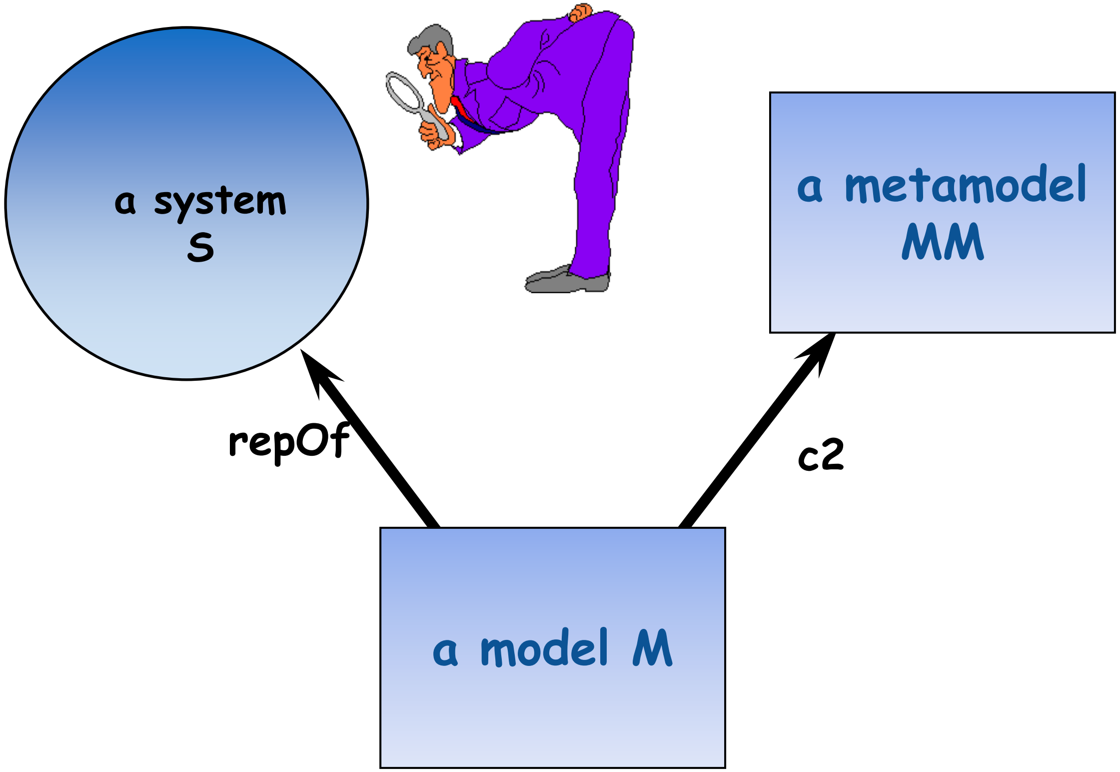
- Two possible sources from models (only two)

1. Transformation of other models

2. Observation of a system

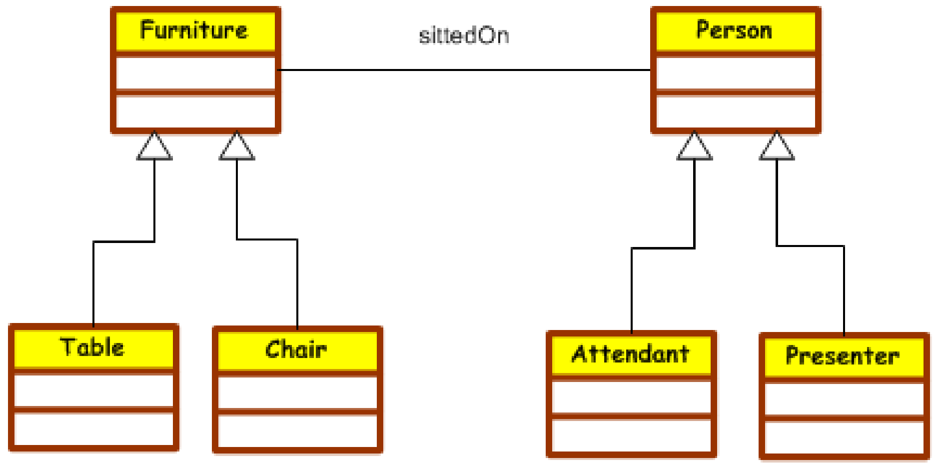


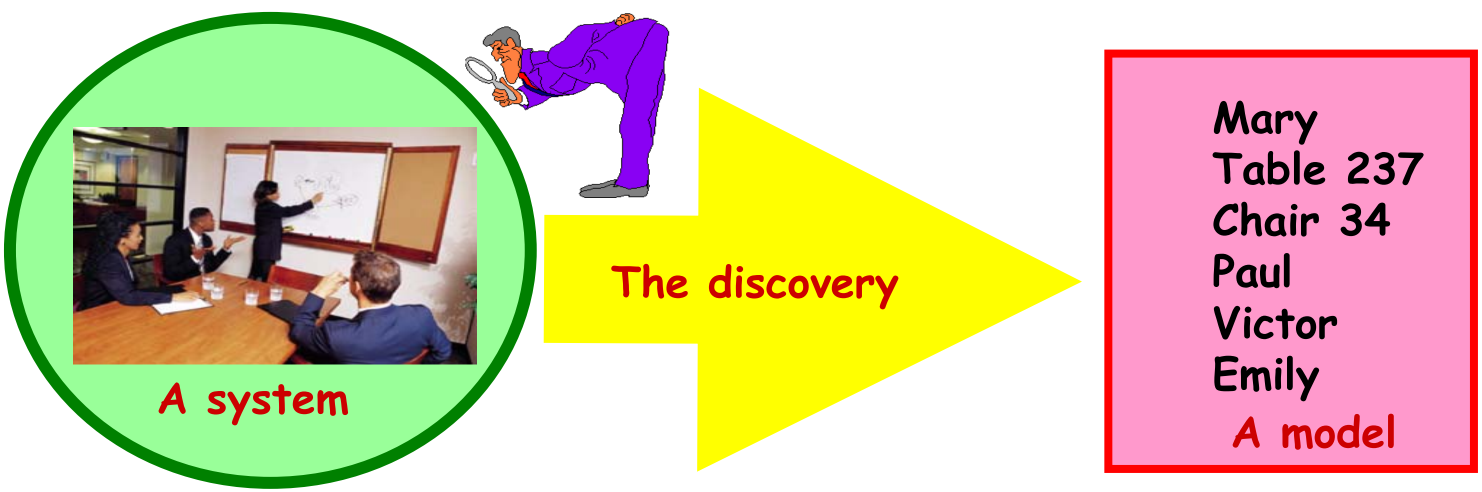
Model/System/Metamodel



A model is the result of observing a system, with respect to a given metamodel, with a given goal in mind.

Metamodels acting as filters





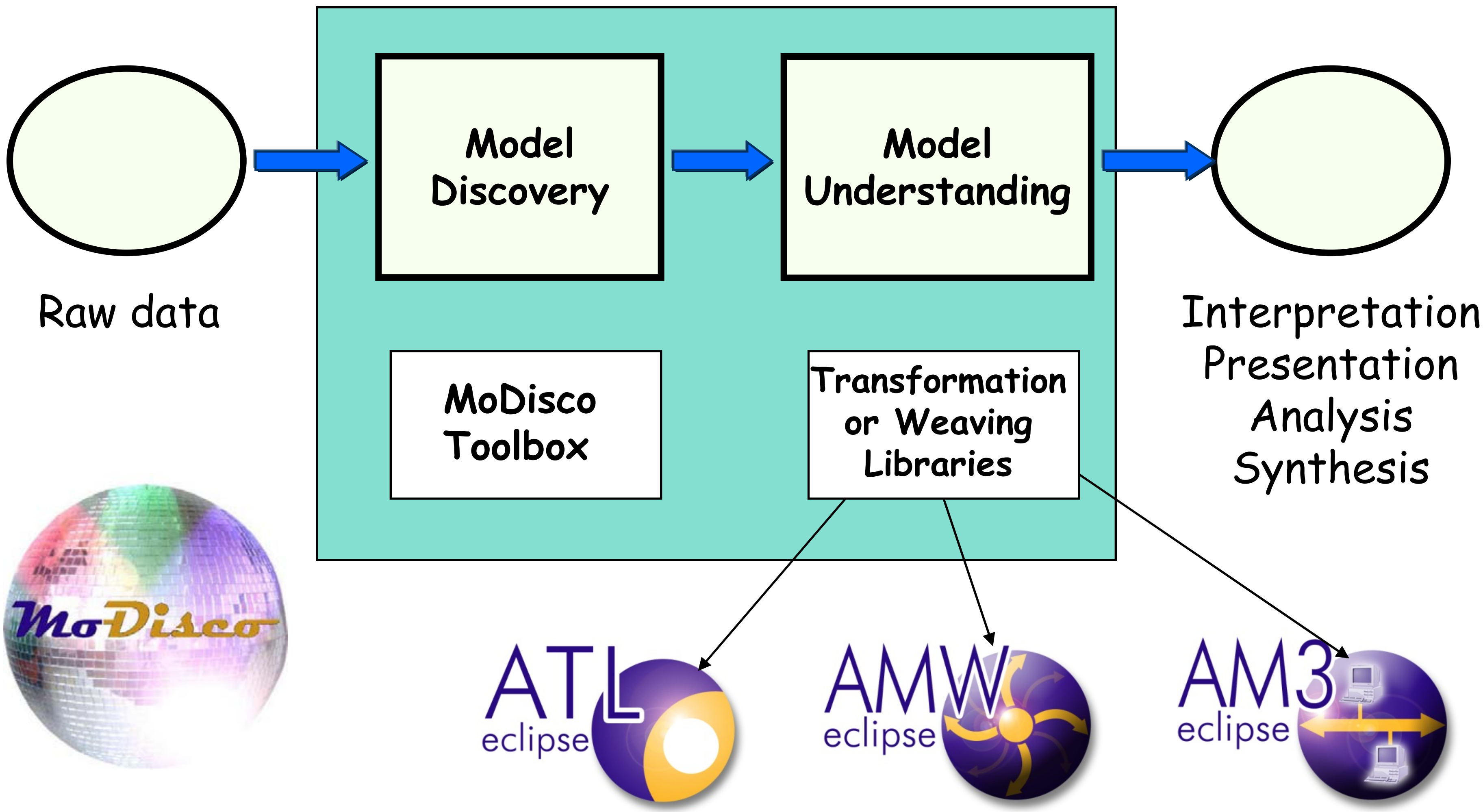
Metamodels are ontologies

- Definition : “In computer science and information science, an **ontology** is a formal representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain.”

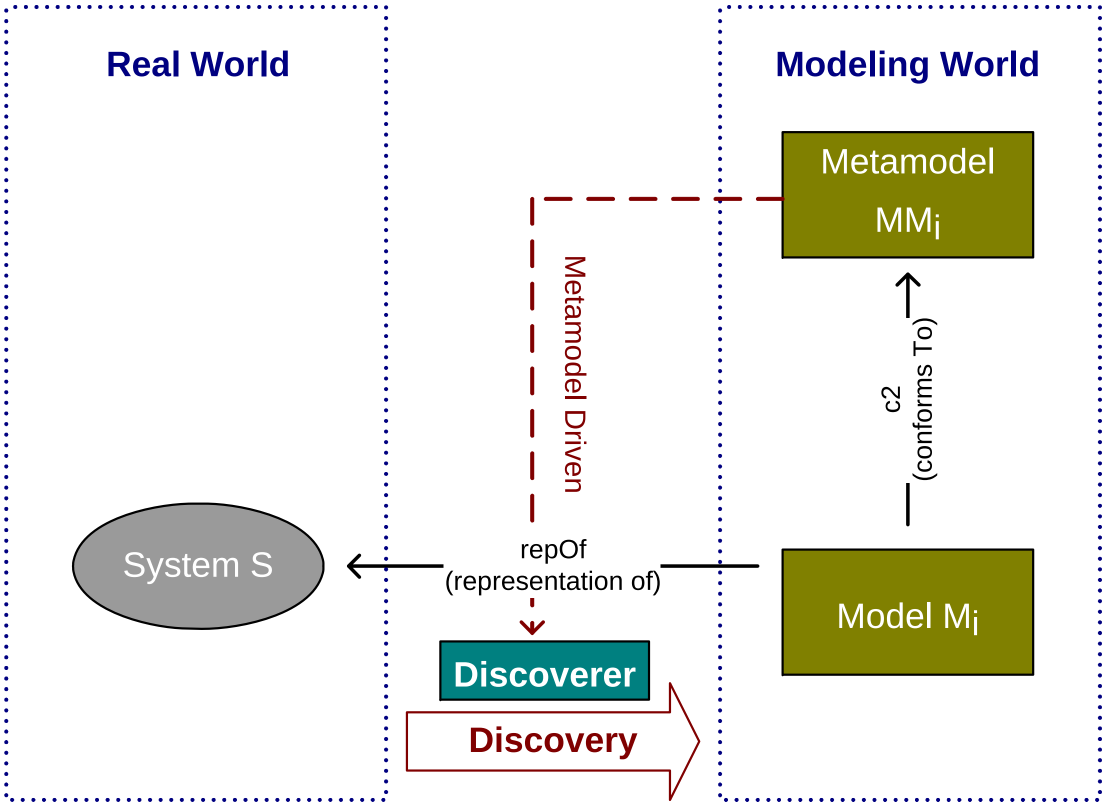
- s/ontology/metamodel/

- Could we similarly consider?
  - ✓ s/ontology/grammar/
  - ✓ s/ontology/XMLschema/

The MoDisco Eclipse Component (Model Discovery)

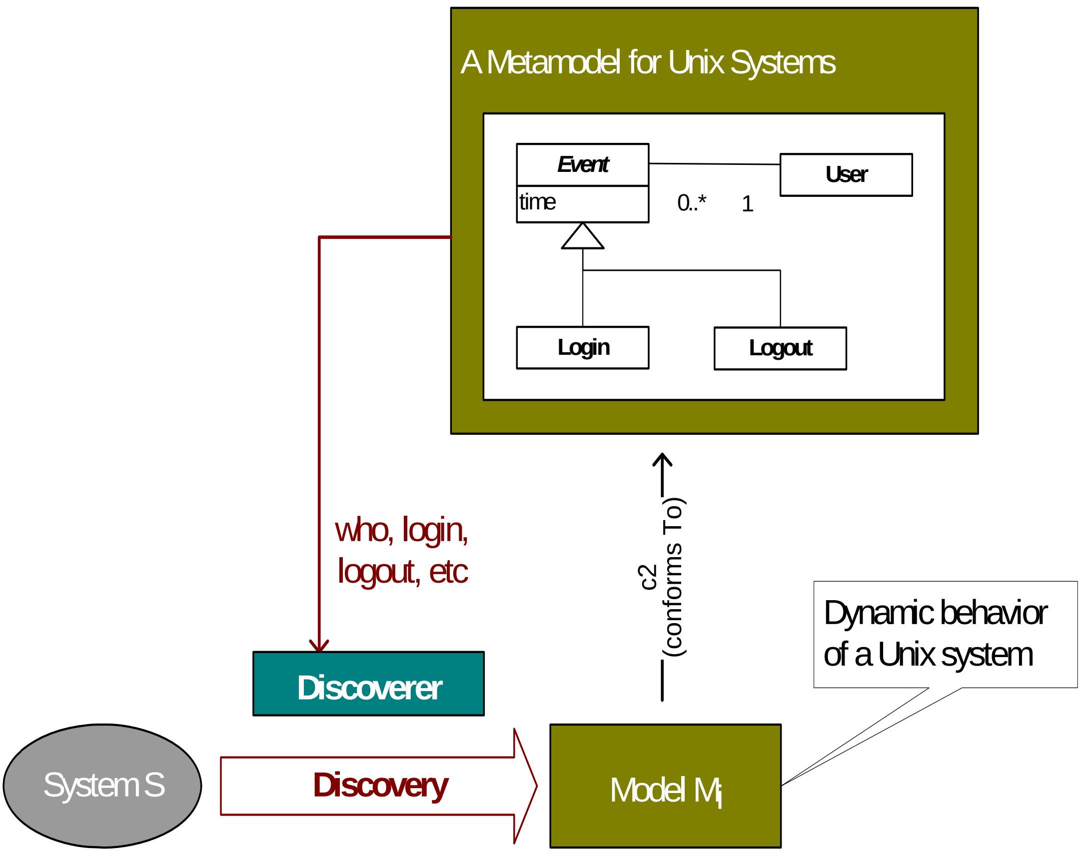


Discovery Principles



- Step 1:
  - Define the metamodel
- Step 2:
  - Create the "discoverer"
- Step 3:
  - Run the discoverer to extract model M_i from system S

Example



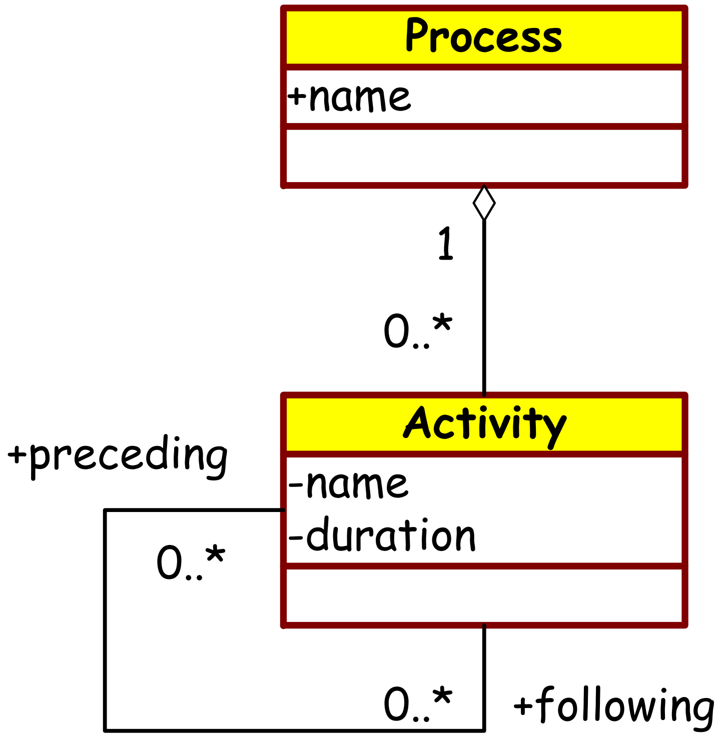
- Example of the Unix users' actions
- Study of the **dynamic** behavior of the system
  - Execution trace of the system

MODEL TAXONOMY

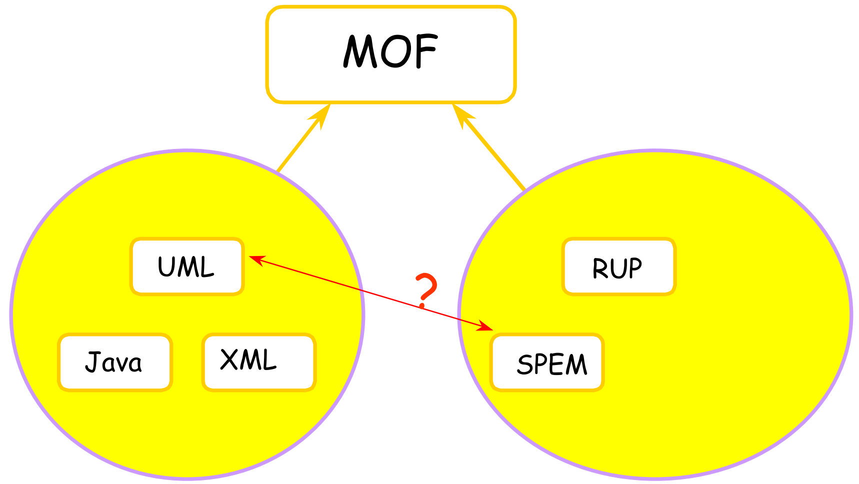
Classification of Models

- Models may represent any software artifact
- Classification of these artefacts provides a classification of the models and in turn of the possible applications of MDE
- Examples
  - ✓ Products and Processes
  - ✓ Analysis and Design
  - ✓ Code and Data
  - ✓ Applications and Platforms
  - ✓ PIMs and PSMs
  - ✓ Problems and Solutions

Product and Process Models



Product and process explicit models



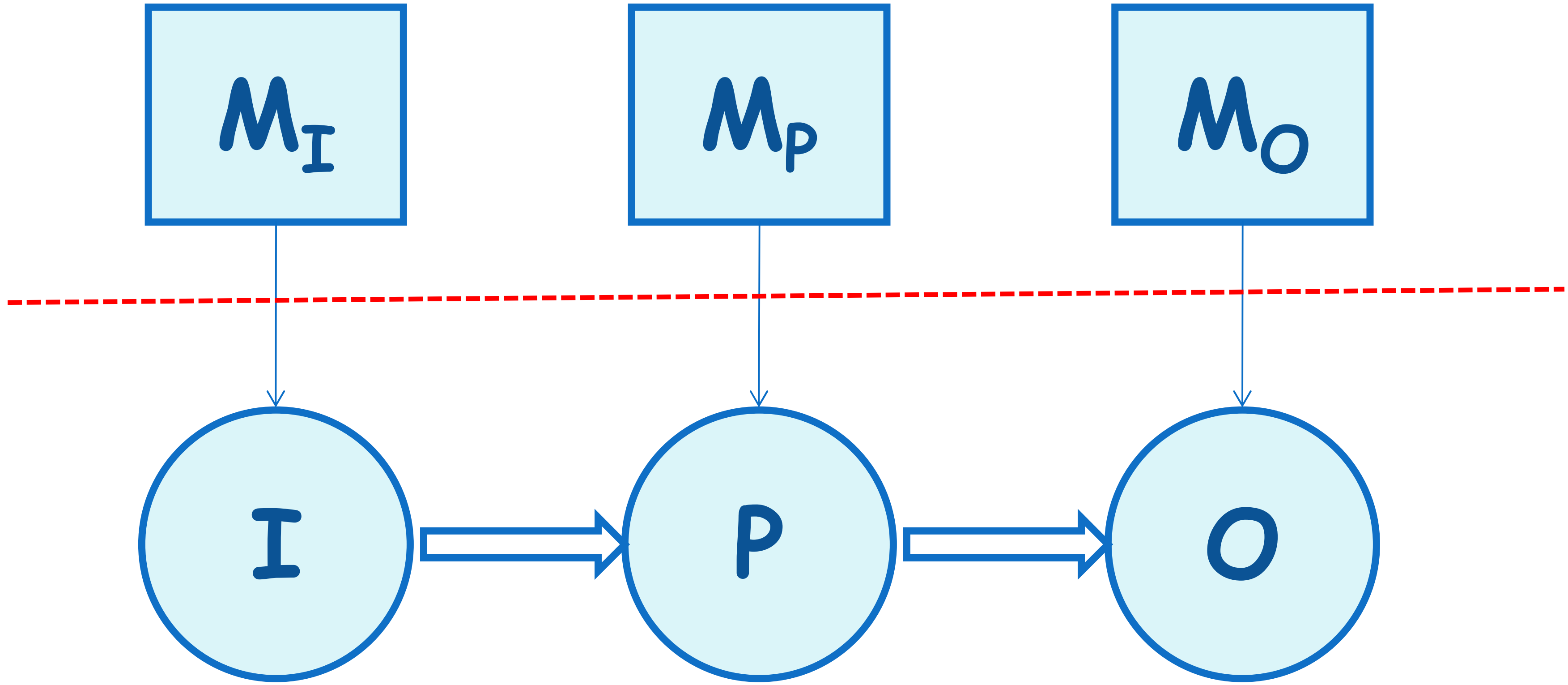
Who's doing what, when, how and why?

Static and Dynamic Models

- Most systems are dynamic
  - ✓ They evolve in time
  - ✓ Example : a washing machine
- Most models are static
  - ✓ They don't evolve in time
  - ✓ Example: a statechart of a washing machine
- Counter examples
  - ✓ Static system : Census results
  - ✓ Dynamic model : Simulation program

System ⇩ Model ⇨	Static	Dynamic
Static	Rare	Impossible
Dynamic	Frequent	Simulation

Code and Data Models



Problem and Solution Models

- Once upon a time, there was a team leader that was going on holidays. Before leaving, (s)he made the last recommendation to his/her small team of three young engineers. For the ongoing project, do not start coding in Java before the UML model is completely finished and that you all agree on this model.

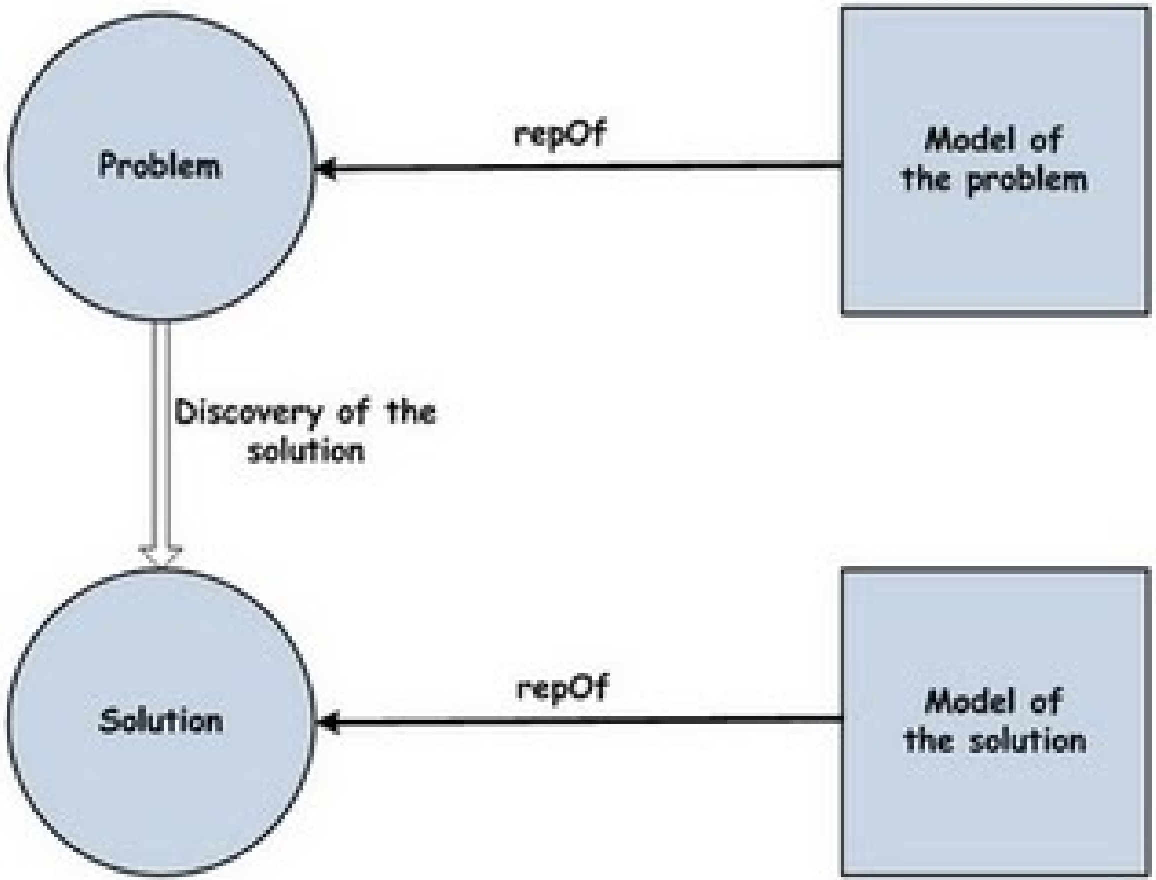
- On the Monday morning, as soon as s/he left, one of the engineers told the other ones of a wonderful discovery he made while twittering in the week end : a very powerful tool to automatically generate UML diagrams from Java code.

- The decision was rapidly taken and all three of them immediately started coding the problem in Java.

- Some days before the end of the holidays of their leader, all the Java code was used to generate UML diagrams and both the code and the UML diagrams were handled to the group leader.

- S/He was quite impressed at the level of detail of the UML model and the narrow correspondence between the code and the model.

Problem and Solution Models

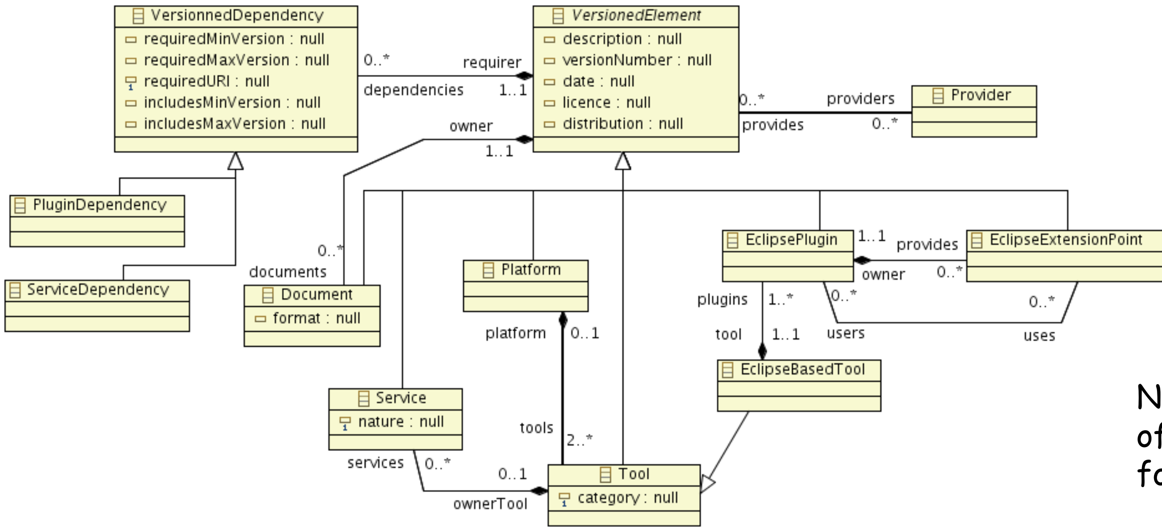


Languages for problem statement

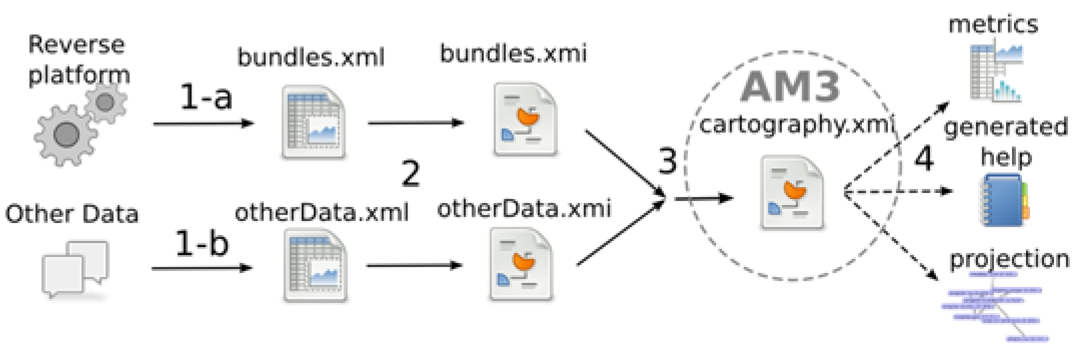
Languages for solutionstatement

*MDE is a unique chance to achieve the goal of separation of the **what** and of the **how**, for example in the educational context.*

Concrete platform models



Nearly 10 years after the introduction of PIMs and PSMs, we are still waiting for concrete platform definition models.



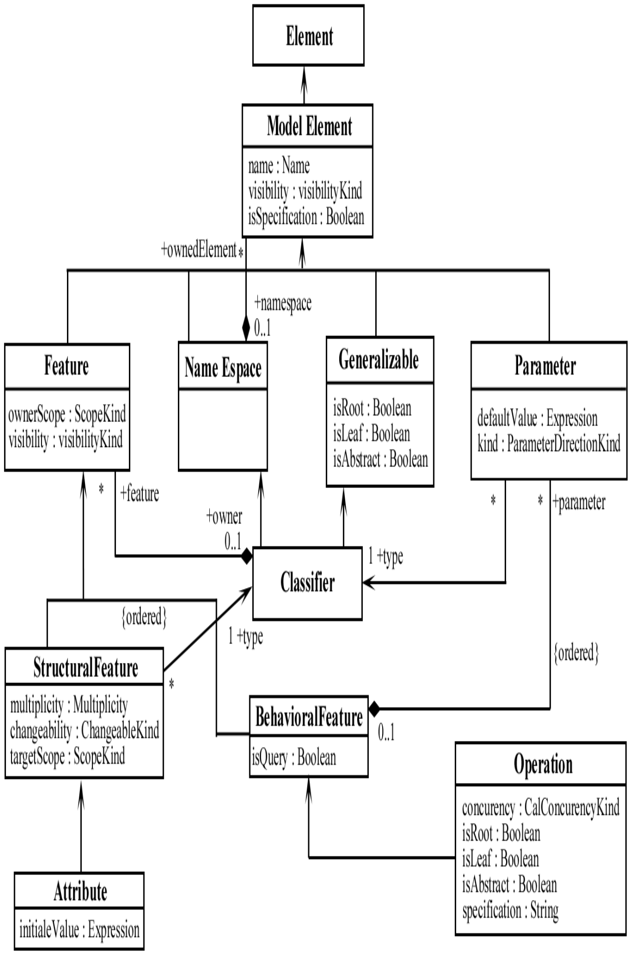
Eclipse

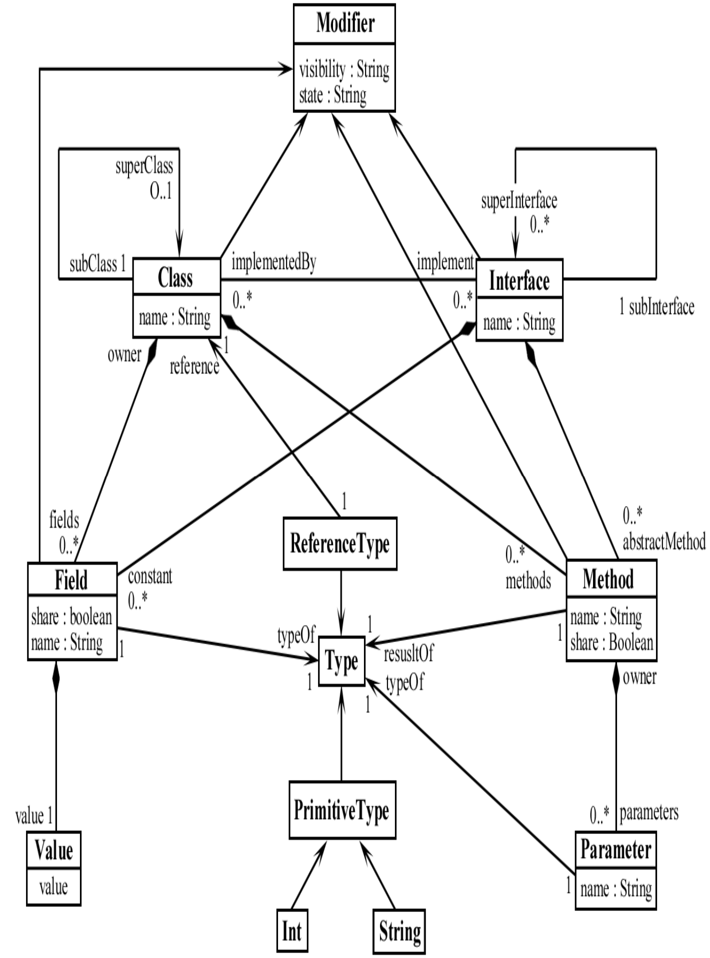
TopCased

SOME APPLICATIONS

Typical applications showing evolution of MDE practices

Example : UML to Java





UML Metamodel

Java Metamodel

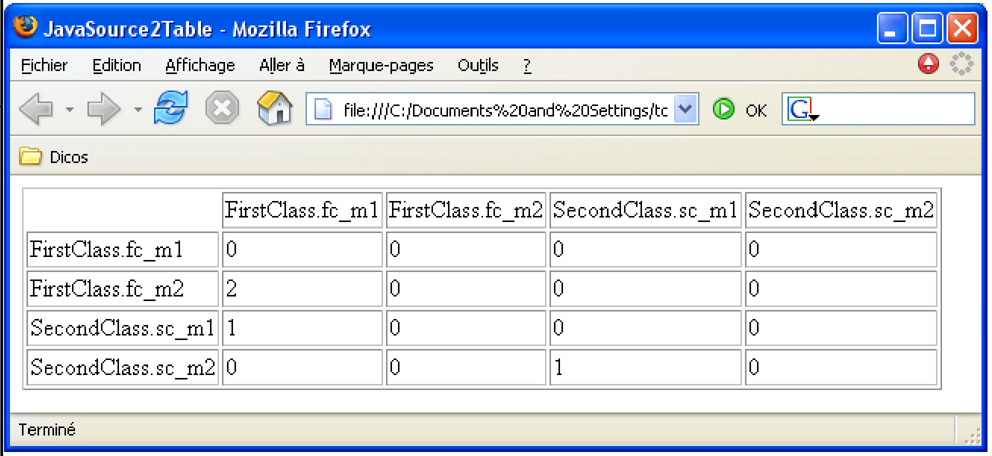
Example : Java to Excel

FirstClass.java

```
public class
    FirstClass {
  public void
    fc_m1(){
  }
  public void
    fc_m2(){
    this.fc_m1();
    this.fc_m1();
  }
}
```

SecondClasss.java

```
public class SecondClass
    {
  public void sc_m1(){
    FirstClass a = new
    FirstClass();
    a.fc_m1();
  }
  public void sc_m2(){
    this.sc_m1();
  }
}
```



	FirstClass.fc_m1	FirstClass.fc_m2	SecondClass.sc_m1	SecondClass.sc_m2
FirstClass.fc_m1	0	0	0	0
FirstClass.fc_m2	2	0	0	0
SecondClass.sc_m1	1	0	0	0
SecondClass.sc_m2	0	0	1	0

Example: XSLT to XQuery

XSLT

```
<xsl:stylesheet […] >
    <xsl:template match="/">
          <emps>
            <xsl:apply-templates
    select="employee"/>
          </emps>
    </xsl:template>
    <xsl:template match="employee">
      <xsl:if test="salary&gt;2000">
        <emp>
          <xsl:value-of select="name"/>
          <xsl:value-of
    select="firstname"/>
        </emp>
      </xsl:if>
    </xsl:template>
</xsl:stylesheet>
```

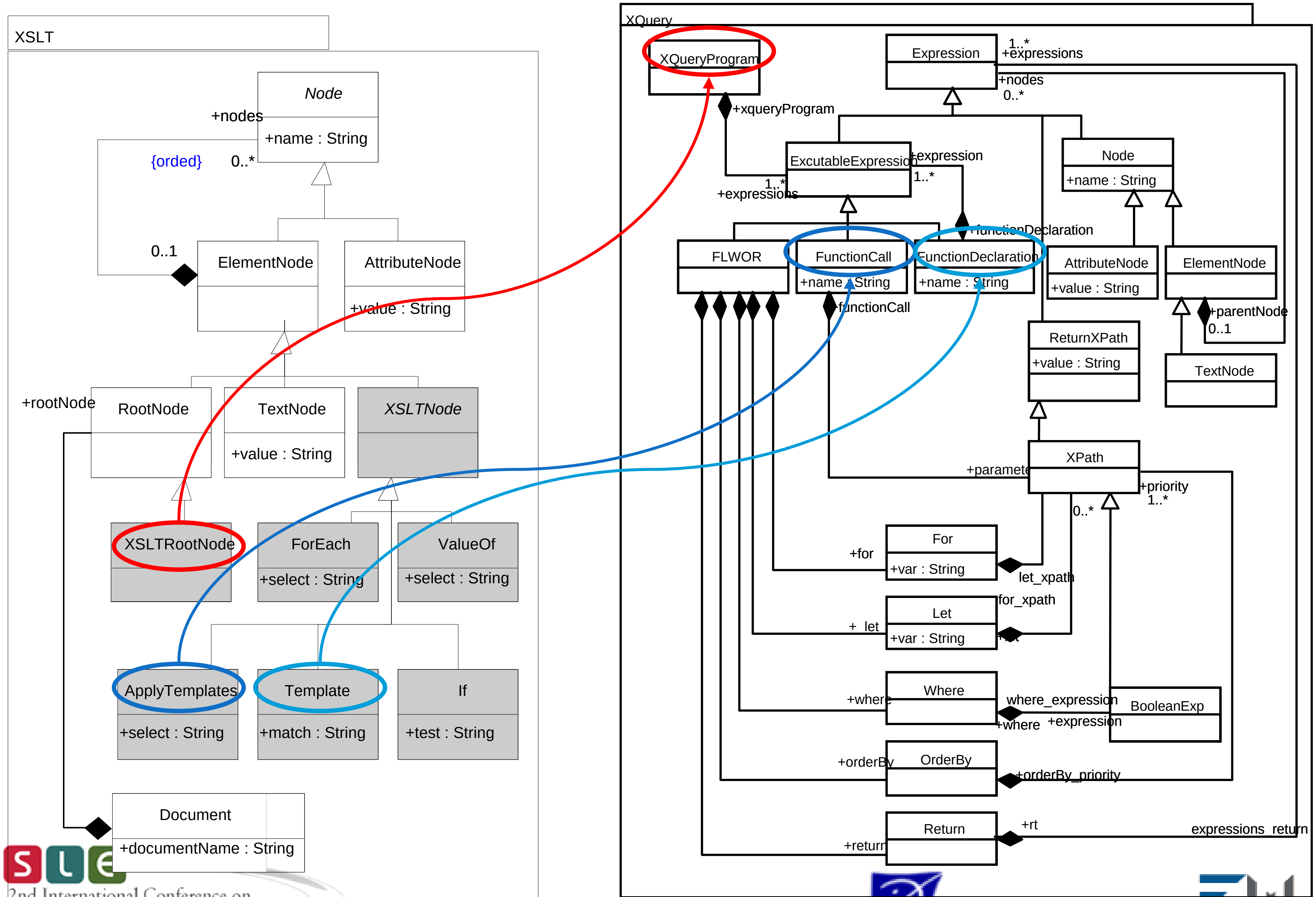
XQuery

```
define function fctemployee($paramVar)
{
  for $var in $paramVar
  return
    let $var := $var
    where $var/salary>2000
    return

     <emp>{$var/name}{$var/firstname}</emp
     >
}
for $var in document("xmlFile.xml")/*
return

     <emps>{fctemployee($var/employee)}</e
     mps>
```

Metamodel-driven transformation in ATL: XSLT & XQuery metamodels



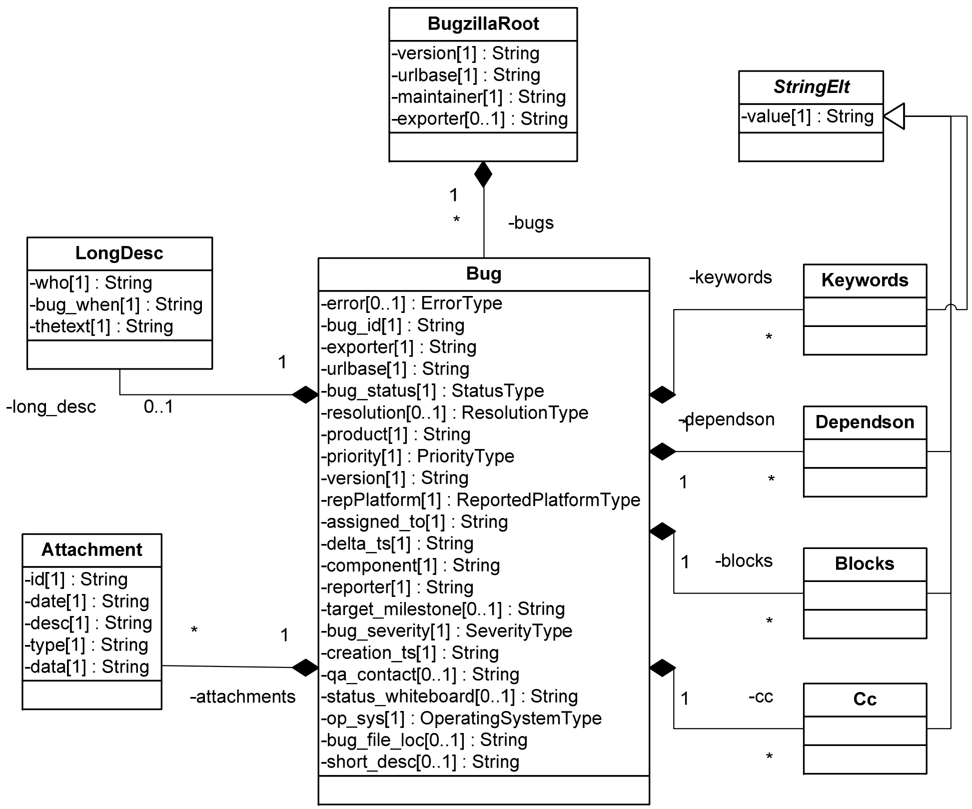
Example: Bugzilla to Mantis

About 30 open source tools to choose among for bug tracking

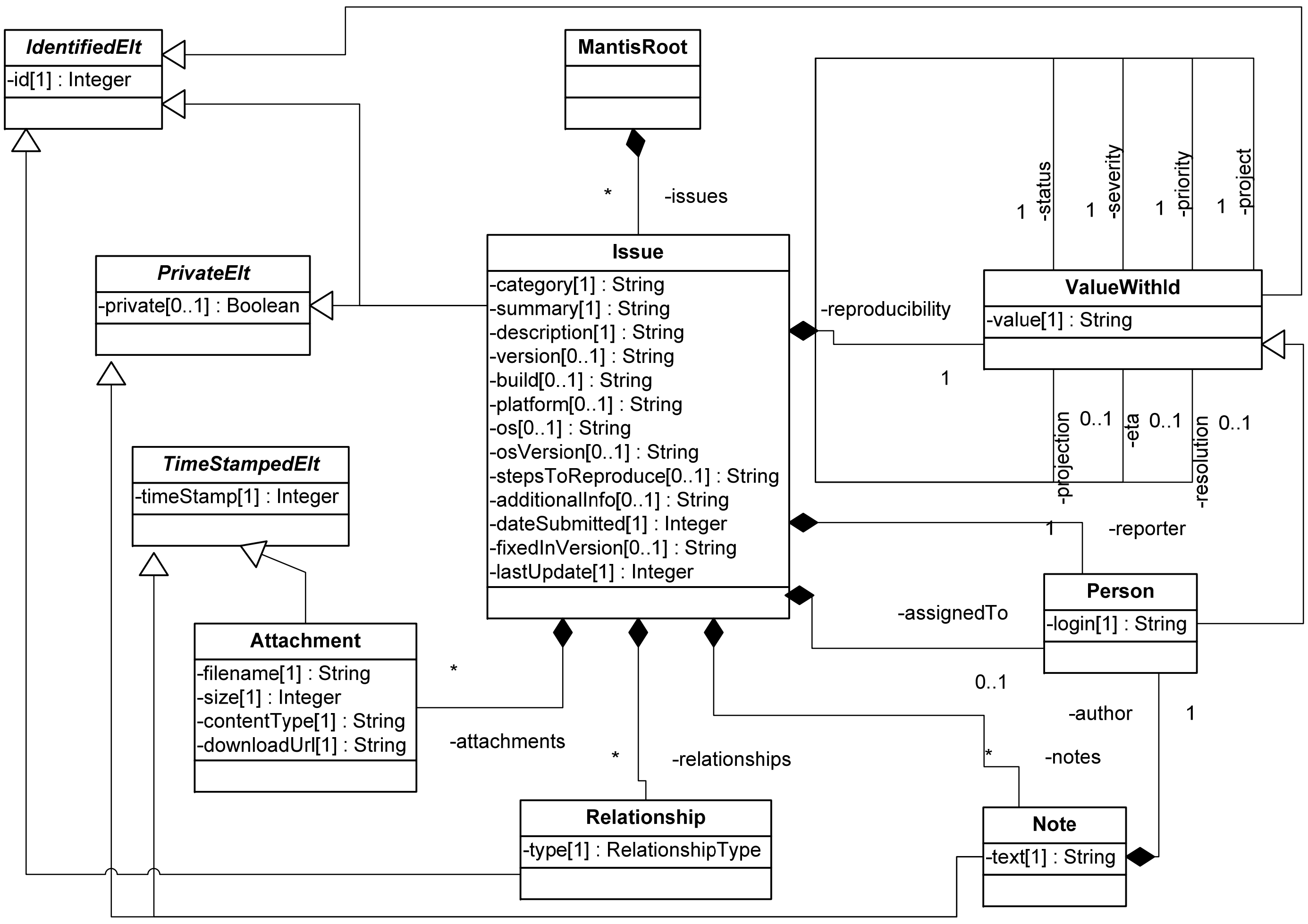
Tool	Lang	Ver	Cust	Temp	Search	RSS	Not	Rep	Hist	Attach	Updated	Demo	Score
ASP.NET Starter	C#/VB		Yes		Yes							No	2
Bug-a-Boo	CGI			Yes	Yes		Yes				Feb 05	Yes	4
Bug Base	Java										Aug 03		0
BugIn	PHP										May 04		0
Bugs Online	ASP				Yes			Yes	Yes		Jan 02	No	3
BugTracker	Java										Apr 01	No	0
BugTracker.NET	C#		Yes	No	Yes	No	Yes	Yes	Yes	Yes	Apr 05	No	7
Bugzilla	Perl		Yes		Yes	No	Yes		Yes	Yes	Jan 05	No	5
Eventum	PHP	1.5.4	Yes		Yes		Yes	Yes	Yes	Yes	Jun 05	No	7
EZ Ticket	PHP							Yes			Jan 04	No	1
Flyspray	PHP				Yes		Yes			Yes	Jan 05	Yes	4
GNATS											Mar 05	Yes	1
Issue Tracker	PHP			Yes			Yes			Yes	Feb 04	Yes	4
ITracker	Java		Yes		Yes		Yes	Yes	Yes	Yes	Aug 04	Yes	7
JTracker	Python		No								May 04	No	0
Mantis	PHP		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	May 05	Yes	10
Midge	Python				Yes						Oct 04	No	0
OpenPSA Support	PHP				Yes			Yes			Jan 05	Yes	3
OTRS	Perl			Yes	Yes		Yes		Yes	Yes	Oct 04	Yes	6
phpBugTracker	PHP			Yes	Yes			Yes		Yes	Nov 04	Yes	5
PloneCollector-NG	Python		Yes		Yes	Yes	Yes	Yes		Yes	Apr 04	No	6
Request Tracker	Perl		Yes	Yes	Yes	Yes	Yes		Yes	Yes	Feb 05	No	7
Roundup	Python		Yes	Yes	Yes		Yes		Yes	Yes	Mar 05	Yes	7
sBugs	Java										Jan 02		0
Scarab	Java		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Apr 04	Yes	9
Subissue											N/A	No	0
SugarCRM			Yes	Yes	Yes						Jun 05	Yes	5
Trac	Python				Yes	Yes		Yes	Yes	Yes	Nov 04	Yes	6
Whups	PHP			Yes	Yes				Yes	Yes		Yes	5
Workbench	PHP										Apr 02	No	1
Zope Issue Tracker	Python										Dec 03	Yes	1
Zwiki Tracker	Python								Yes			Yes	2

Different data models and functionalities

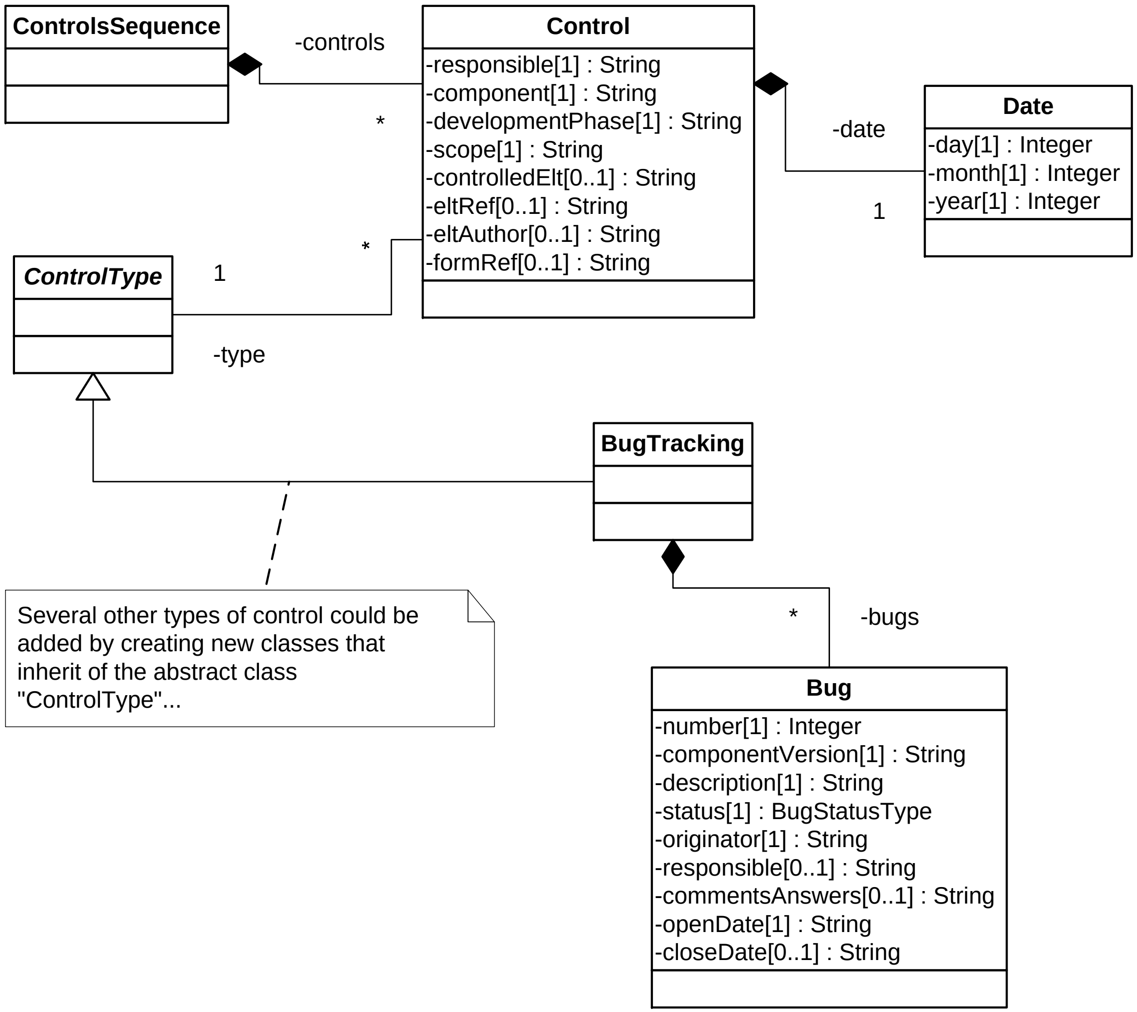
Bugzilla metamodel (simplified)



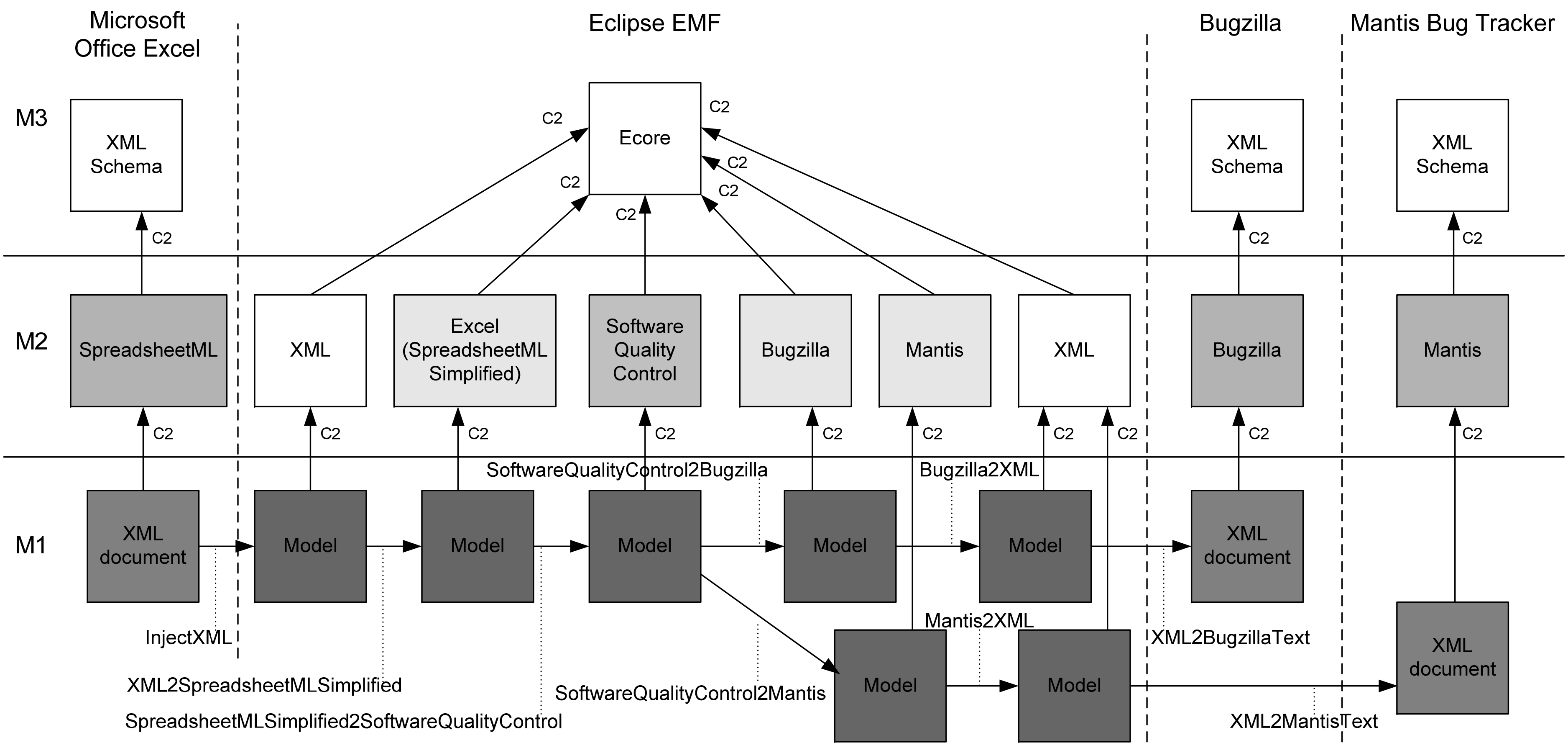
Mantis metamodel (simplified)



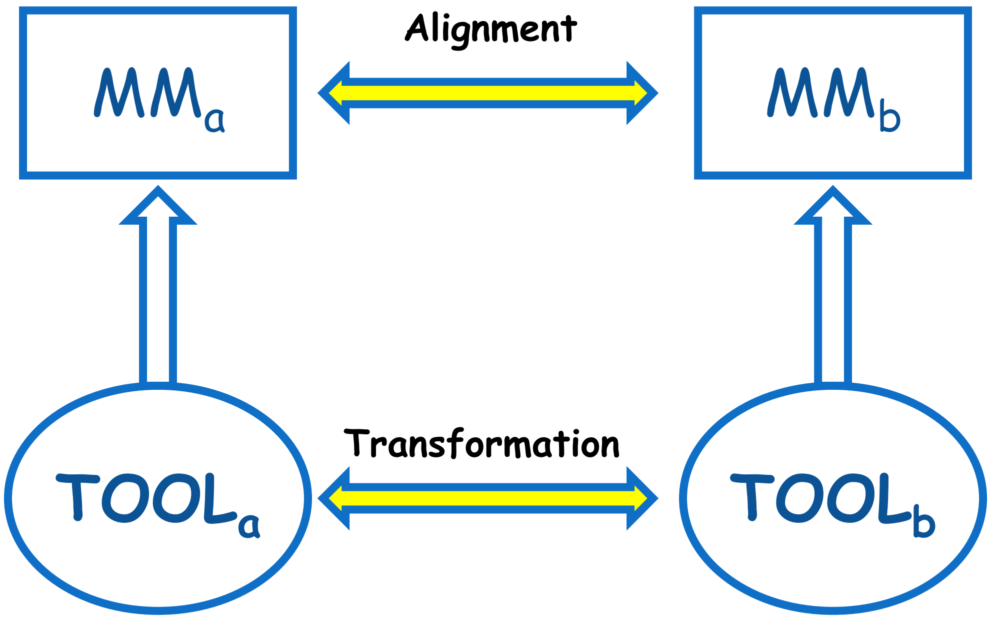
Bug control metamodel (pivot)

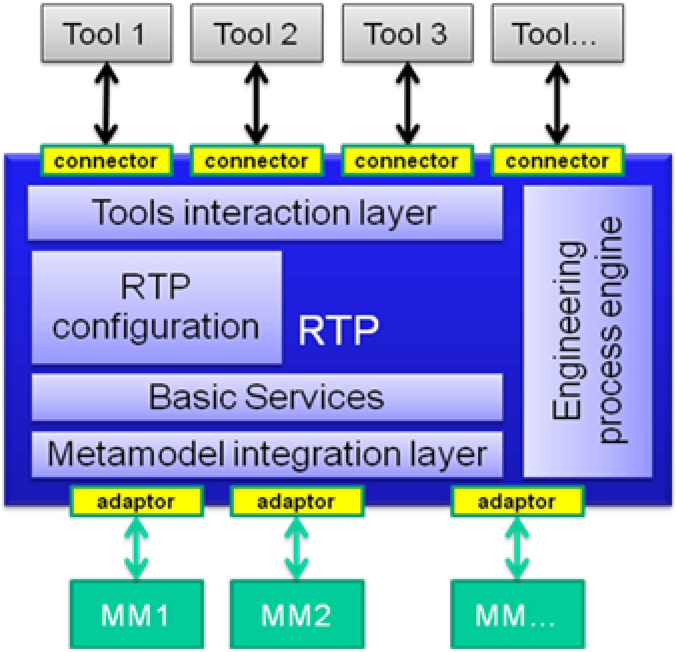


Excel-to-Bugzilla and Excel-to-Mantis bridges



Tool interoperability revisited

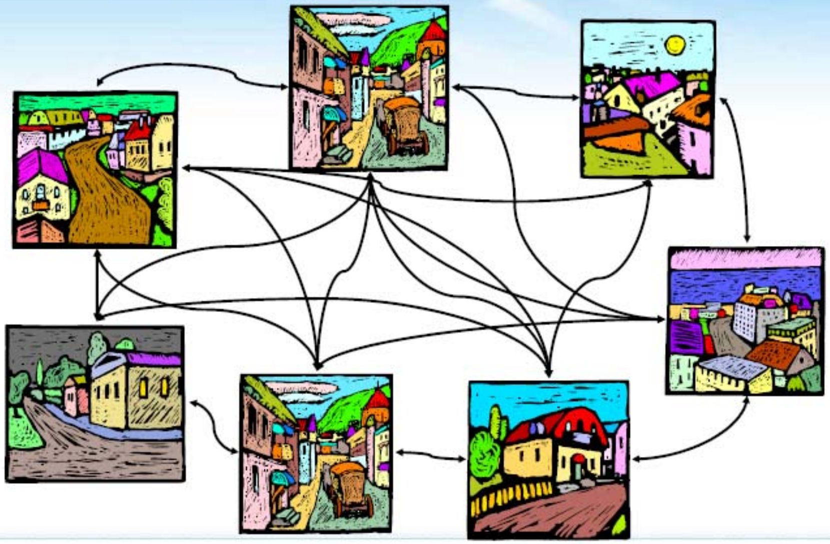




The CESAR project RTP :
(Reference Technology Platform)
a proposal

The «Village metaphor» by Antonio Vallecillo

Bridges between Semantic Domains



The Prolog village
The Petri net village
The Coloured Petri Net Village
The Z village
The B village
The Maude village
The Coq village
etc.

Expressing correspondences

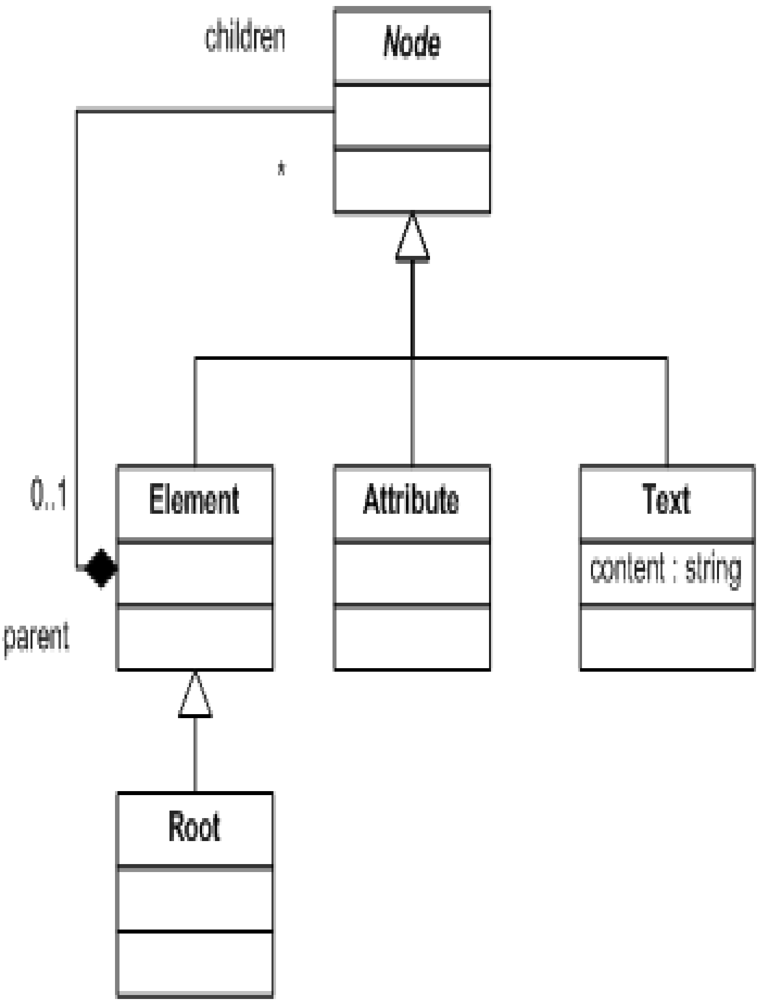
- As **Model Transformations**
  - Possible if correspondences can be expressed as functions
  - Pairwise consistency can be formally studied
    - One form of consistency involves a set of correspondence rules to steer a transformation from one language to another. Thus given a specification S_1 in viewpoint language L_1 and specification S_2 in viewpoint language L_2, a transformation T can be applied to S_1 resulting in a new specification $T(S_1)$ in viewpoint language L_2 which can be compared directly to S_2 to check, for example, for behavioral compatibility between allegedly equivalent objects or configurations of objects [RM-ODP, Part 3]
- As **Weaving Models**
  - Possible if correspondences are just mappings

EVOLVING MDE SCOPE

Some examples of feedback

- Textual vs Visual DSLs
- Partial Correspondances
- Higher order transformations
- Transformation graphs
- Infinite Models

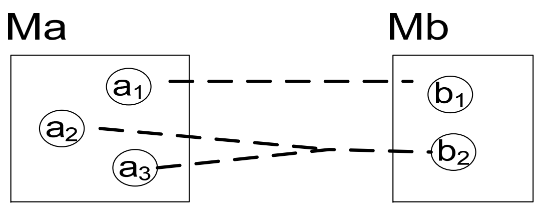
Visual vs. Textual Languages

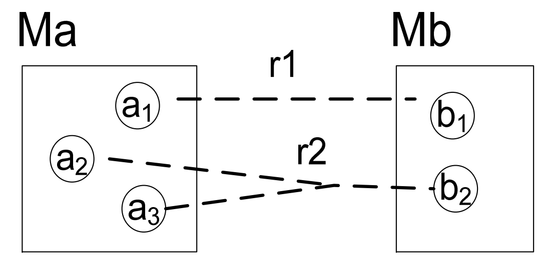


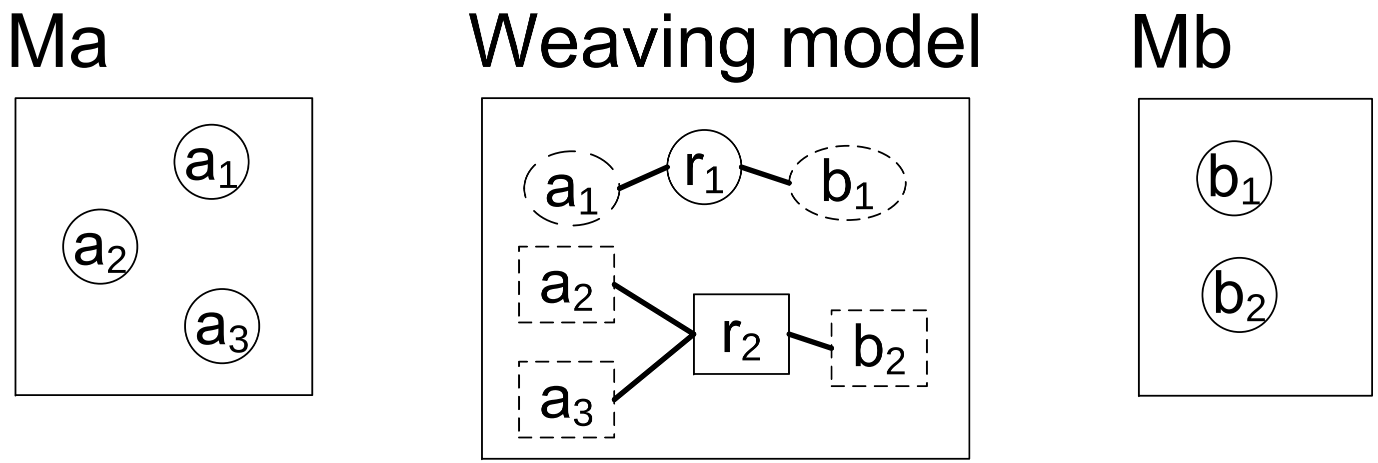

```
-- @name           XML
-- @version1.0
-- @domains
-- @authors        AtlanModTeam
-- @date           April 2006
-- @extends
-- @description    XML Metamodel
-- @see

package XML {
    -- @begin XML_Metamodel
    abstract class Node {
        reference parent [0-1] : Element oppositeOf children ;
    }
    class Element extends Node  {
        reference children [*] container : Node oppositeOf
    parent ;
    }
    class Attribute extends Node {
    }
    class Text extends Node {
        attribute content : String ;
    }
    class Root extends Element {
    }
    -- @end XML_Metamodel
}
```

Partial Correspondances







1. How to represent the partial correspondances? (M c2 an extensible MM)
2. How to compute the partial correspondances?
3. How to use the partial correspondances?

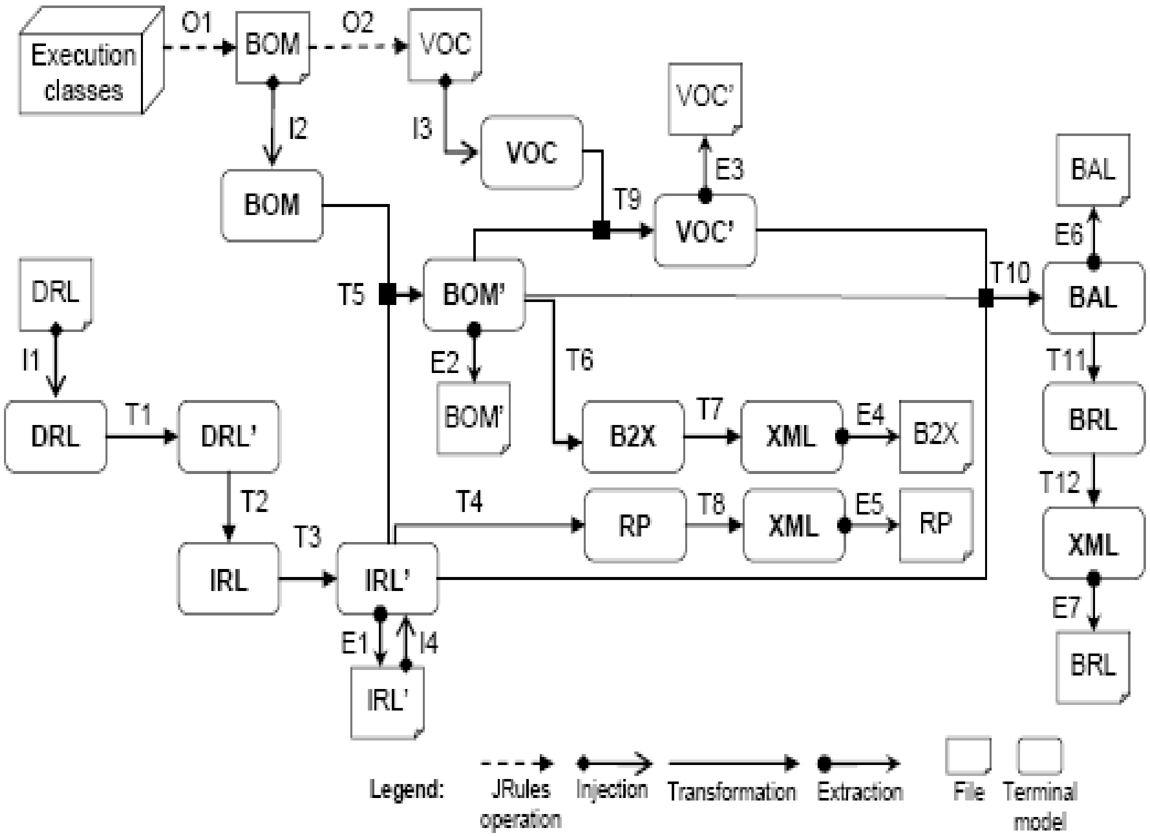
Classification of HOTS

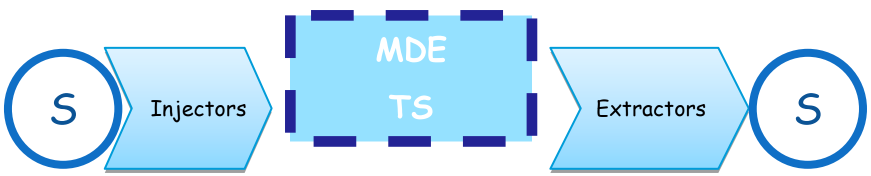
Name	Language	Source MM	Target MM	Type	Ref.
AMWtoATL_KM32SQL	ATL	AMW	ATL	Implementation	[9]
AMWtoATL_MantisBug	ATL	AMW	ATL	Implementation	[17]
AMWtoATL	ATL	AMW	ATL	Implementation	[1]
AMWtoXSLT	ATL	AMW	XSLT	Implementation	[1]
ATL2BindingDebugger	ATL	ATL	ATL	Weaving	[3]
ATL2Tracer	ATL	ATL	ATL	Weaving	[23]
ATL2WTracer	ATL	ATL	ATL	Weaving	[2]
ATL2Problem	ATL	ATL	Problem	Analysis	[4]
KM32CONFATL	ATL	KM3	ATL	Generic	[20]
KM32ATLCopier	ATL	KM3	ATL	Generic	[7]
AMWtoATL_Kelly	ATL	AMW	ATL	Implementation	[16]
MMD2ATL	ATL	KM3	ATL	Implementation	[15]
MMTtoMT	ATL	ATL, Ecore	ATL	Execution	[14]
MSDSL2EMF	ATL	KM3	ATL	Generic	[11]
Superimpose	ATL	ATL, ATL	ATL	Composition	[36]
ATLCopy	ATL	ATL, ATL	ATL	Composition	[36]
HITransform	MOFScript	MOFScript	MOFScript	Variants	[32]
Topcased	ATL	ATL	ATL	Analysis	[33]
Metamodel2Derivation	ATL	Ecore	ATL	Generic	[13]
DUALLyLeft2Right (2)	ATL	AMW, Ecore, Ecore	ATL	Implementation	[29]
Easystyle	XSLT	HTML	XSLT	Implementation	[10]
HOT4Tests	ATL	Ecore	ATL	Testing	[6]
Mutators (11)	ATL	ATL, Trace	ATL	Mutation	[6]
SingleApplication	ATL	ATL	ATL	Execution	[6]
patchgen	ATL	AMW	ATL	Implementation	[18]
propagate	ATL	ATL,INMM,INMM,AMW	ATL	Implementation	[18]
VariabilityMM_HOT	GReAT	GME, GReAT	GReAT	Analysis	[27]
MML2MMR (2)	ATL	AMW, Ecore, Ecore	ATL	Implementation	[22]
AML2ATL	ATL	AML	ATL	Extension	[19]
UITransReconfig	ATL	ATL, USRMM	ATL	Adaptation	[34]
Ecore2RDF (2)	ATL	Meo, Ecore, OWL	ATL	Implementation	[21]

On the Use of Higher-Order Model Transformations

Massimo Tisi[1], Frédéric Jouault[2], Piero Fraternali[1], Stefano Ceri[1], and Jean Bézivin[2]

Language interoperability at work



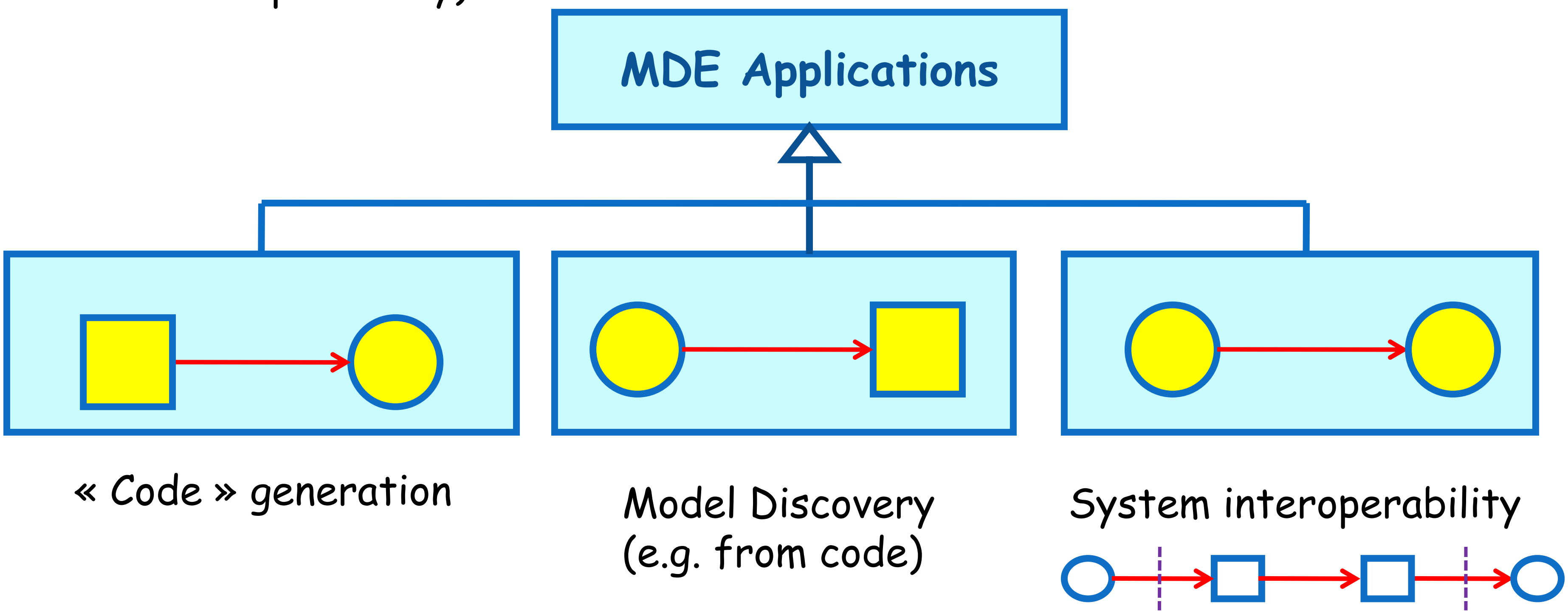


Credit to Marco Didonet del Fabro, IBM/ILOG

CONCLUSIONS

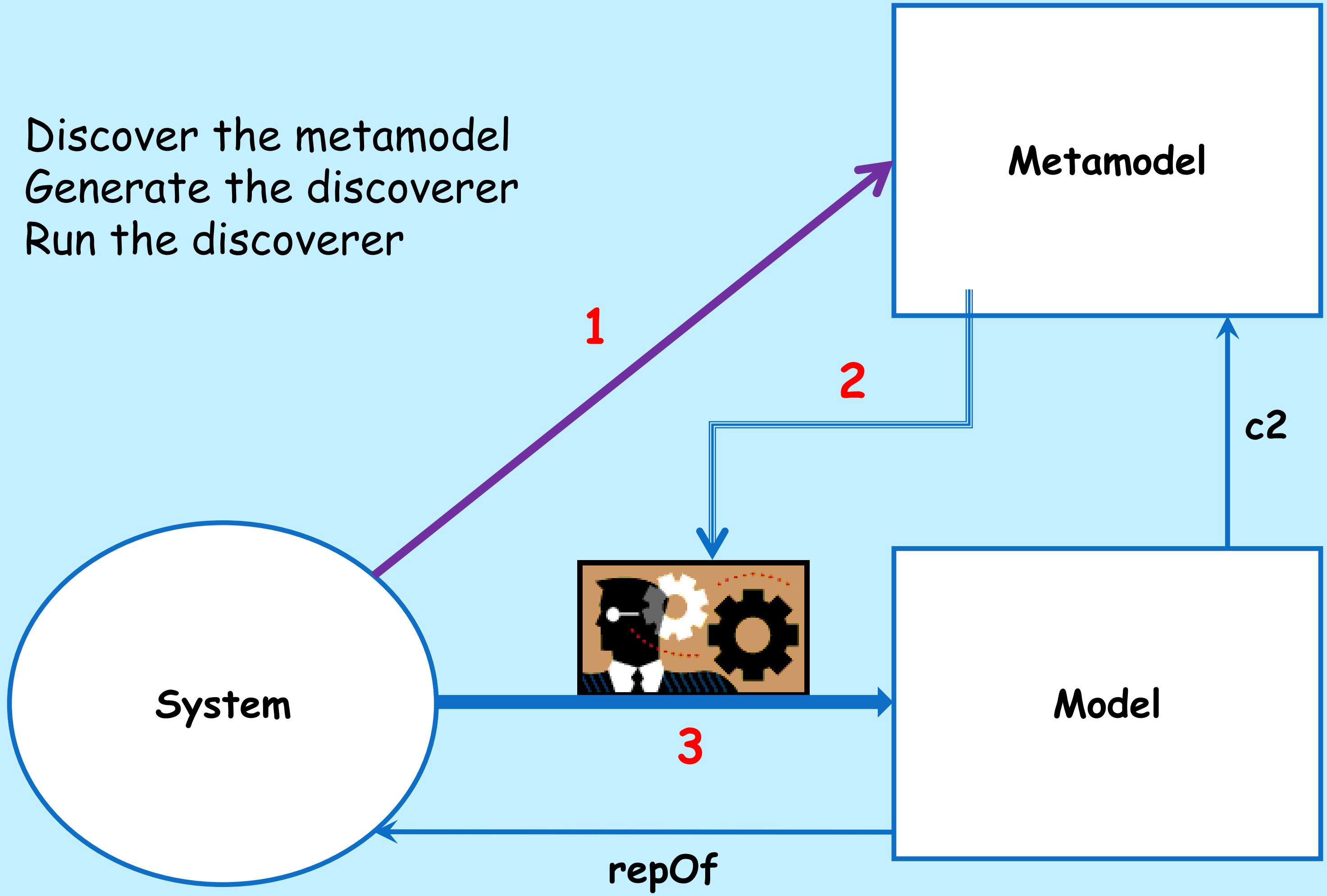
Three main types of MDE applications

- Three levels of complexity
  - ✓ **S ⇐ M** (MD Software Development for development automation)
  - ✓ **S ⇒ M** (MD Reverse Engineering for legacy modernization)
  - ✓ **S ⇔ M ⇔ M ⇔ S** (Run Time Transformation for systems interoperability)



Typical discovery process

1. Discover the metamodel
2. Generate the discoverer
3. Run the discoverer



The "Towers of Models" Grand Challenge

A more thorough science-based approach to informatics and ubiquitous computing is both necessary and possible. We often think in terms of models, whether formal or not. These models, each involving a subset of the immense range of concepts needed for ubiquitous computer systems, should form the structure of our science.

- Even more importantly, the relationships (either formal or informal) among them are the cement that will hold our towers of models together. For example, how do we derive a model for senior executives from one used by engineers in designing a platform for business processes, or by theoreticians in analyzing it?

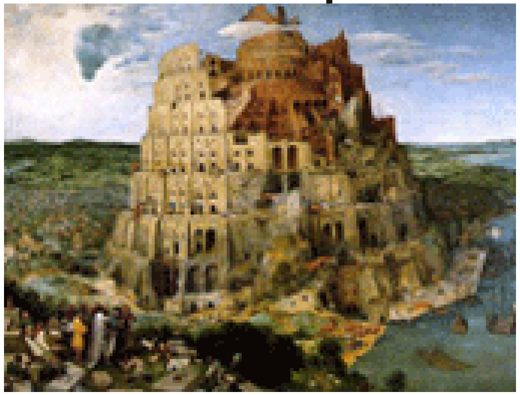
- The essence of software engineering and informatics is formulating, managing, and realizing models.

Robin Milner

The SLE conference

- SLE is building this area of tolerance and competition where engineering support for various kinds of software modeling may be experimented, compared and promoted.
- This is the place where Robin Milner vision of the « Towers of Models » may be carried up to its theoretical foundations and to its most practical applications.
- Thanks to the organizers for having made this possible.

The tower of Babel



Bruegel the Elder 1563

Thanks

- ✓ Questions?
- ✓ Comments?

http://www.emn.fr/x-info/atlanmod/

Jean Bézivin
Jean.Bezivin{noSpamAt}inria.fr
AtlanMod research team, INRIA & EMN, Nantes, France