History and Context of MDE

Jean Bézivin

JBezivin@gmail.com

About the Speaker

Jean Bézivin

- Presently emeritus professor at the University of Nantes and industrial consultant on software modeling
- Founder and Previous lead of the AtlanMod INRIA research team (model support for software creation, evolution and operation)
- Co-founder of the **ECOOP** and **TOOLS** conferences in the 80's
  - ✓ And observer of the procedural to OT paradigm shift
- Co-founder of the **MODELS** and **ICMT** conferences (~2000)
  - ✓ The most difficult thing with the first one was to convince people to change the name of the conference from <<UML>> to MODELS.
    - ❖ i.e. that "doing research on UML" is rather an oxymoron
    - ❖ the important notion is DSL but this was realized later
  - ✓ The most difficult thing with the second is still to convince that:
    - ❖ Modeling without automation is futile
    - ❖ Automation needs transformation

About the Series of Lectures

Principles and Applications of Model Driven Engineering

- **Lecture 1: History and context of MDE**
- **Lecture 2: Basic Principles of MDE**
- **Lecture 3: Classification of MDE applications**
- **Lecture 4: Where will be MDE in 2030? (the future)**

Lecture 1: History and context of MDE

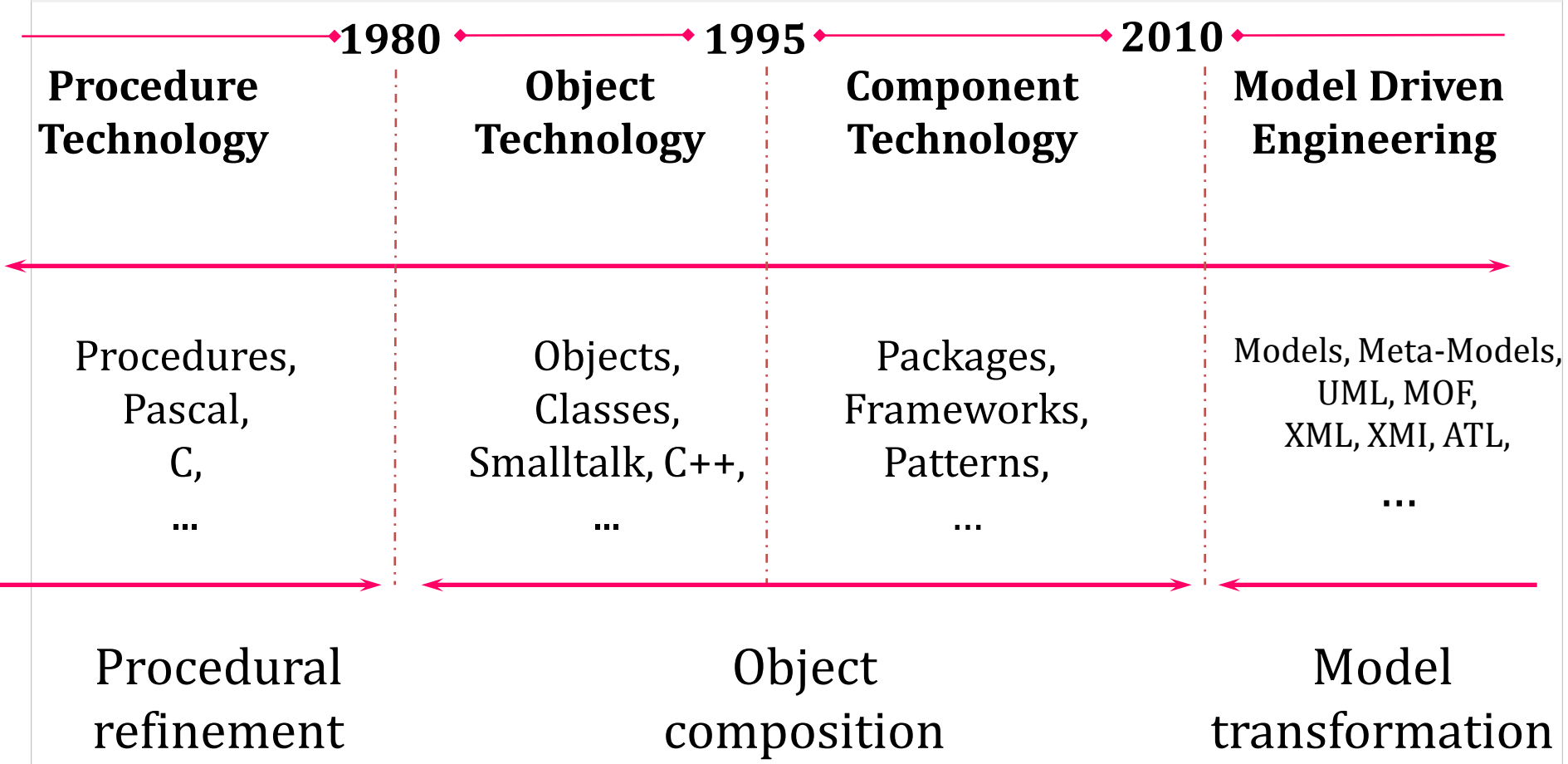
- ✓ Context
- ✓ History
- ✓ Basic Notions

- ❑ Where are we heading? (Lecture #4)
- ❑ Where do we stand now? (Lecture #2, #3)
- ❑ From where are we coming? (Lecture #1)

Model Driven Engineering

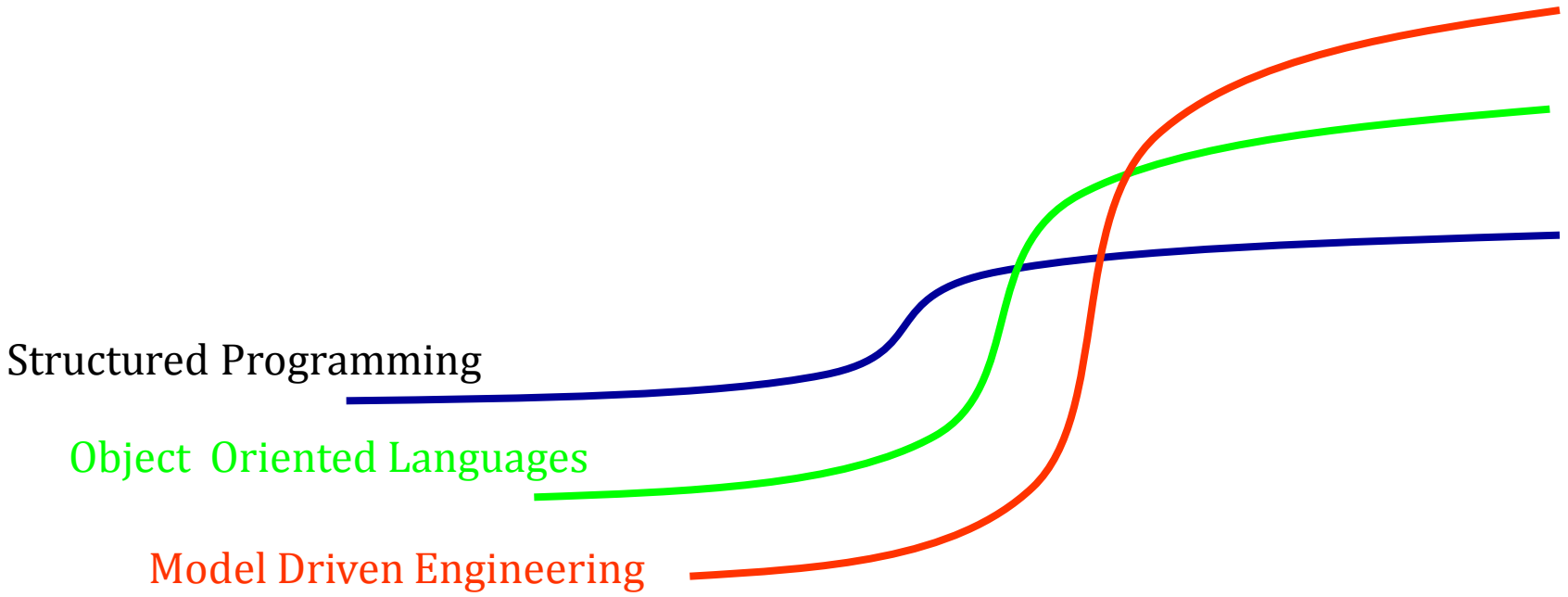
- Basic Notions
  - This lecture will introduce most MDE concepts used in the remaining lectures.
    - ✓ The important notions of Technical Space and DSL will be discussed later
  - These will be first presented intuitively and informally
  - They will be revisited later in a more precise way

MDE is about paradigm change



It's about an important paradigm change: unification by objects -> by models

Technology waves

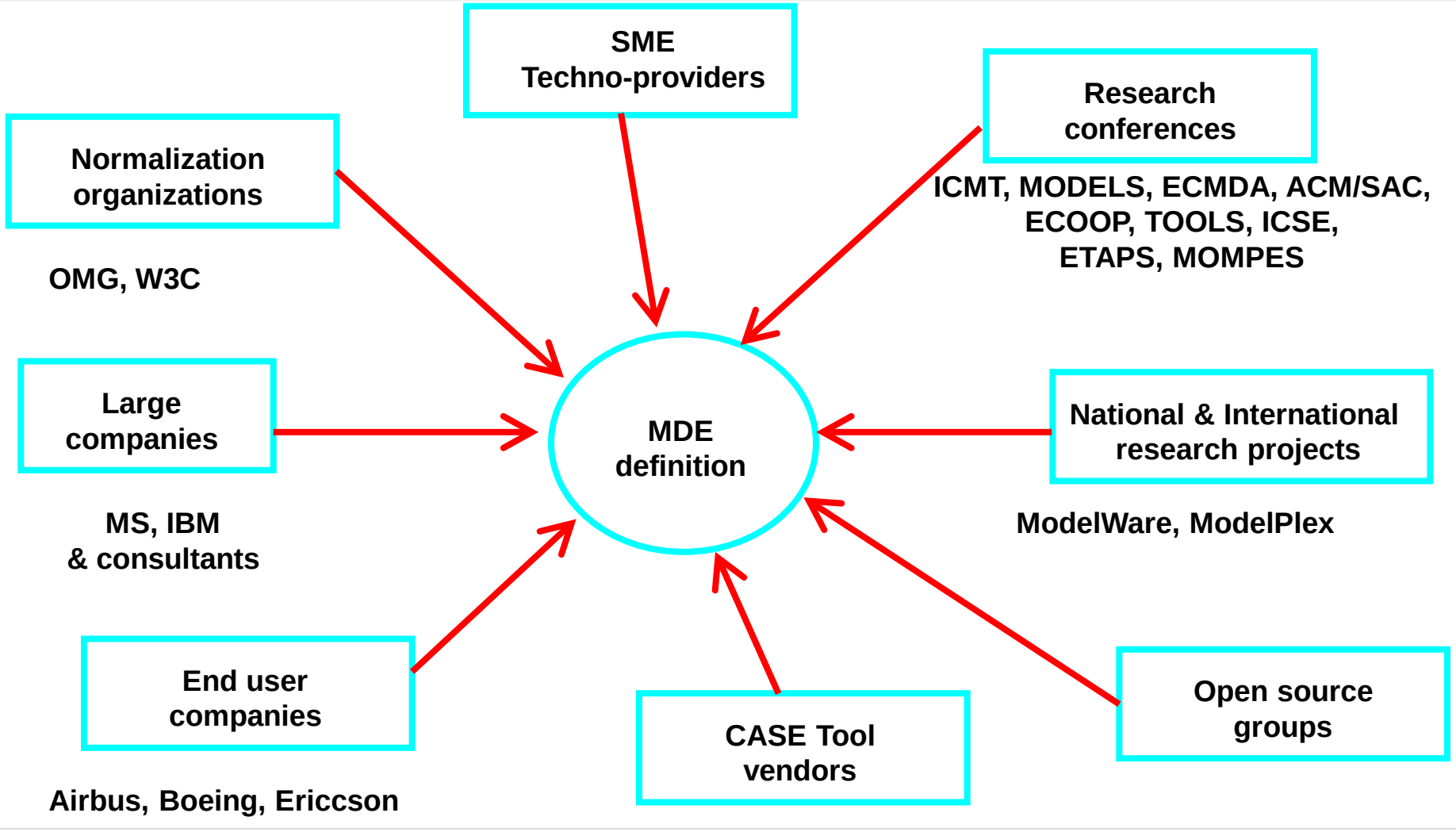


Each wave does not replace the previous one, but complements it.

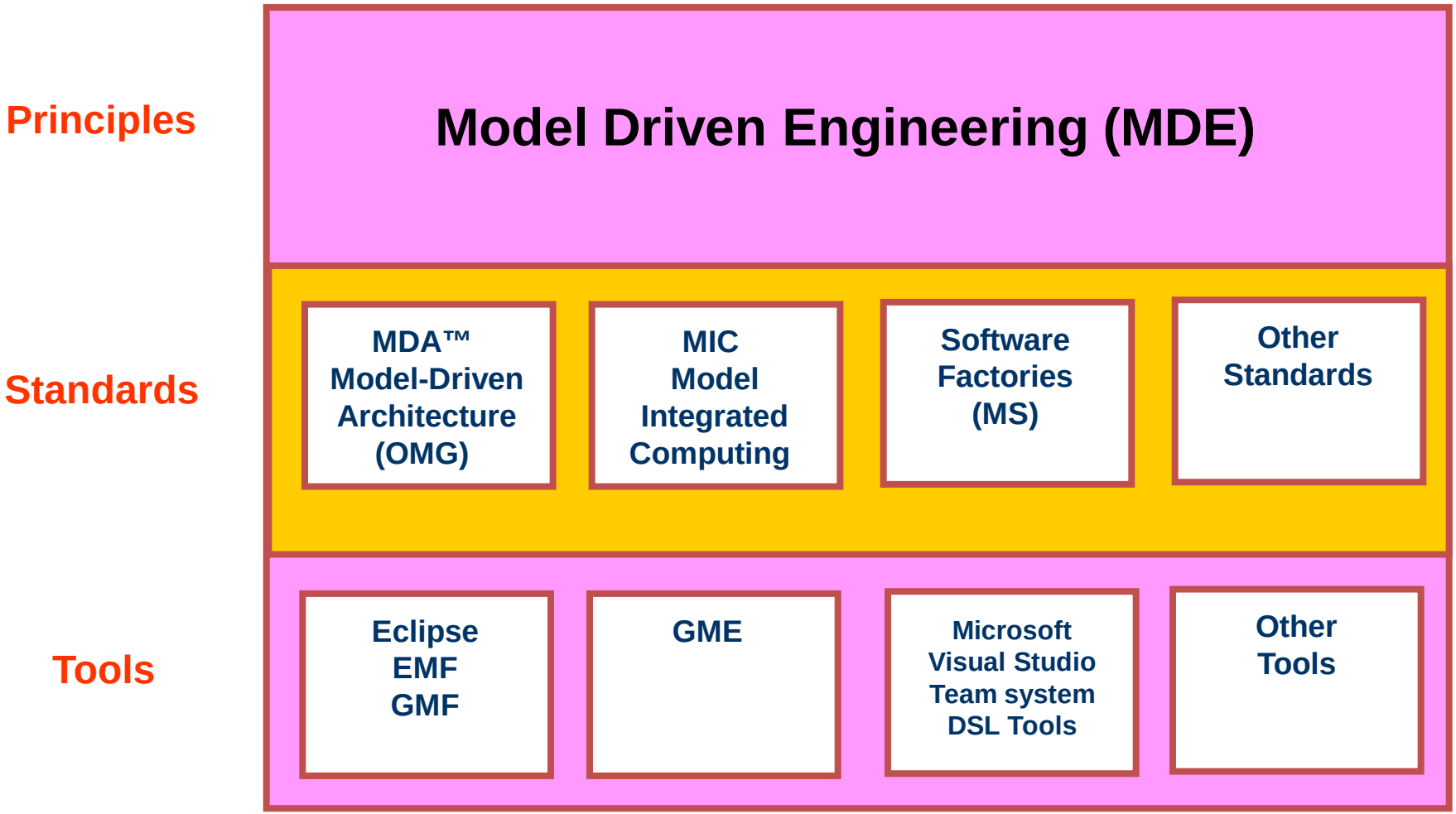
Multiple acronyms

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

Multiple contributors

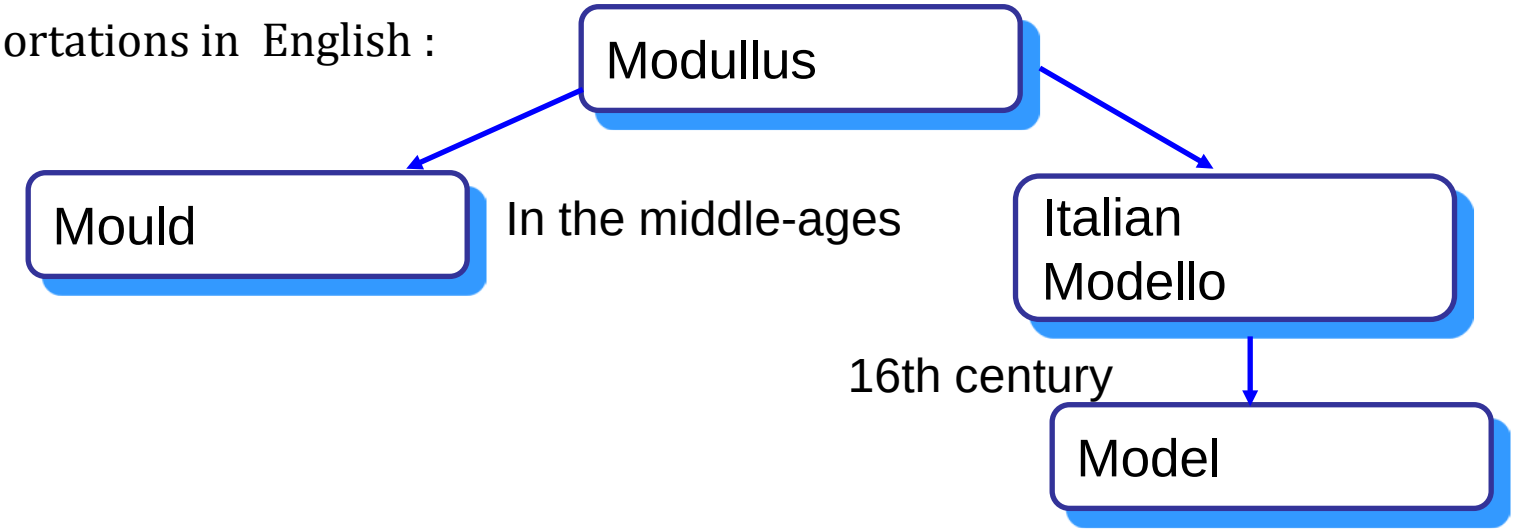


Multiple Levels



Models

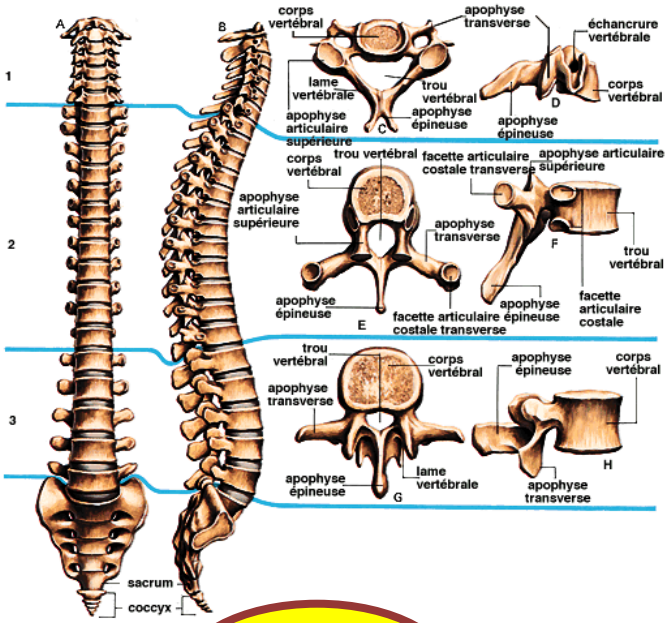
- A model is the simplified image of a system
  - This short definition should be completed
- What is a system ?
  - "*A system is a set of elements in interaction*" (von Bertalanffy)
  - The word system comes from the greek "*sun-istémi*" (I compose)
- Model comes from the latin "*modullus*", diminutive of "modus" (measure)
  - Initially this term was used in architecture to designate the dimension ratio between two elements of a building in construction.
- Two importations in English :

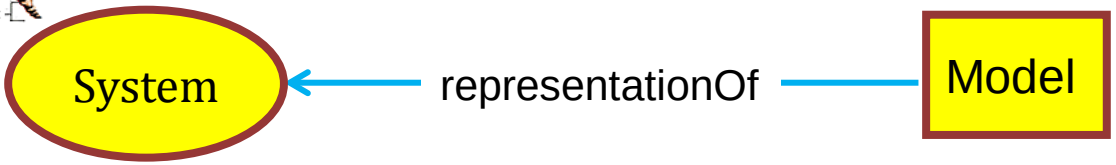


The word is recent, the idea is old

Plato (427-347 before JC), in *Timeus* compares vertebras
to door hinges 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.





Don't confuse the model and the system

This is not a pipe by Magritte

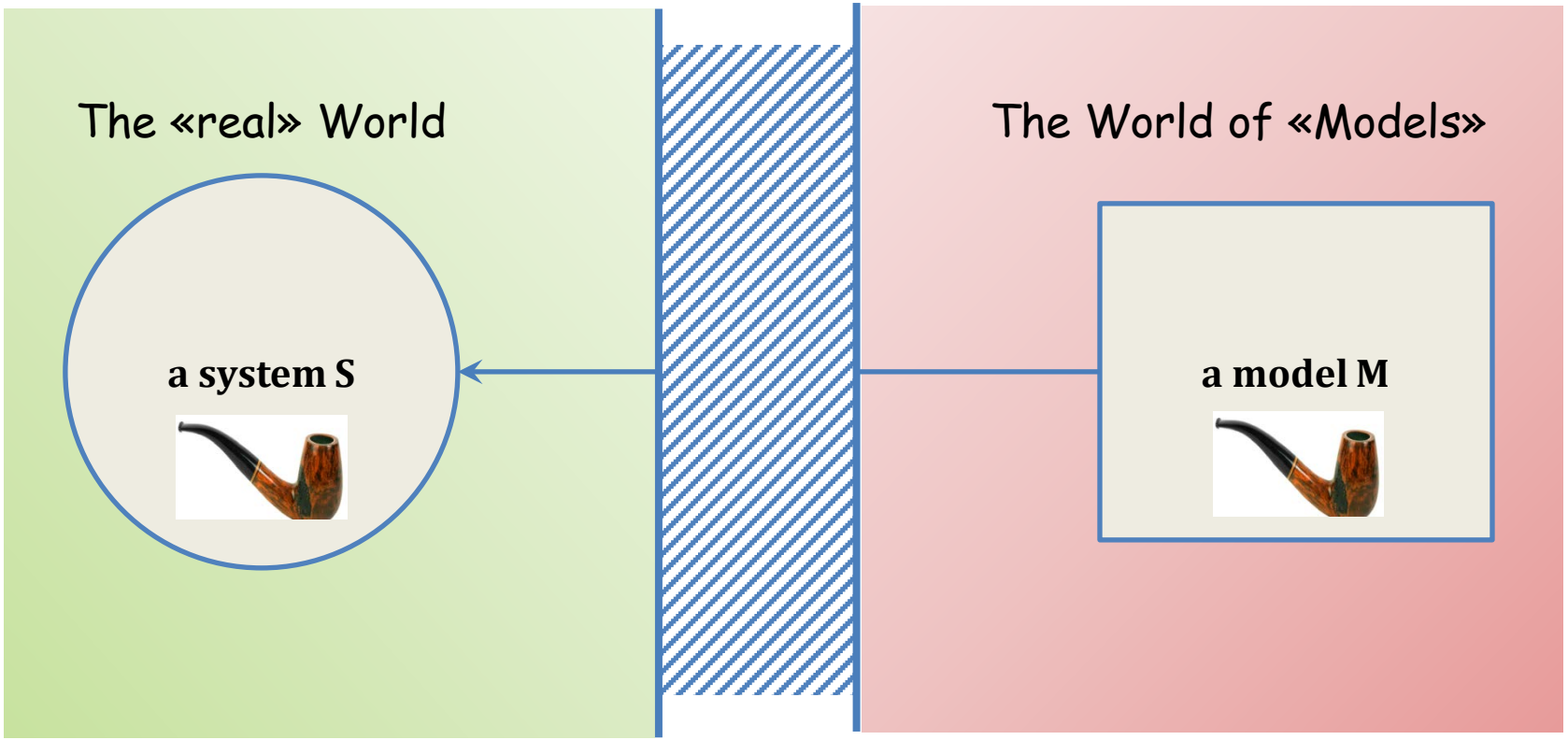
This is just a model of a pipe;
you can't use it to smoke

This is not a meal

This is just a model of a meal,
showing price and appearance;
but you can't eat it

Systems and Models

Squares and Circles



Legends and explanations

Model element

Metamodel element

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

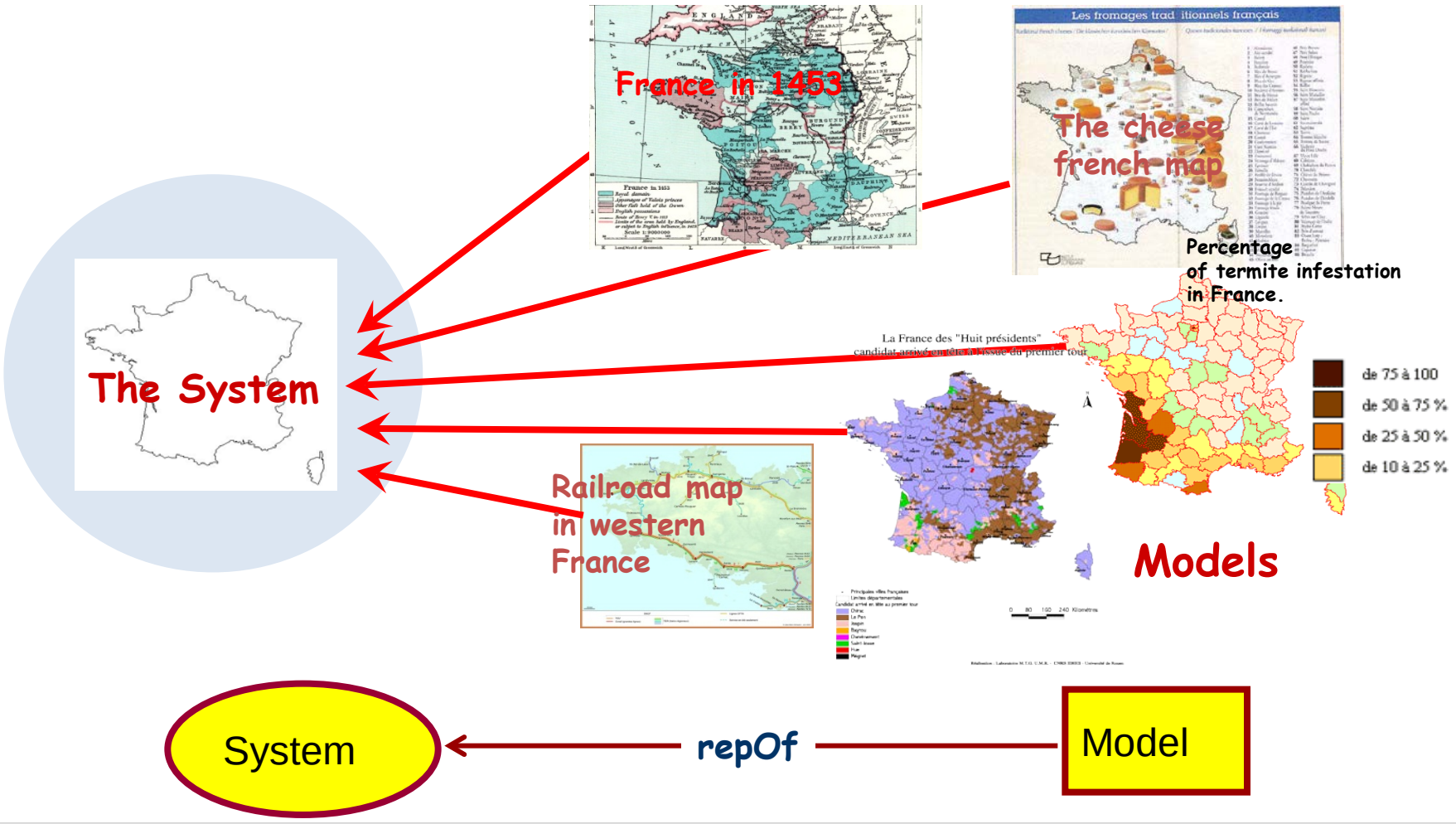
Metamodel

c2

Model

The legend
is the metamodel

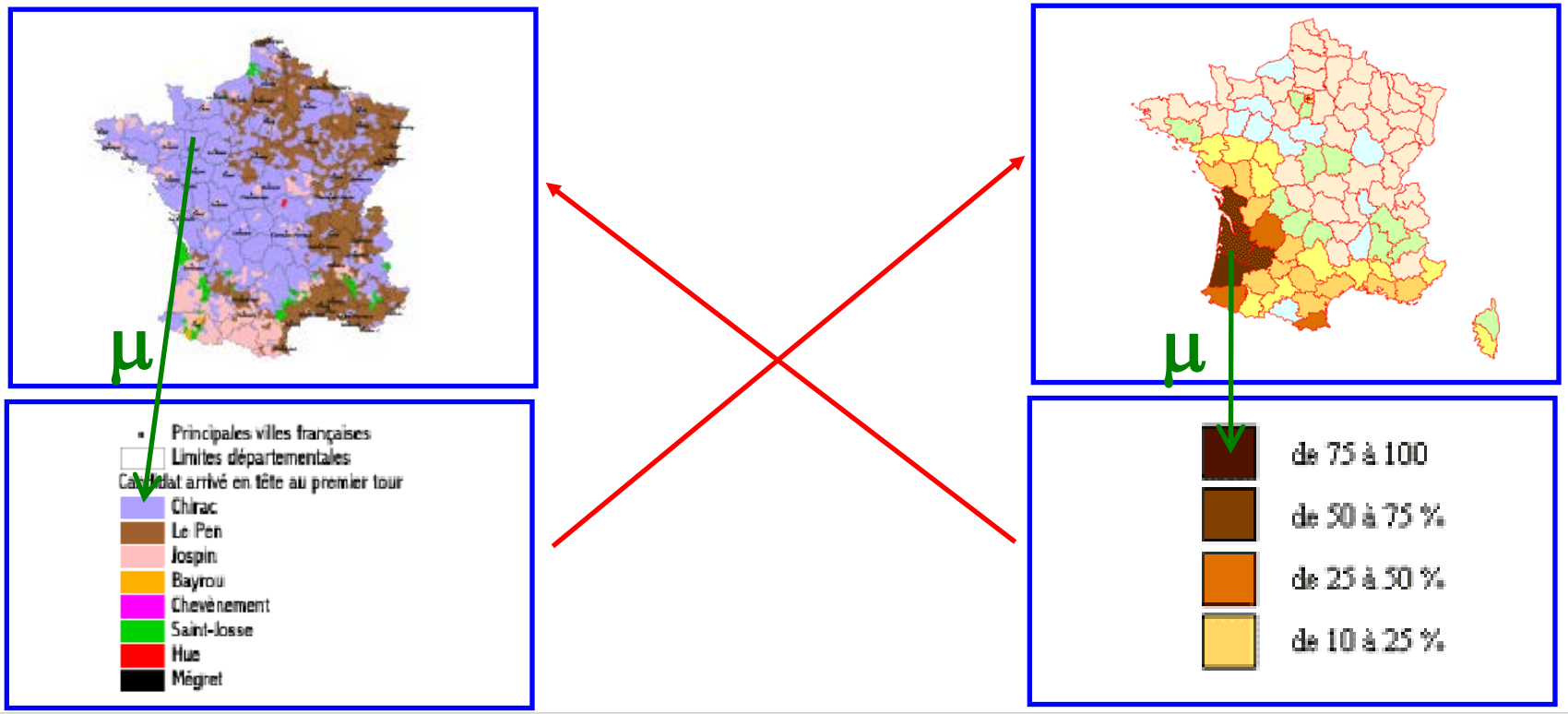
Multimodeling



Map without legend is meaningless

First round of political election in France in 2002.

Percentage of places infested by termites in France.



Another Notation (DSL)

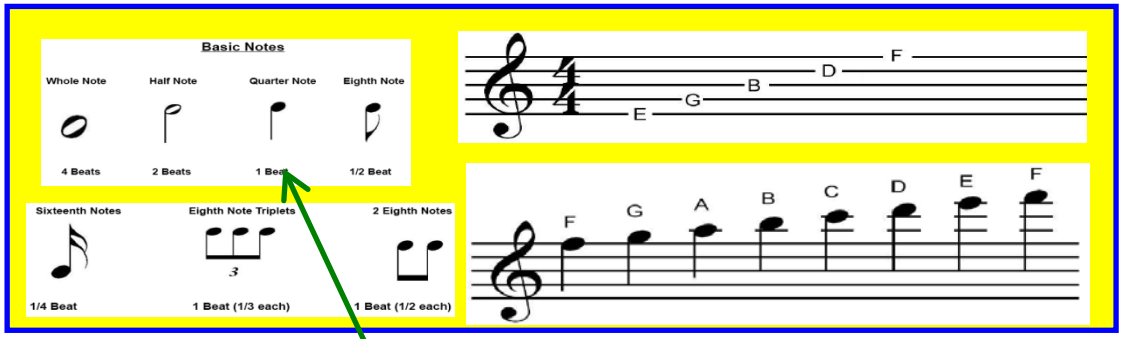
Music notation

Metamodel

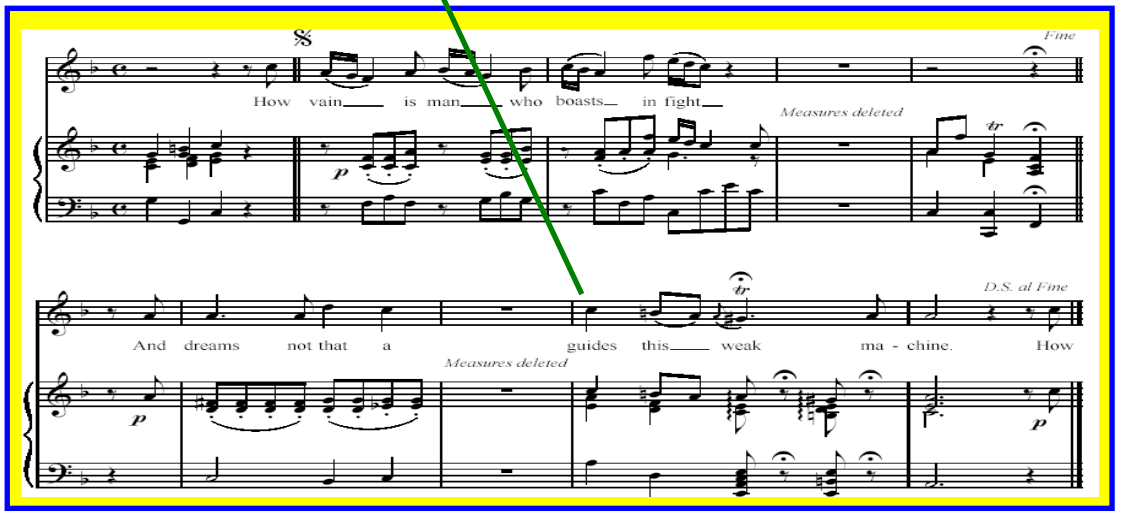
c2

Model

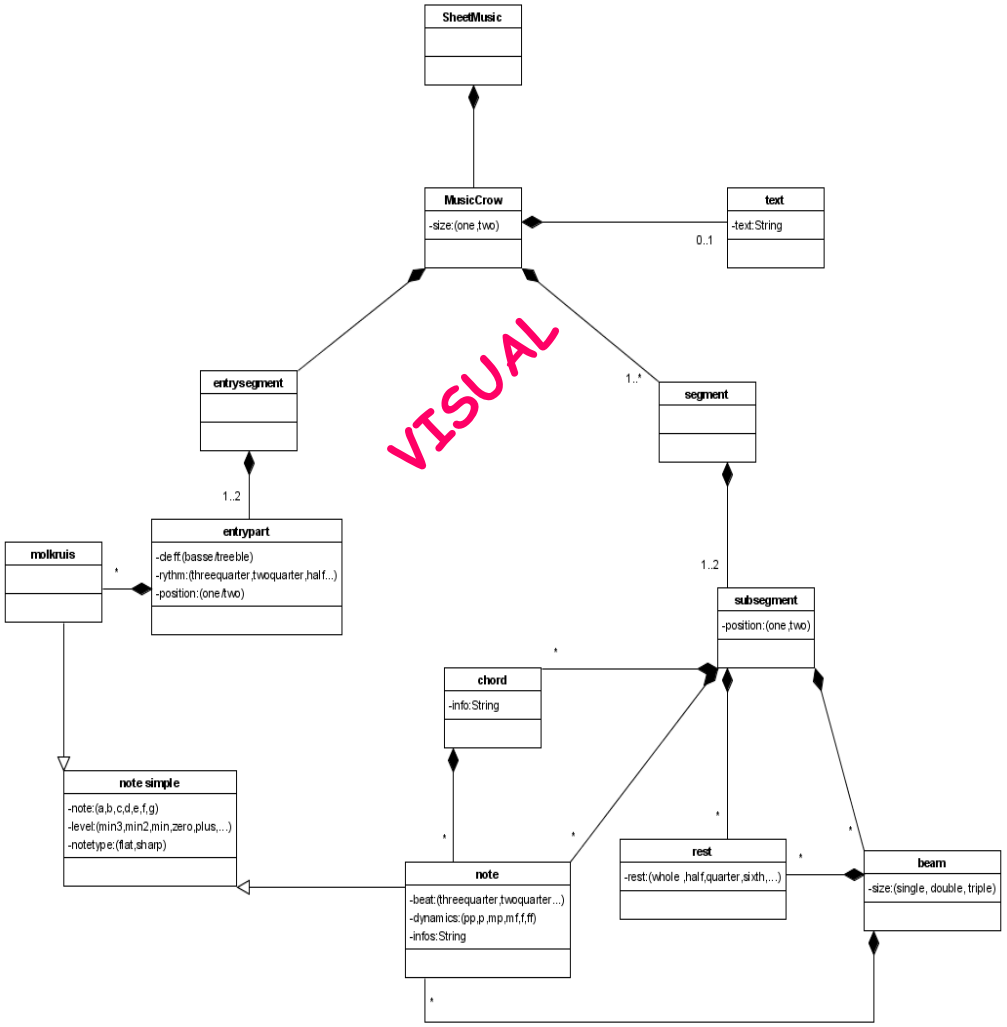
Music sheet

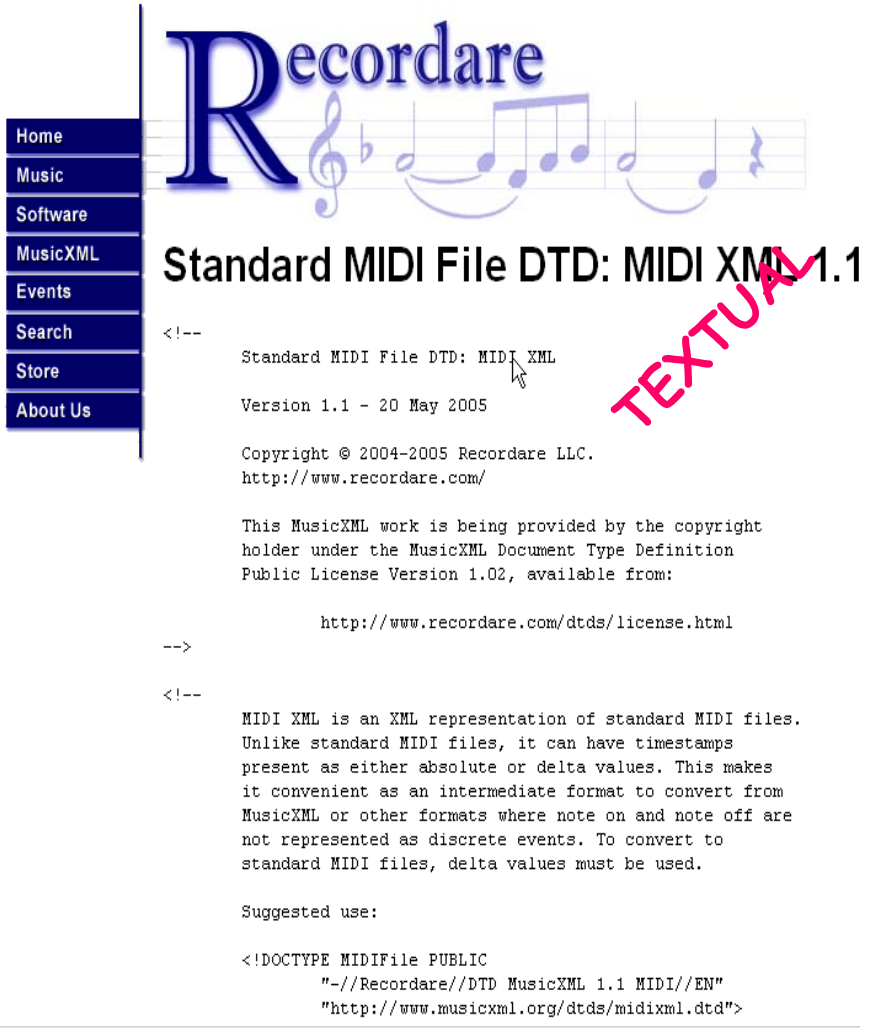


μ

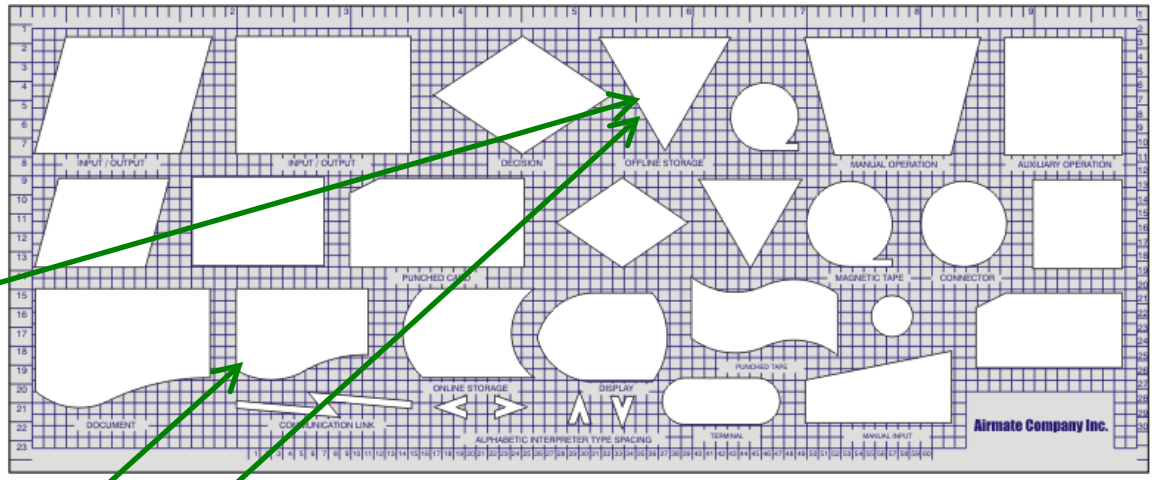


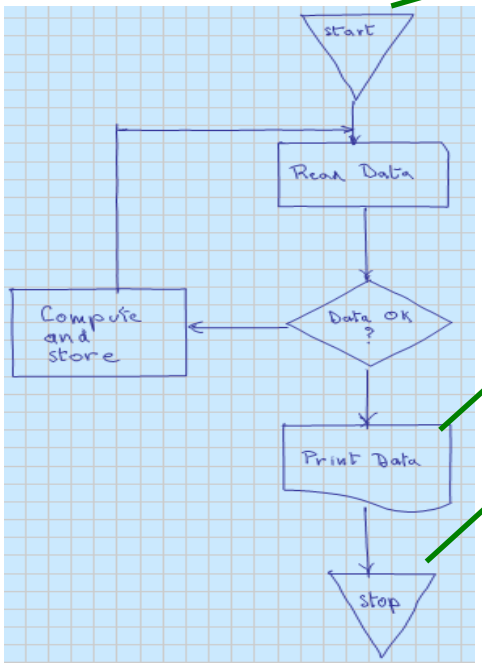
MusicXML: several concrete syntaxes





Flowcharting DSL in the 60's





μ

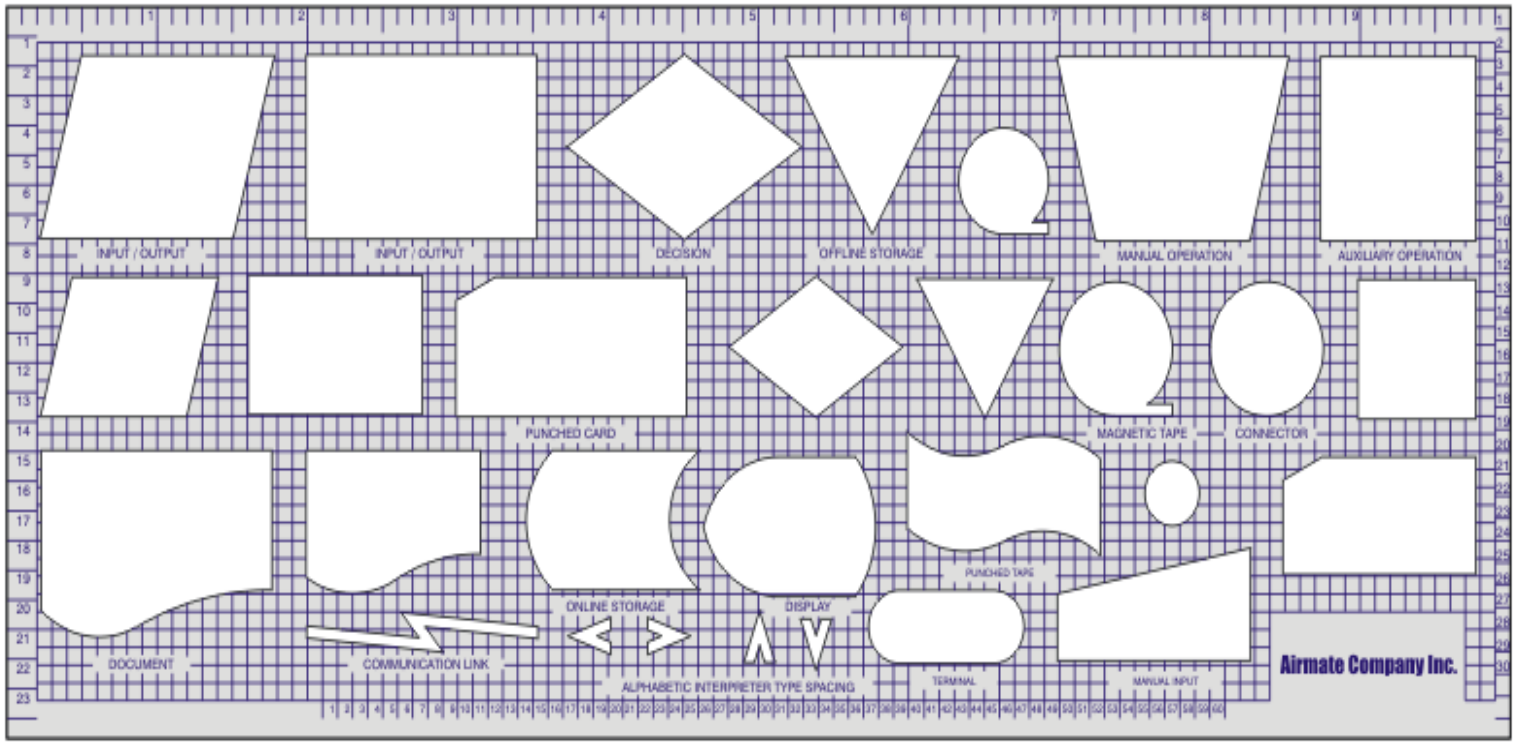
Metamodel

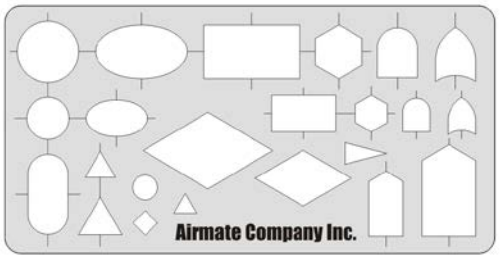
c2

Model

A metamodel is an explicit specification of a shared conceptualization

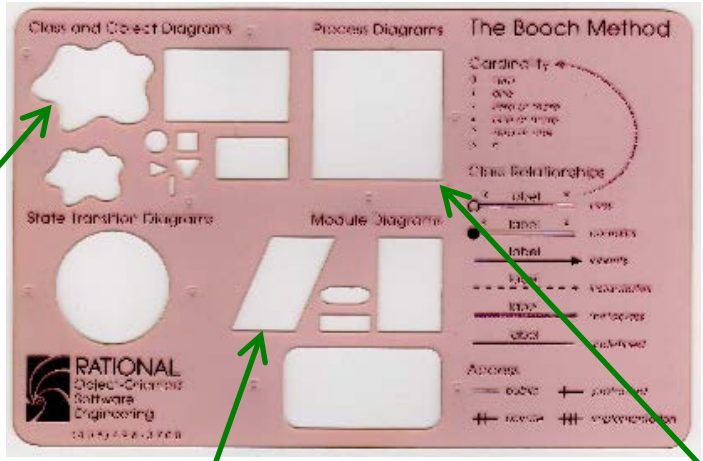
A set of concepts and relations between them





Many templates for several corporations and usages

The Rational/Booch Template

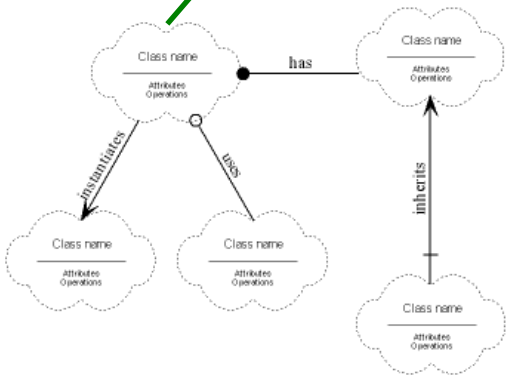


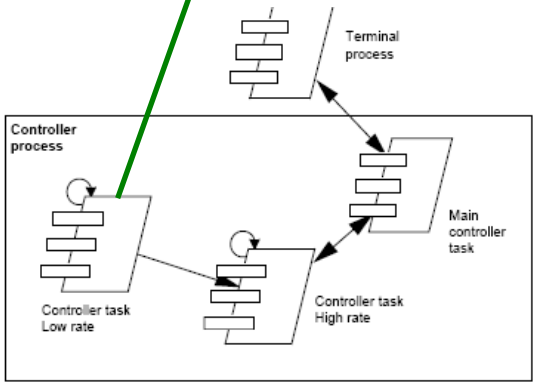
Concrete syntaxes (visual)

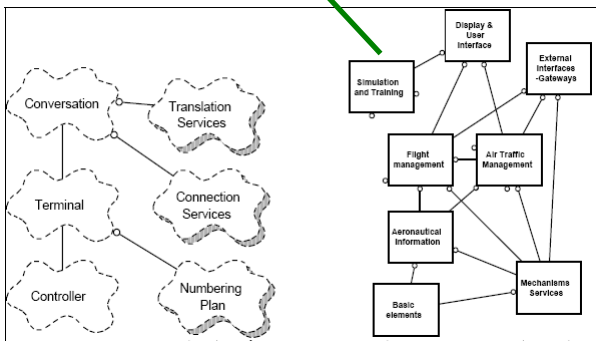
μ

μ

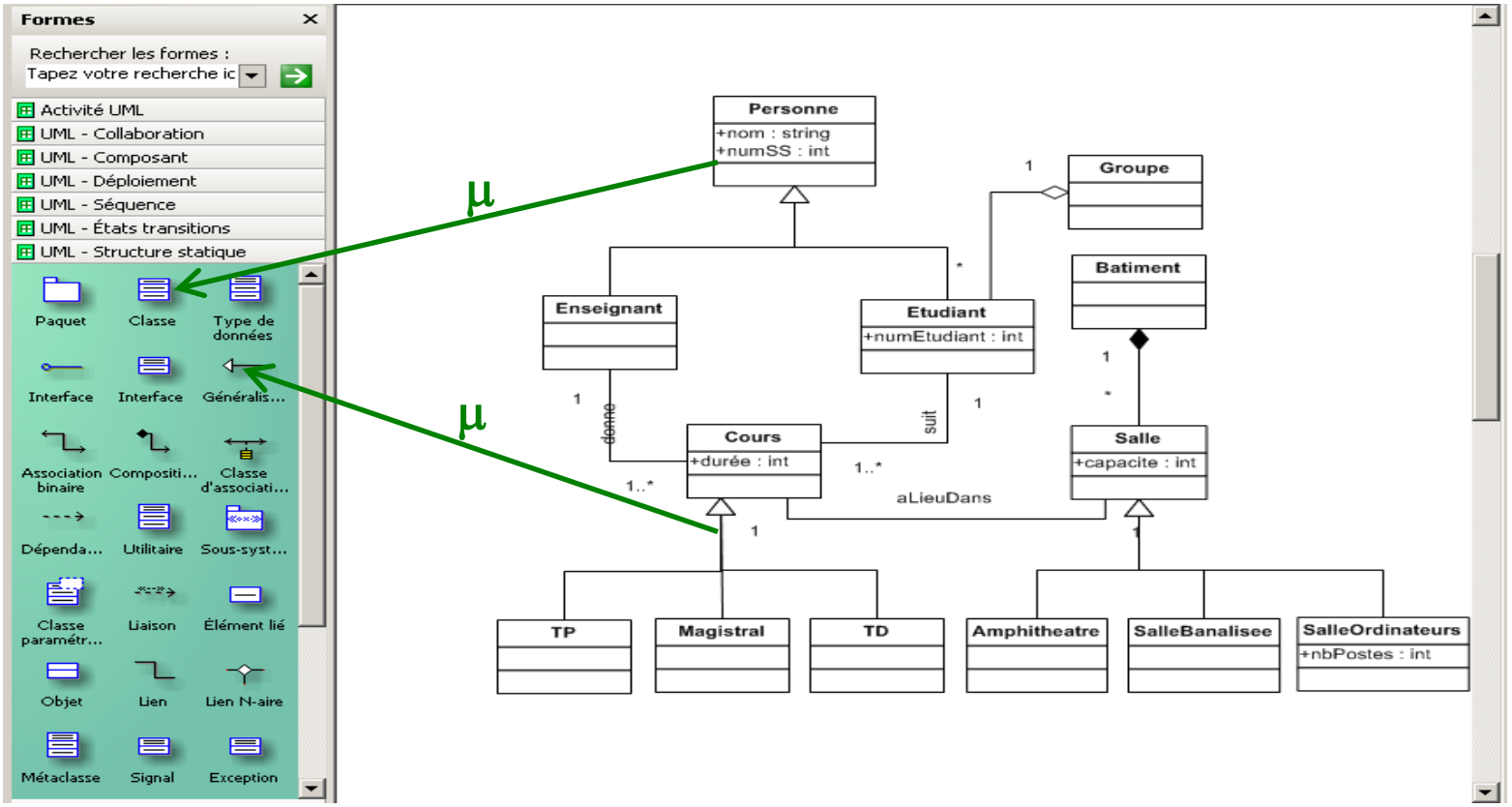
μ

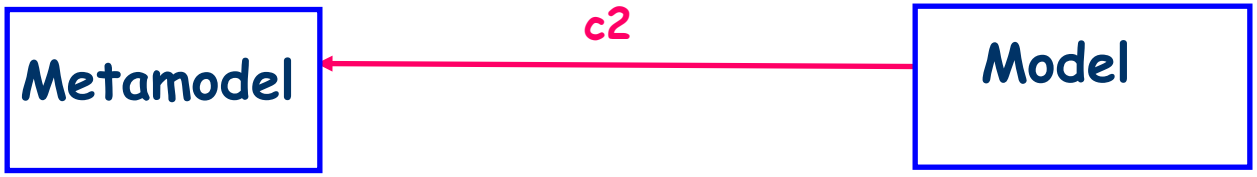




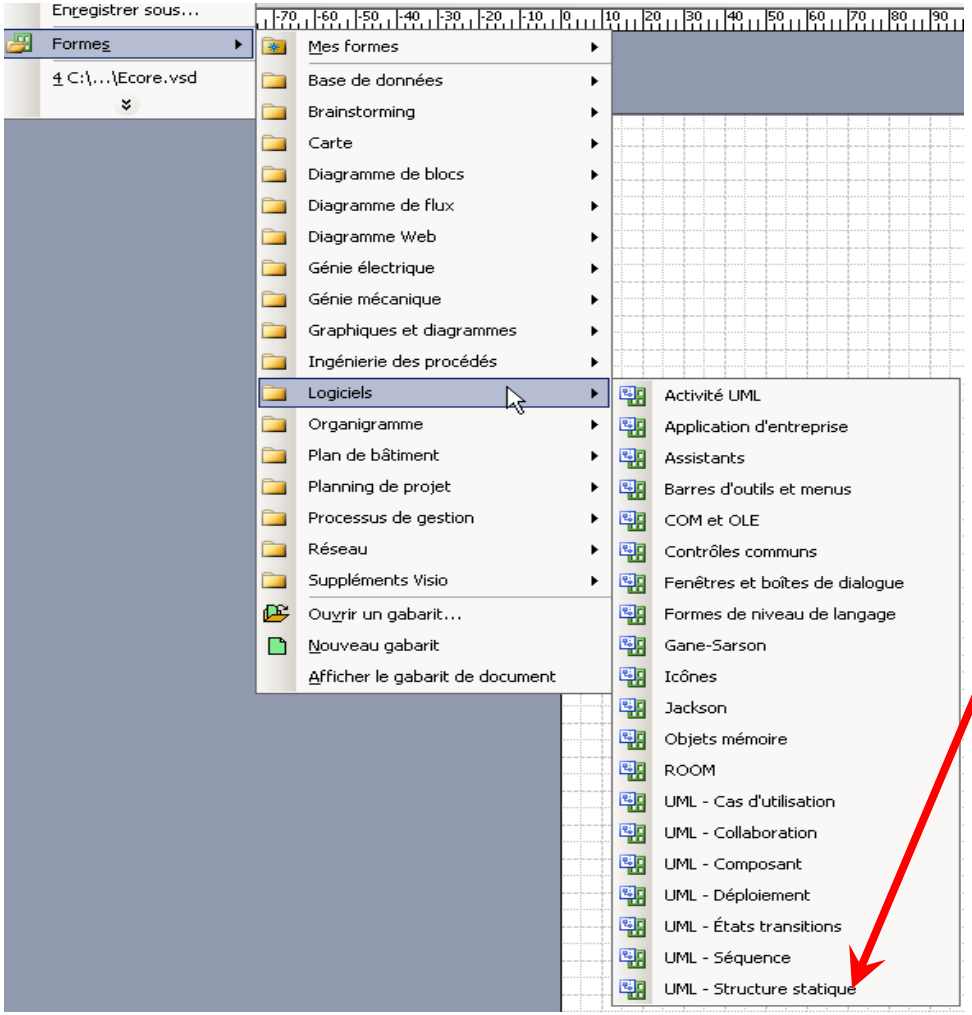


Towards Metamodel-based tools

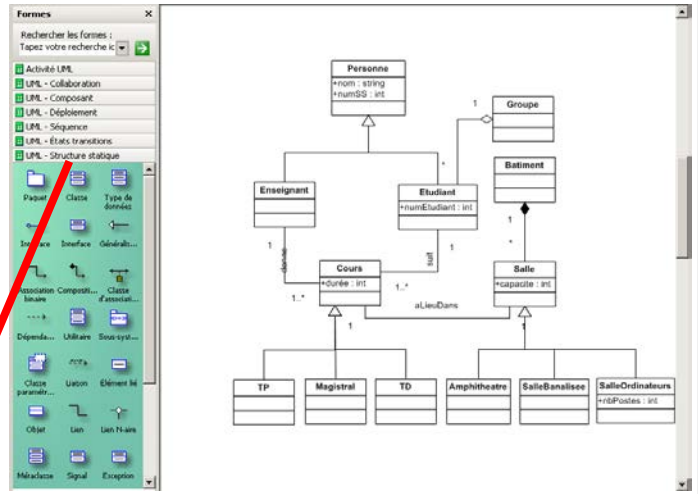




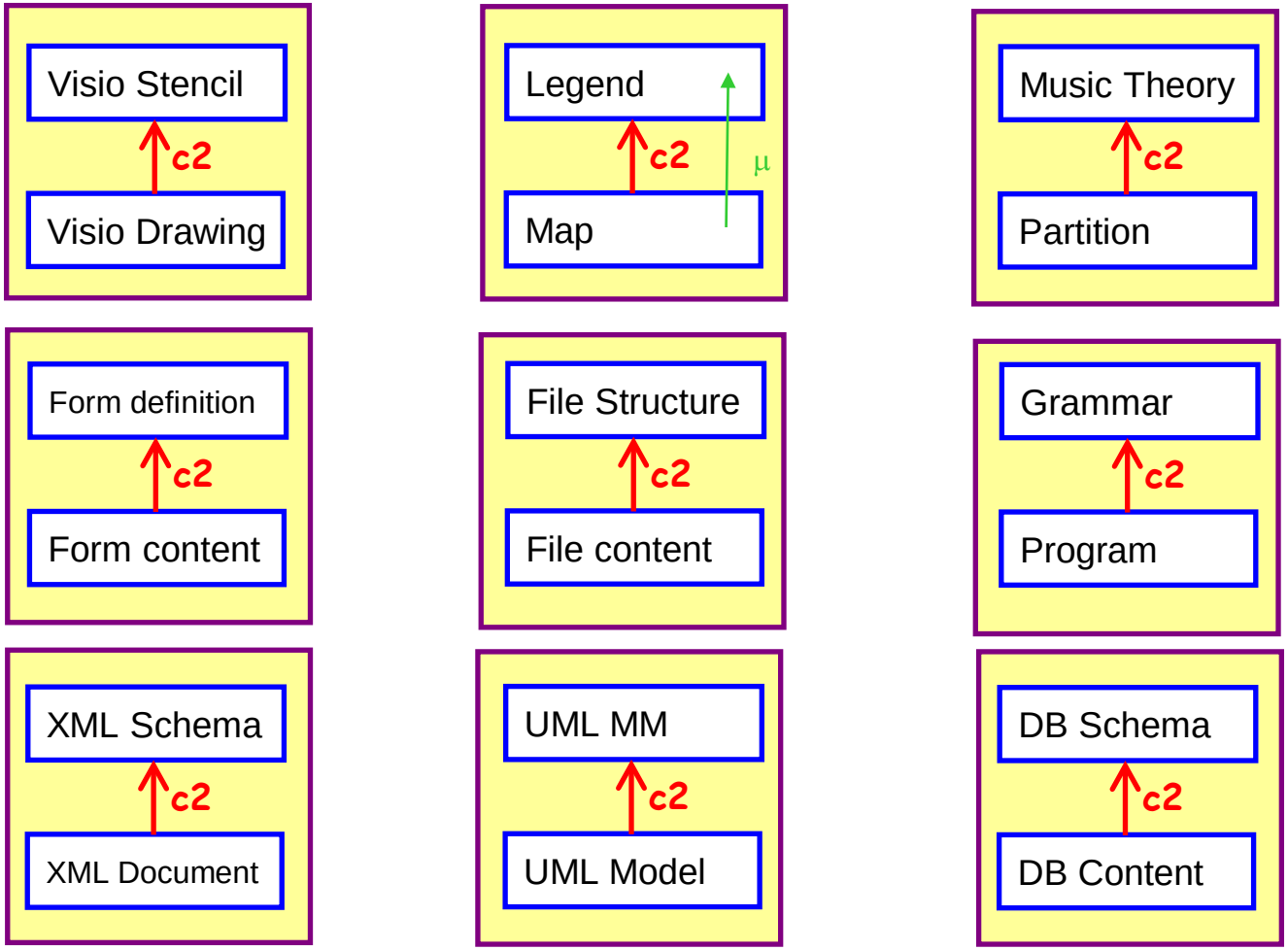
A library of stencils. A Metamodel agnostic tool.



μ



Some typical situations



Some software modeling formalisms

- Periods
  - Flowcharts (~1950)
  - Petri Nets (~1960-1970)
  - PSL/PSA (~1967)
  - State Diagrams (~1967)
  - SADT (~1969)
  - DFD (~1975)
  - Entity-Association (Chen, ~1976)
  - JSD (~1982)
  - AD/Cycle (~1982)
  - UML (~1996)
  - MDA (~2000)
- Note: The relation between formal methods and modeling has not been simple

Early modeling formalisms

- Modeling formalisms have a long and rich history
  - They appeared rapidly after the first programming languages and often parallel the history of programming languages (flow charts were used with assembly language)
- Some of the bright ideas learnt in using these early modeling formalism found their way into UML
- But the vast majority of them just got lost
  - There is a need to revisit these initial efforts in software modeling and to recover the good ideas, beyond the UML standardization/simplification/reduction
- The mentiond list is just a small subset of these initiatives

Flowcharts

- A **flowchart** is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows.
- Flowcharts are used
  - To understand already written programs (descriptive)
  - To guide the writing of new programs (prescriptive)
- Flowcharts **reveal** the control structure (sequencing of operations) of a program and **hides out** the flow of data (separately represented by data flow diagrams)
- John von Neumann developed the flow chart (originally, diagram) to plan computer programs. ("Planning and coding of problems for an electronic computing instrument, Part II, Volume 1" (1947), reproduced in von Neumann's collected works)
- Robert W. Floyd used flowcharts to introduce his seminal paper "Assigning Meaning to Programs"

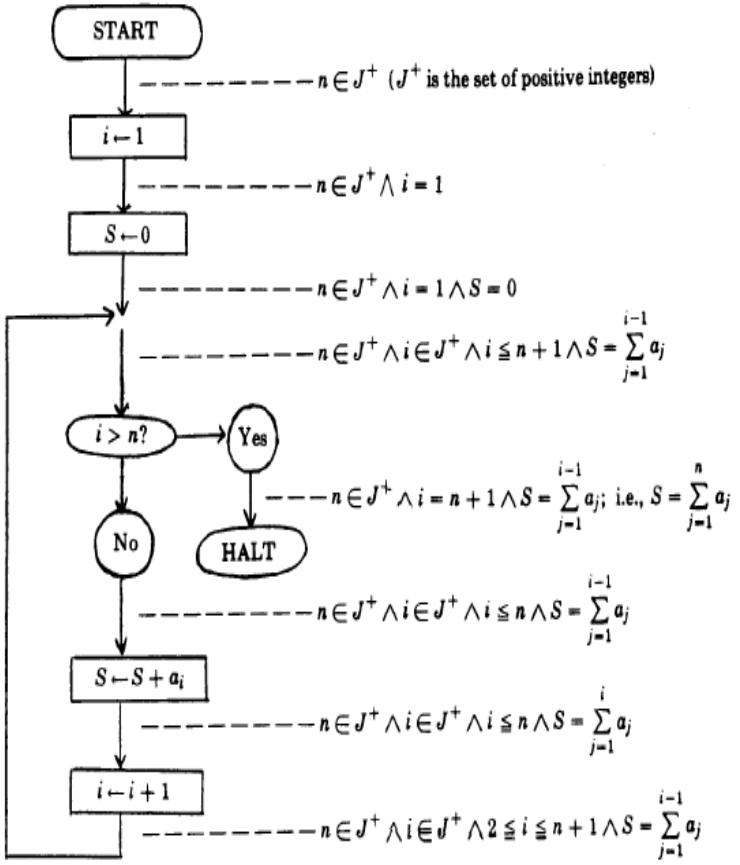
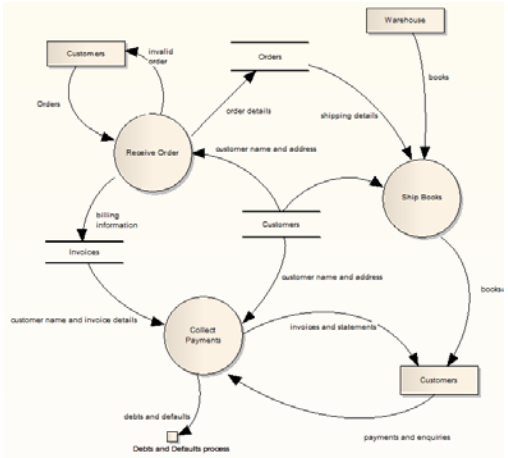
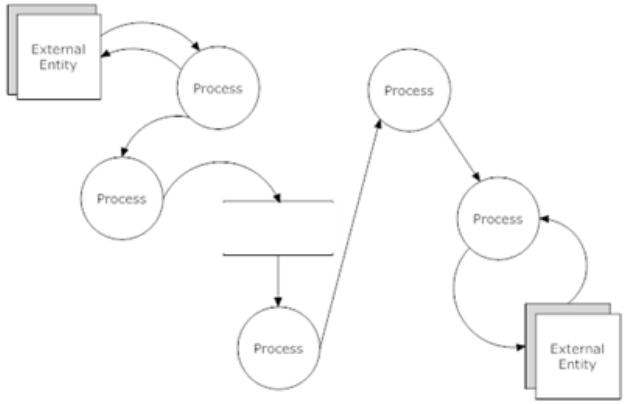


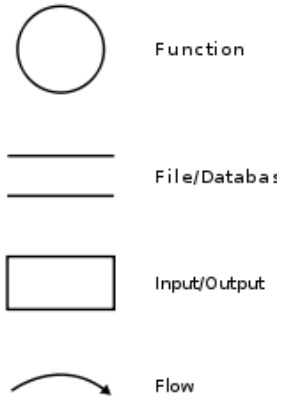
FIGURE 1. Flowchart of program to compute $S = \sum_{j=1}^{n} a_j$ $(n \geqq 0)$

DFD (Data Flow Diagrams)

- A data flow diagram (DFD) or “bubble chart” is a graphical representation of the "flow" of data through an information system (data flow v. workflow). A DFD shows where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel like in a flowchart.
- Data flow diagrams were proposed in [Larry Constantine and Ed Yourdon (1975) *Structured Design*. Yourdon Press].





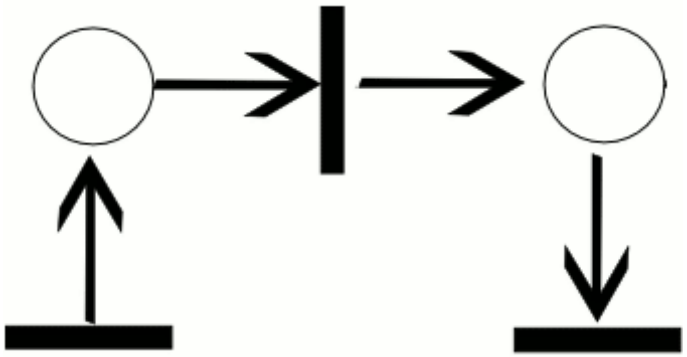


Petri Nets

- Petri nets originally developed in the early 70s for description and analysis of synchronization, communication and resource sharing between concurrent processes in distributed systems.
- Directed bipartite graph, in which the nodes represent **transitions** (i.e. events that may occur, signified by bars) and **places** (i.e. conditions, signified by circles). **Arcs** run from a place to a transition or vice versa.
- Places may contain marks called **tokens**. Any distribution of tokens over the places will represent a configuration of the net called a *marking*. A transition of a Petri net may *fire* whenever there are sufficient tokens at the start of all input arcs; when it fires, it consumes these tokens, and places tokens at the end of all output arcs. Execution of Petri nets is nondeterministic.

On the contrary of most other formalisms, we need an animated gif to describe a Petri Net.

=> Dynamic model.

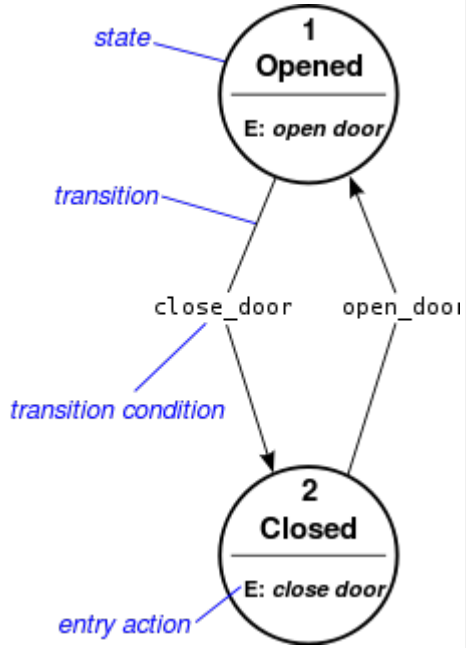


PSL/PSA (Modeling Systems for more than 40 years)

- Problem/Statement Language/Problem Statement Analyzer (from http://www.pslpsa.com)
  - Modeling Systems for more than 40 years
  - PSL/PSA has a history going back to late 1967/early 1968. It has a long and distinguished applied research foundation, probably the most extensive research project ever, in the modeling of requirements.
  - PSL/PSA is probably the world's oldest requirement modeling language.
  - PSL/PSA is probably also one of the first non-executable modeling language.

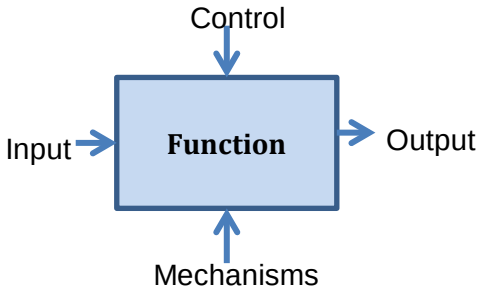
State Diagrams

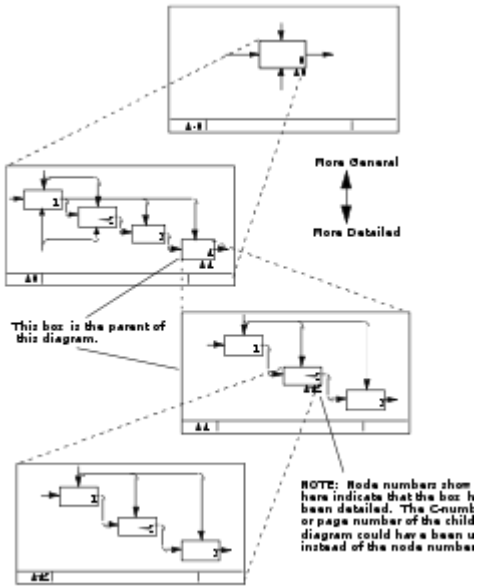
- A state diagram is a type of diagram describing the behavior of systems composed of a number of states.
- Many variants of state diagrams exist, with different semantics and different expression capabilities. Sometimes it is possible to convert from one form to another one.
- State diagrams can be used to graphically represent finite state machines as introduced in 1967 by the book *Sequential Machines and Automata Theory* (Taylor Booth).
- David Harel statecharts have gained widespread usage since a variant has become part of UML.



SADT

- Structured Analysis and Design Technique (SADT) is a software engineering methodology for describing systems as a hierarchy of functions developed and field-tested during the period of 1969 to 1973.
- SADT is a diagrammatic notation designed specifically to help people describe, understand and develop systems. It offers building blocks to represent entities and activities, and a variety of arrows to relate boxes.
- SADT can be used as a functional analysis tool of a given process, using **successive levels of details**.

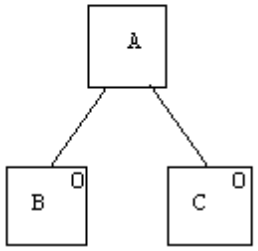




JSD (Jackson System Development)

- JSD was first presented by Mikael Jackson in 1982, in a paper called "A System Development Method" and in 1983 in the *System Development* book, with several innovative ideas, including:
  - **Development must start with describing and modeling the real world**, rather than specifying or structuring the function performed by the system. A system made using JSD method performs the simulation of the real world before any direct attention is paid to function or purpose of the system.
  - The way of implementing the system is based on **transformation of specification into efficient set of processes**. These processes should be designed in such a manner that it would be possible to run them on available software and hardware.
- As originally presented, the method consisted of six steps, each step using a given formalism
  - Entity/Action Step
  - Initial Model Step
  - Interactive Function Step
  - Information Function Step
  - System Timing Step
  - System Implementation Step

Jackson structure diagram



Jackson structure text

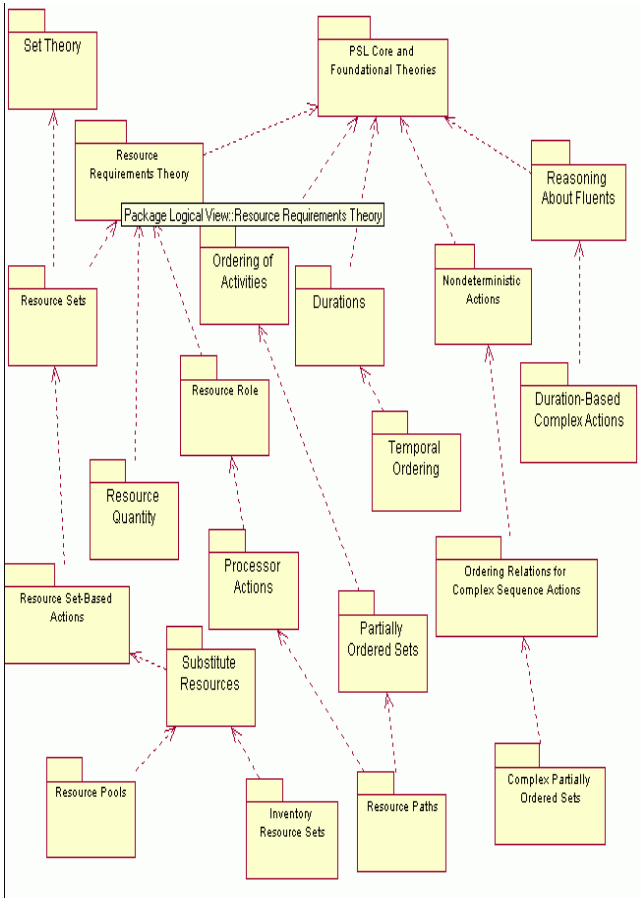
```
A sel   <cond-1>
  do B;
A alt   <cond-2>
  do C;
A end
```

Pseudocode

```
if  <cond-1>  then
  do B;
else if   <cond-2>  then
  do C;
endif
```

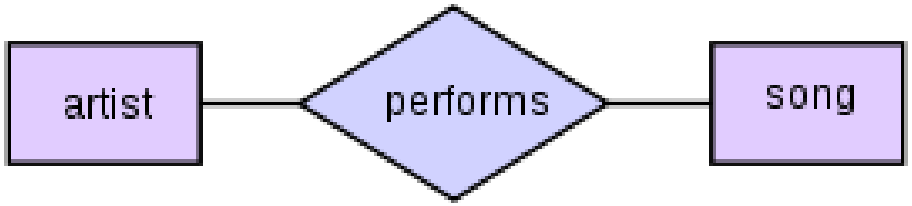
Process Specification Language

- The Process Specification Language (PSL) is used to describe different kinds of processes.
- It defines a neutral representation for manufacturing processes that supports automated reasoning.
- The terms are specified in an ontology that provides a formal description of the components and their relationships that make up a process. The ontology was developed at the NIST, and has been approved as an ISO international standard.
- PSL can be used for the representation of manufacturing, engineering and business processes, including production scheduling, process planning, workflow management, business process reengineering, simulation, process realization, process modeling, and project management.



Entity-Relationship Model

- An entity-relationship model (ER model) is an abstract and conceptual representation of data.
- Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion.
- Diagrams created by this process are called entity-relationship diagrams or ER diagrams.
- ER diagrams usually refers to the techniques proposed in Peter Chen's 1976 paper. However, variants of the idea existed previously, and have been devised subsequently.



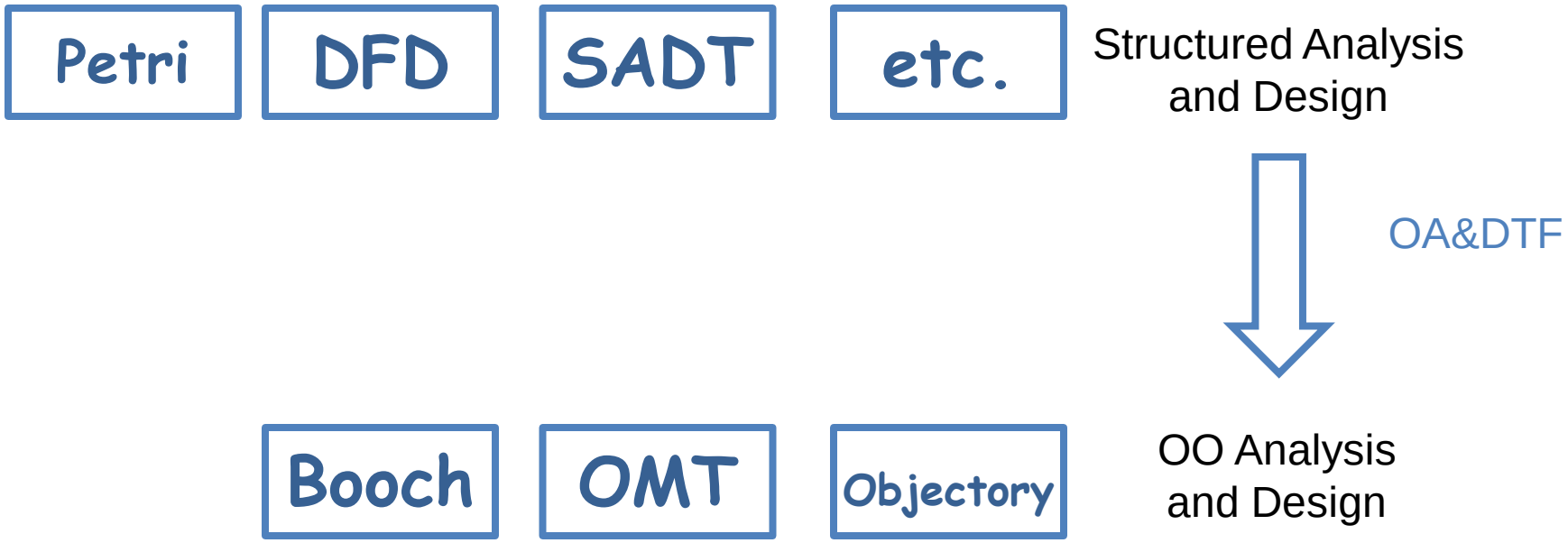
AD/Cycle

- CASE tools (Computer Assisted Software Engineering) were at their peak in the early 1990s. IBM proposed AD/Cycle, an alliance of software vendors centered around IBM's Software repository using IBM DB2 in mainframe and OS/2.
- The main ideas were metamodel based interoperability of CASE tools and repository support for models (IBM repository manager).
- AD/Cycle was a failure
  - With the decline of the mainframe, AD/Cycle and the Big CASE tools died off, opening the market for new generation CASE tools. Most of the leaders of the CASE market of the early 1990s ended up being purchased by Computer Associates.
  - An announced goal of AD/Cycle by IBM was to achieve a 10-fold improvement in programmer productivity.
- But unfortunately the MDA people have not learn from this failure

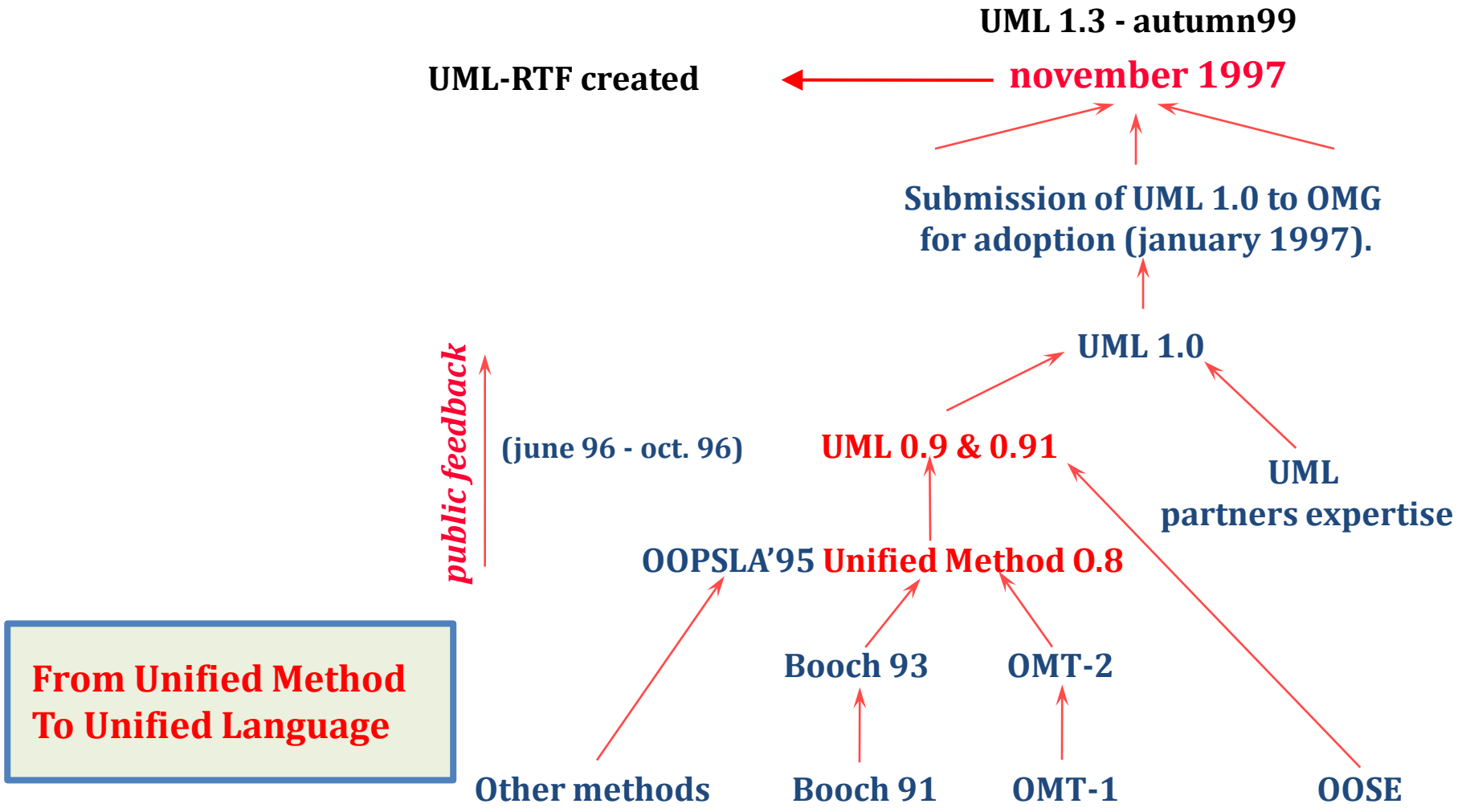
The story started in1986

- The story started 25 five years ago in Portland
  - *The first OOPSLA was held in Portland, in November 1986. About 600 people attended, about 50 papers were presented, and the attendees heard about Smalltalk, Lisp, Flavors, CommonLoops, Emerald, Trellis/Owl, Mach, Prolog, ABCL/1, prototypes, and distributed/concurrent programming.*
- No mention of models in the program, **but** there was a panel:
  - *Object-Oriented Programming without an Object-Oriented Language – Panel with Grady Booch, Ed Seidewitz, Mike Start, Donald Firesmith*
  - OOPSLA then regularly hosted a number of tutorials on various **OO methods** to use with OOP
  - OOPSLA tutorials + OMG OOADTF ⇨ Creation of UML in 1997

From Procedures to Objects (OOP)



UML main contribution: Separation of concerns



UML 1.0

Minutes of the OMG ADTF Meeting in Nice France

The following are the minutes of the OMG Analysis and Design Task Force meeting held in Nice France on November 7, 1996. Mike Bradley of BellSouth chaired the meeting. Mike Meier of IBM took minutes.

Attendees

- o Mike Bradley - BellSouth
- o Don Kavanaugh - SSA Object Technology
- o Mike Meier - IBM
- o Guus Ramackers - Oracle
- o Jim Rye - Digital
- o Michael Senbiss - SAP
- o Wolfgang Zuck - SAP AG
- o Oliver Weigert - SAP AG
- o Huet Landy - DISA
- o Cris Kobryn - MCI Systemhouse
- o Serban Gheorghe - ObjecTime Limited
- o Philippe Desfray - Softeam
- o Trygve Reenskaug - Taskon
- o Karl-Heinz Weiss - Pabl.Admin.Berlin
- o Jean Bézivin - University of Nantes
- o Joaqin Keller - France Telecom keller@ws.net.fr
- o Ed Eykholt - Rational Software
- o Georges Reich - Reich Technologies
- o Armond Inselberg - Lockheed Martin
- o Coun Scott - Andersen Consulting
- o Oliver Remand - Reich Technologies
- o Dipayan Gangopadhyay - IBM
- o Cory Casanova - Data Access
- o Geoff Hambrick - IBM
- o Yinchi Nini - Ricoh
- o Dan Klawitter - Boeing
- o Dominique Didov - Initut Eurecom
- o Norbert Bieberstein - IBM
- o Sridhar Iyengar - sridhar.Iyengar@mv.unisys.com1

November 1996

The Unified Modeling Language

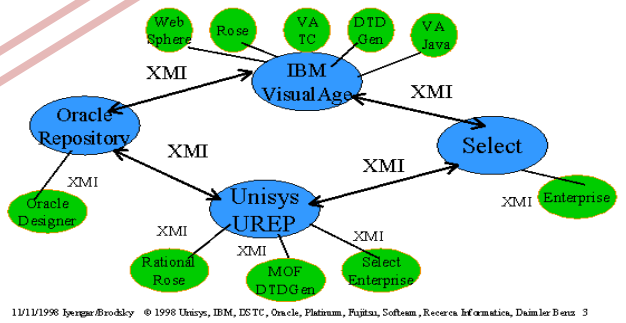
Update to OMG OA&DTF

26 June 1997

Montreal, Canada

Ed Eykholt, Cris Kobryn,
Sridhar Iyengar, Gunnar Overgaard,
Grady Booch, Ivar Jacobson,
Jim Rumbaugh

June 1997

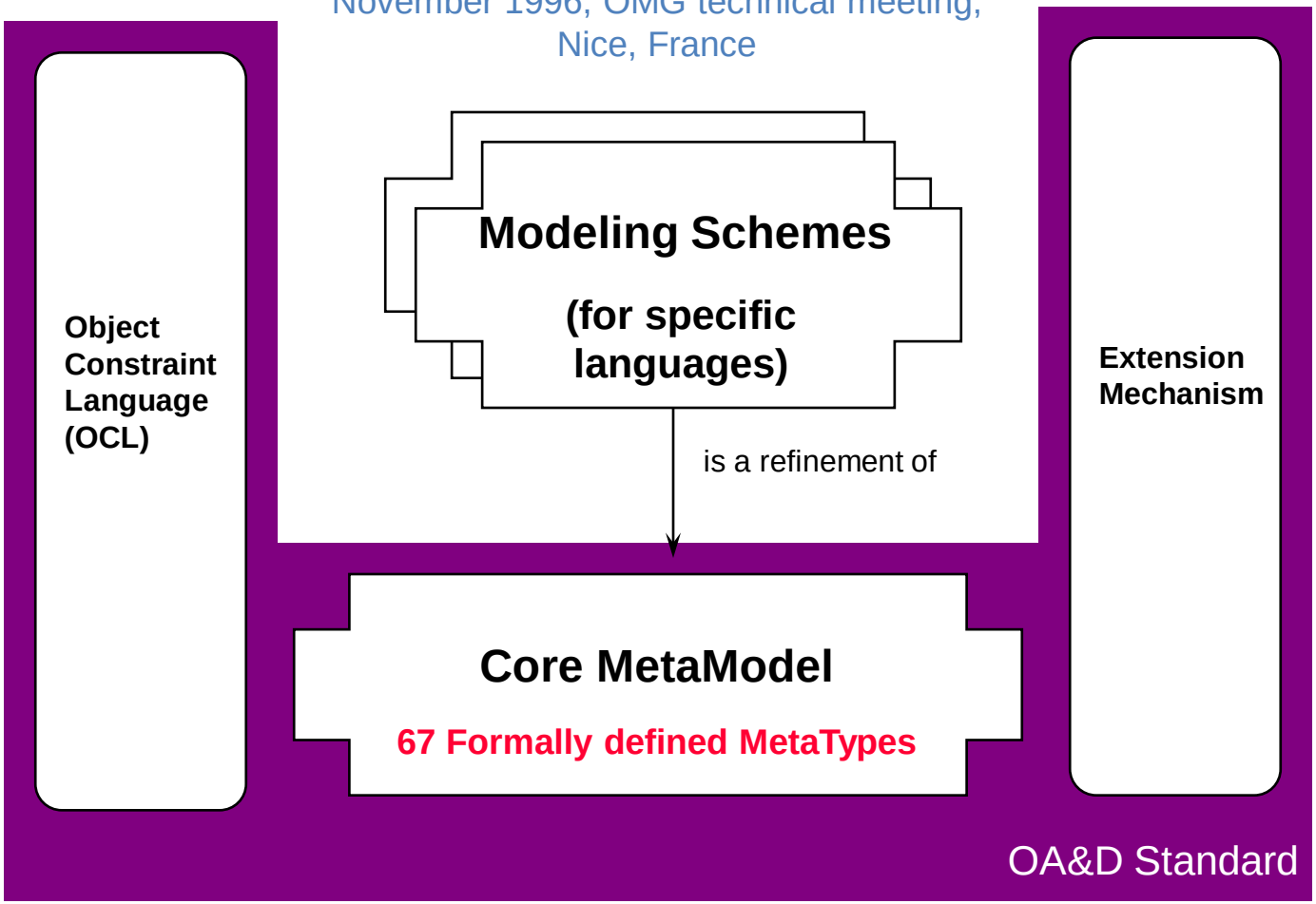


From SMIF to XMI

IBM/Objectime Proposition

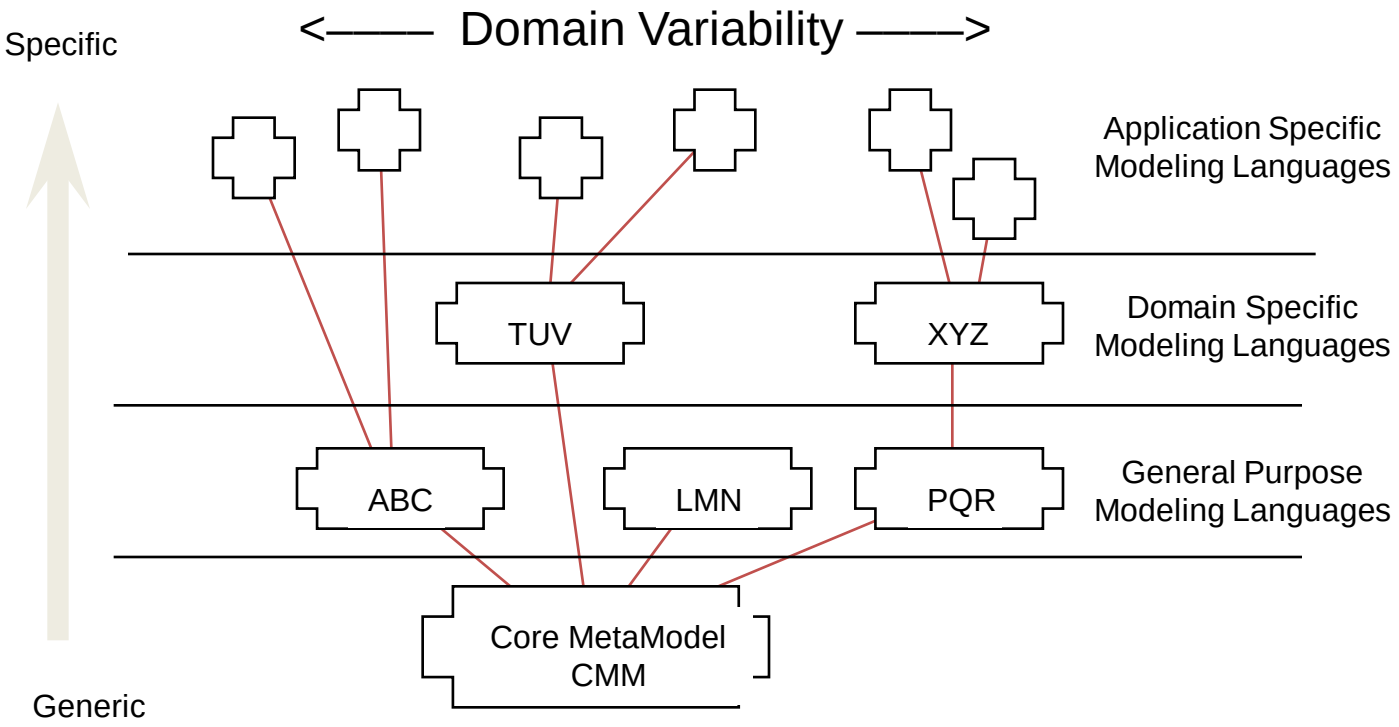
"Schemes" ~ CDIF «Subject Areas»

November 1996, OMG technical meeting, Nice, France



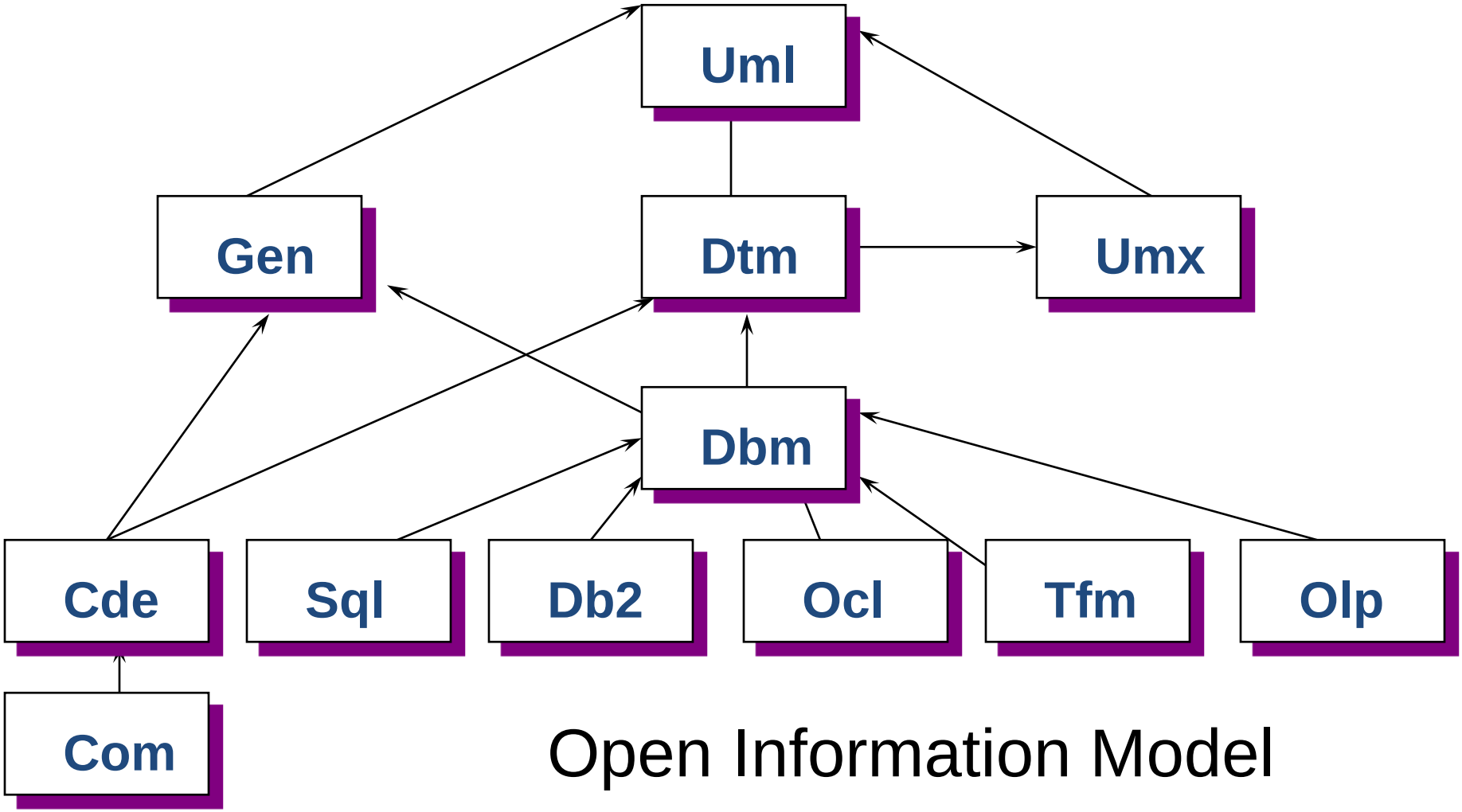
CMM: Multi language support

November 1996, OMG technical meeting,
Nice, France



Families of modeling languages

Microsoft OIM proposal

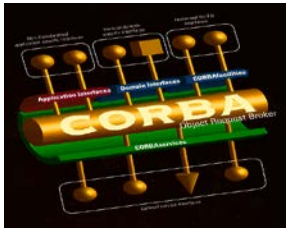


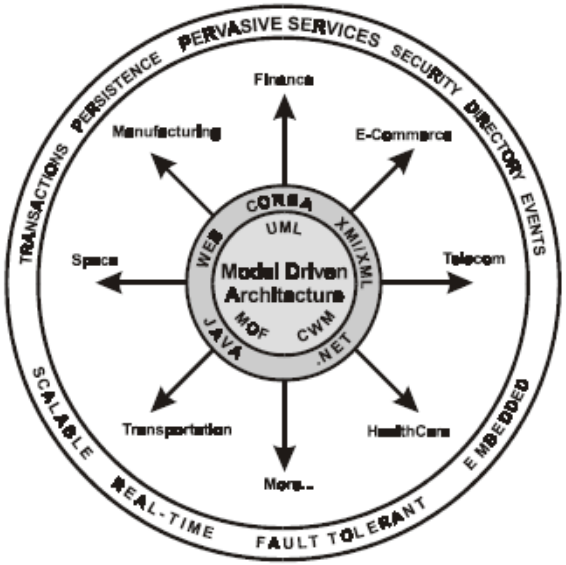
The initial move: from Middleware to Modelware

Model Driven Architecture

by Richard Soley and the OMG Staff Strategy Group

Object Management Group
White Paper
Draft 3.2 – November 27, 2000





Preface: OMG's Accomplishments

It's about integration. It's about interoperability. For eleven years, the Object Management Group (OMG) has focused on making sure that you can integrate what you've built, with wha you're building, with what you're *going* to build. As the pace of technology continues to quicken, and the demands of integrating your existing legacy systems, your new intranet, and your e-business fall on your shoulders, you need an architecture that makes interoperability central to your infrastructure. The bad news is that there will never be a single operating system, single programming language, or single network architecture that replaces all that has passed; the good news is that you can still manage to build systems economically in this environment.

Infrastructure Standards

CORBA was a powerful first step, but we have more steps to take.

The OMG has led the way in providing vendor- and language-independent interoperability standards to the enterprise. Our mission started with a focus on CORBA which took its most recent shape in 1999 with

PIM/PSM based definition.

Very UML-centric

Separating the platform independent and dependent parts of a system (PIM/PSM)

We don't want anymore to pay such a high price for simply moving our information system to a new middleware platform (COM, CORBA, Java, HTML, XML, DotNet, etc.) when our business system stays stable.
We are prepared to pay a last price for building the abstract models of our business and services that will guarantee us against technological obsolescence.
From there, any platform provider will also have to provide the mapping solutions from standard business models before we buy.

November 2000

In search of definitions

- MDA™ has been defined more twelve years ago,
- Tools and applications have been broadly produced and used,
- Some industry leaders accept the move to model driven practices as an irreversible trend,
- Research is producing hundreds of papers each year in numerous and well attended conferences,
- … and however we are not yet able to produce a satisfactory and consistent completely consensual definition of model driven engineering!

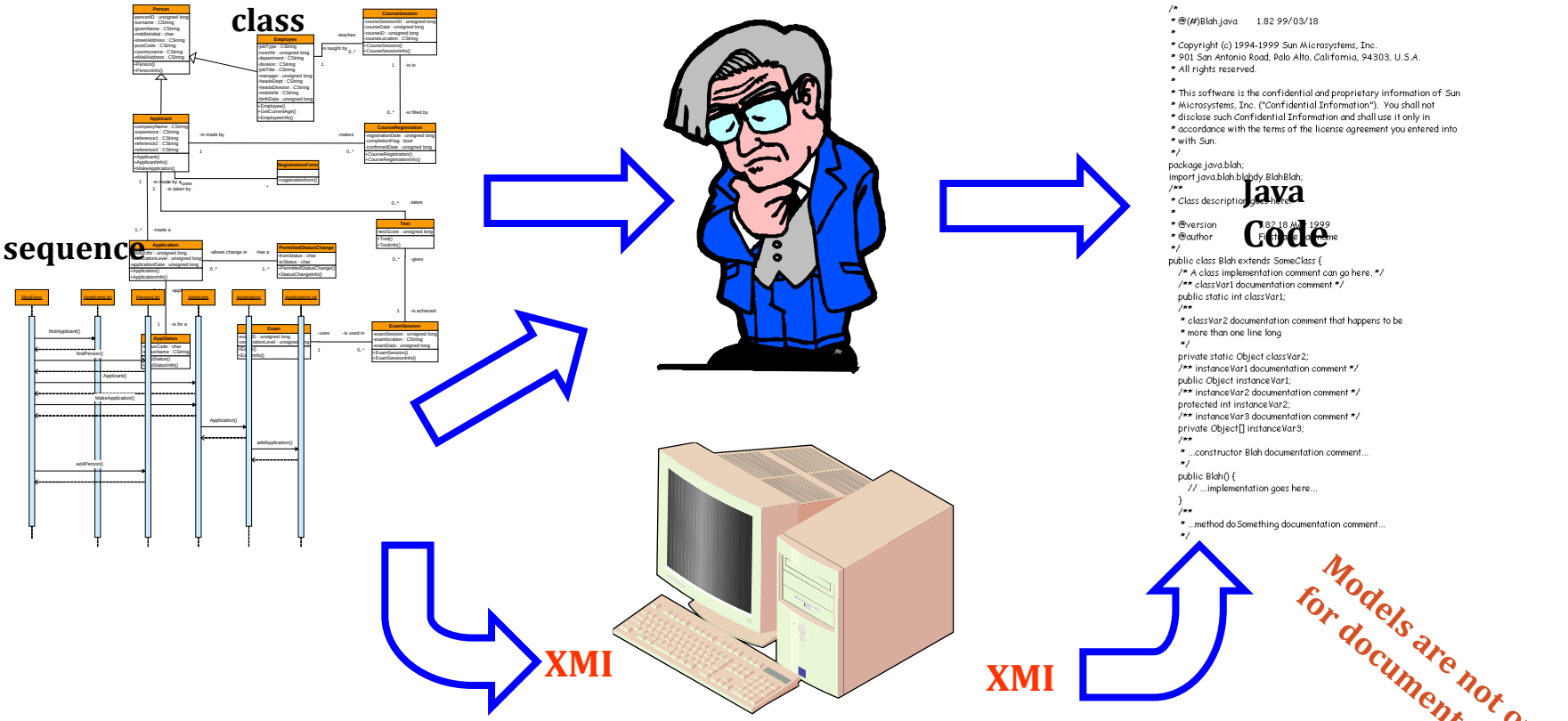
A definition of MDA

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.
- 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.

No much mention of PIM/PSM. Much less UML-centric

From contemplative to productive approaches

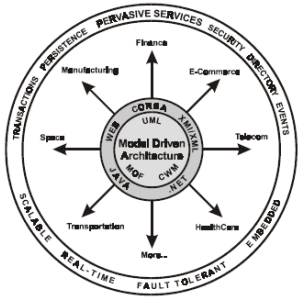


From human-readable to computer-understandable.
From **hand-crafting** (e. g. Design Patterns) to **full automation** (e. g. Model Transformation)

Starting point : The OMG definition of MDA™

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

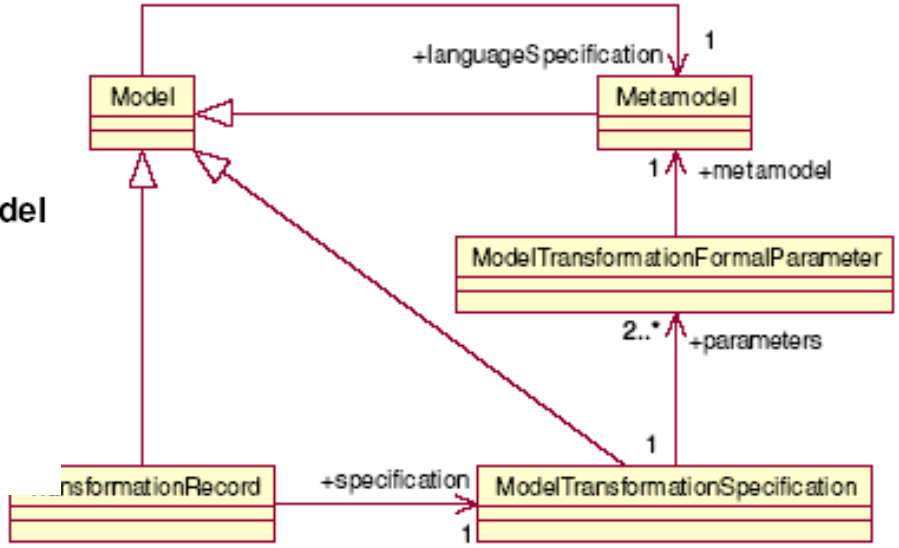
A Proposal for an MDA Foundation Model



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 …

Some loose definitions of what is a model

- Phil Bernstein, "A Vision for Management of Complex Systems".

  A model is a complex structure that represents a design artifact such as a relational schema, an interface definition (API), an XML schema, a semantic network, a UML model or a hypermedia document.

- OMG, "UML Superstructure".

  A model captures a view of a physical system. It is an abstraction of the physical system, with a certain purpose. This purpose determines what is included in the model and what is relevant. Thus the model completely describes those aspects of the physical system that are relevant to the purpose of the model, at the appropriate level of detail.

- OMG, "MDA Guide".

  A formal specification of the function, structure and/or behavior of an application or system.

- Steve Mellor, et al., "UML Distilled"

  A model is a simplification of something so we can view, manipulate, and reason about it, and so help us understand the complexity inherent in the subject under study.

- Anneke Kleppe, et. al. "MDA Explained"

  A model is a description of (part of) a system written in a well-defined language. A well-defined language is a language with well-defined form (syntax), and meaning (semantics), which is suitable for automated interpretation by a computer.

✓*All of these definitions are partially correct*
✓*None is complete*
✓*None is really useful for the real engineer*
✓*We need a workable definition for "model"*

What is a model?

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

"The Nature of Modeling."
Jeff Rothenberg
in Artificial Intelligence, Simulation, and Modeling,
L.E. William, K.A. Loparo, N.R. Nelson, eds.
New York, John Wiley and Sons, Inc., 1989, pp. 75-92

http://poweredge.stanford.edu/BioinformaticsArchive/PrimarySite/NIHpanelModeling/RothenbergNatureModeling.pdf

Bernstein: united world of models

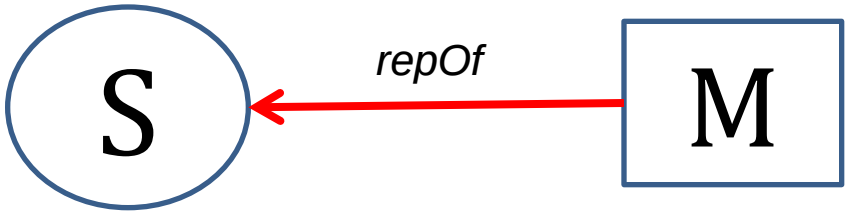
By **model** we mean a complex structure that represents a design artifact, such as a relational schema, object-oriented interface, UML model, XML DTD, web-site schema, semantic network, complex document, or software configuration. Many uses of models involve managing changes in models and transformations 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

Another definition of MDE

- The use of typed graphs as the main artefact to **represent** phenomenon of the real world (to understand them, to analyze them, to act on them, to use them in the planning of actions)

- Systematic use of the representation relation *repOf (M,S)* between systems and models



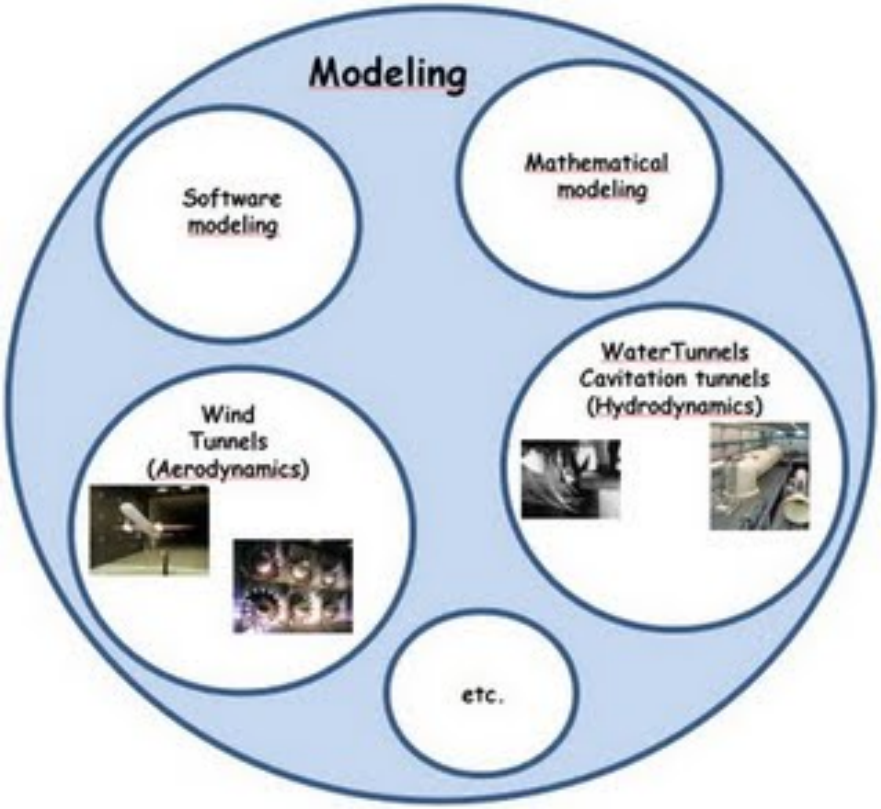
- Three main operations on models: create/delete, store/retrieve, transform.

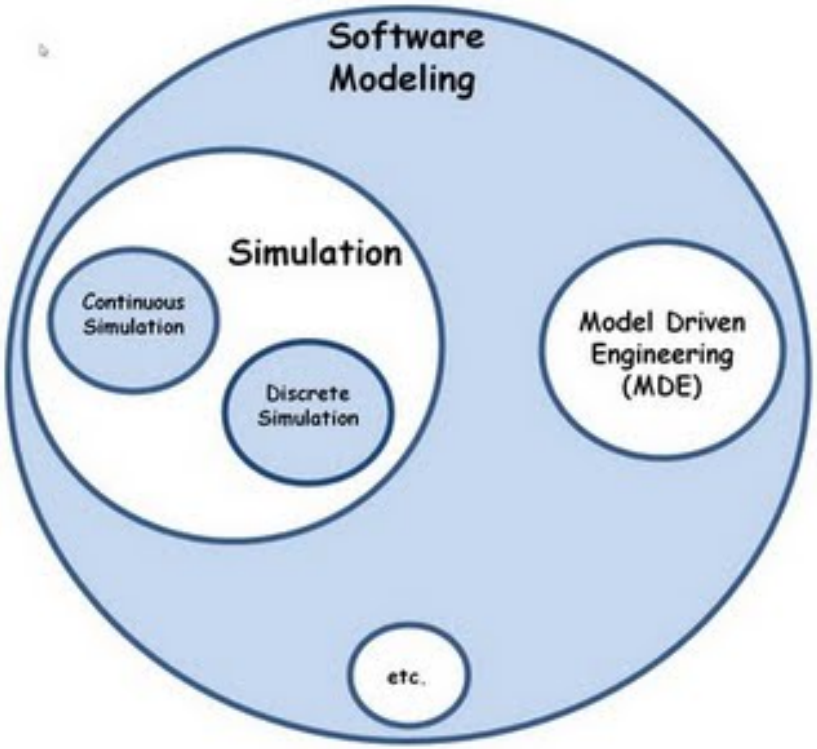
➢ Issues of Graph-based Representation

A Definition of MDE

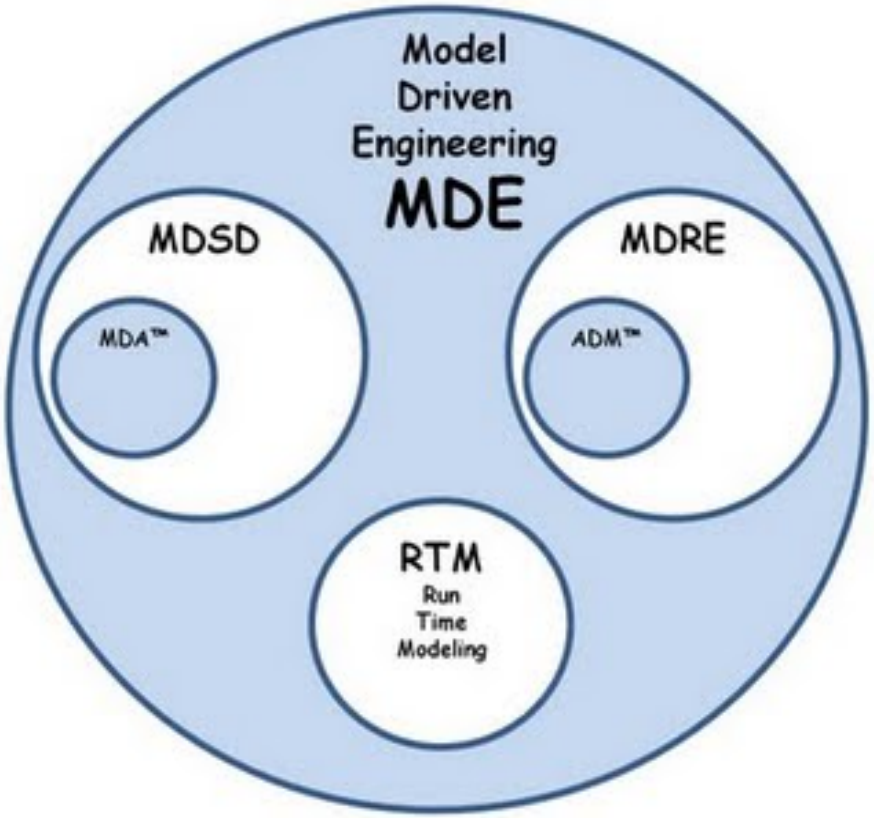
***MDE** is a set of **well defined practices** based on tools that use at the same time **metamodeling** and **model transformations,** to achieve some **automated goals** in the **production**, **maintenance** or **operation** of **software intensive systems**.*

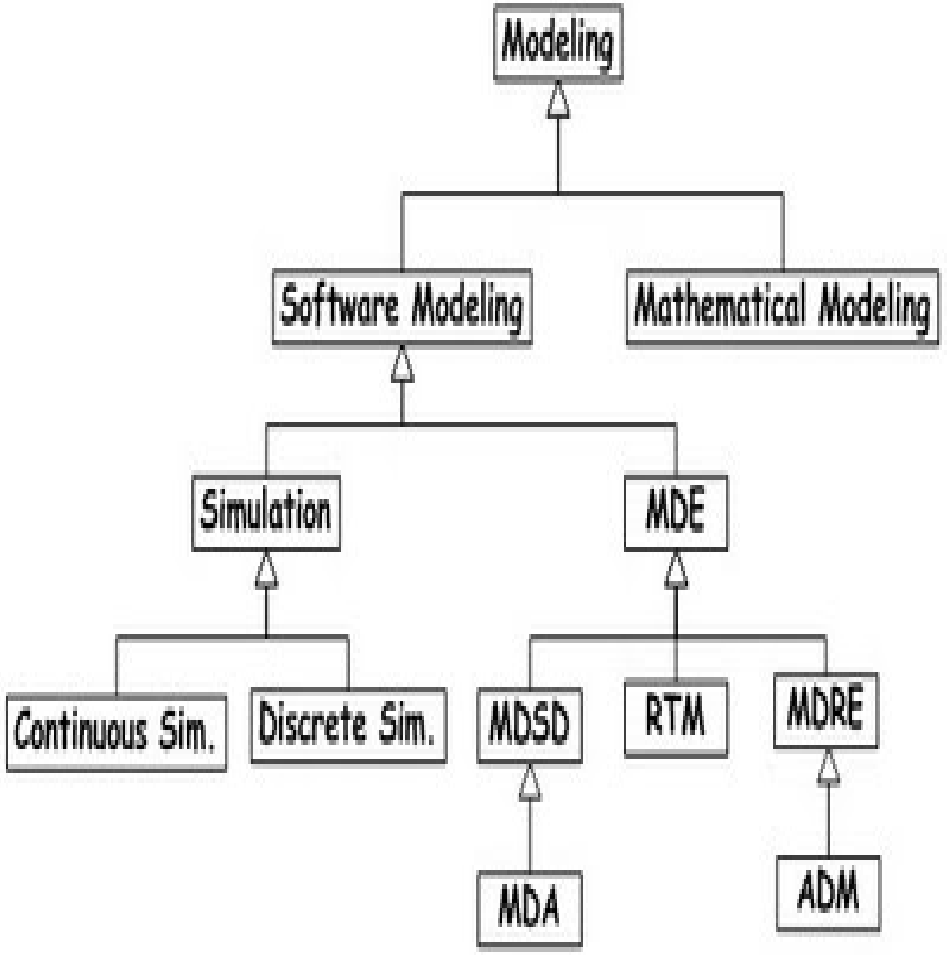
Modeling at large





Modeling at large



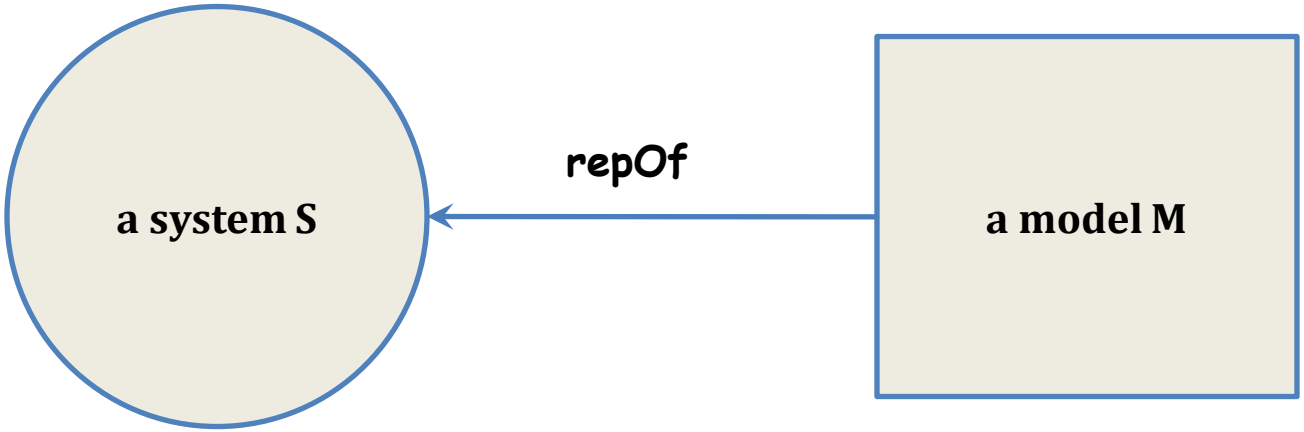


The essence of MDE

- Three main ideas
  - Models represent systems
  - Models conform to Metamodels
  - Models may be transformed into other models
- These three ideas need to be refined later
  - Taxonomy of Metamodels
  - Taxonomy of Operations
  - etc.

Systems and Models

Squares and Circles

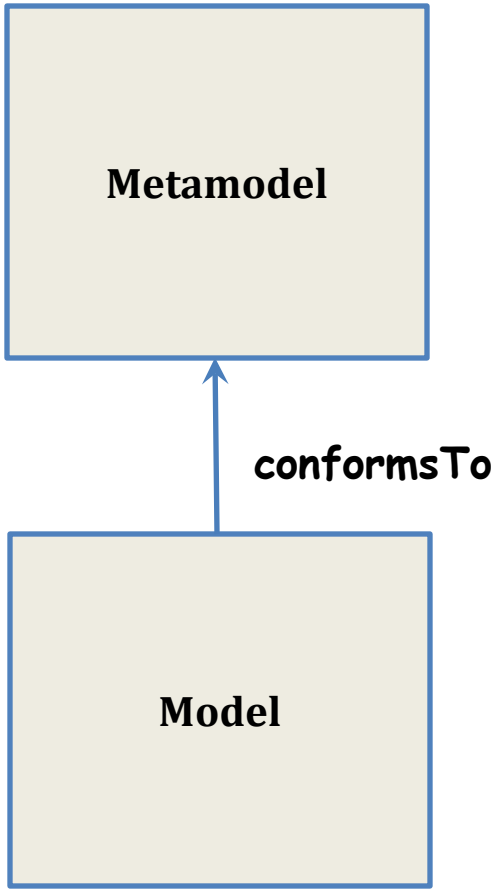


A situation or a phenomenon of the real or imagined world.

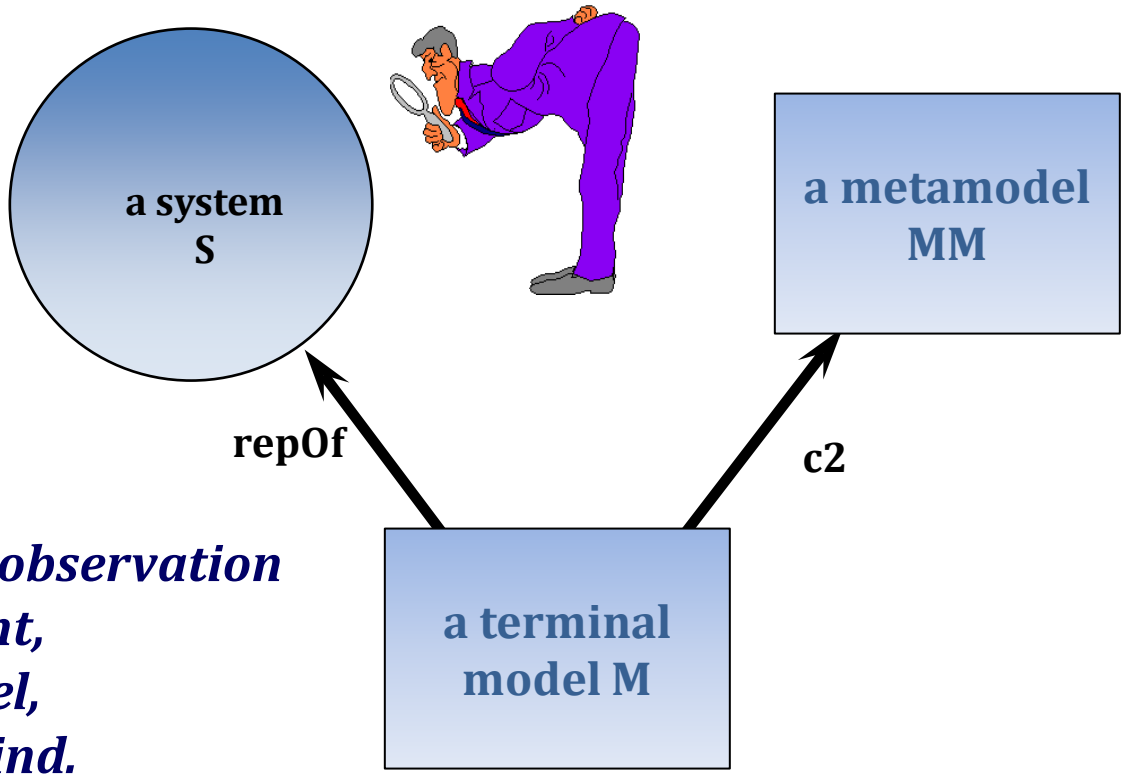
Metamodeling

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.

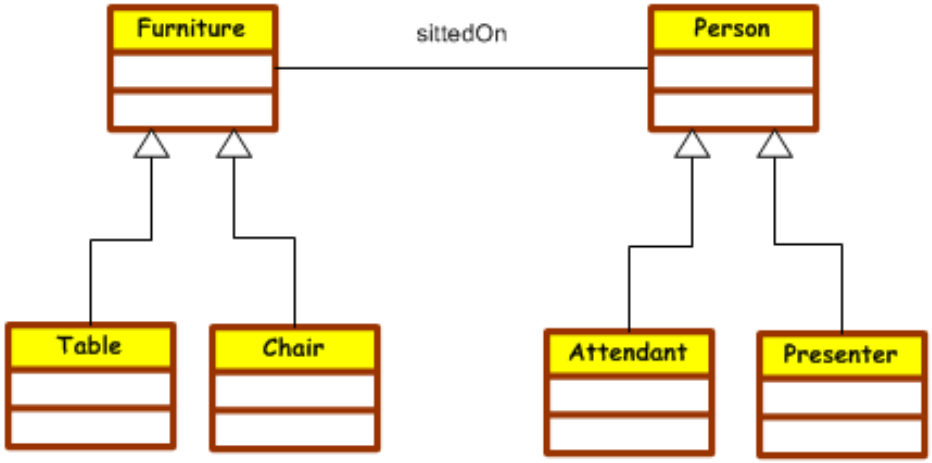


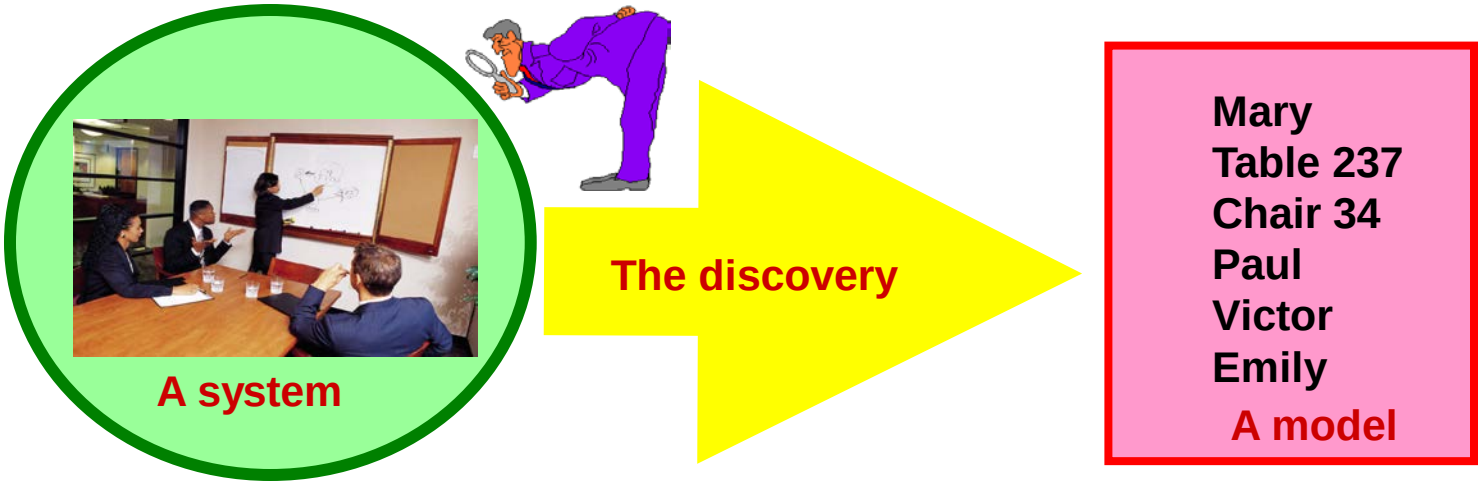
System Metamodel Model



A model is the result of the observation of a system, by a given agent, with respect to a metamodel, with a given objective in mind.

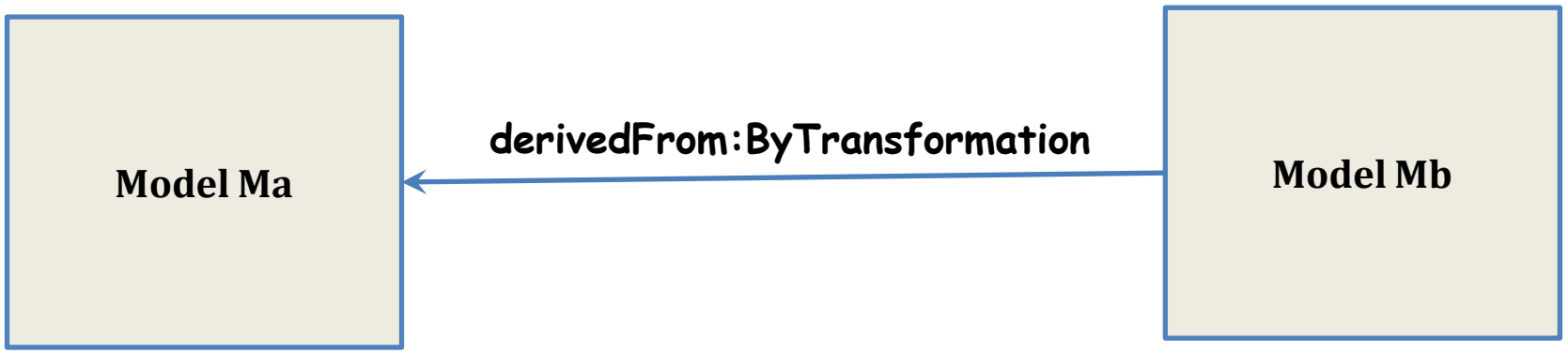
Metamodels acting as filters





Model Transformations

A model may be transformed into another model



This is achieved by a well defined model transformation.
There are many kinds of transformations.

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

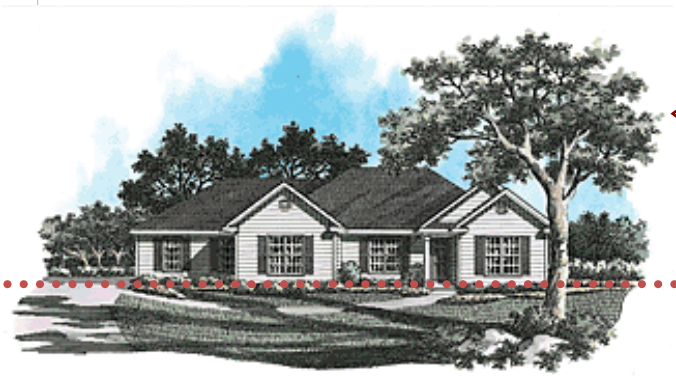
国立情報学研究所
National Institute of Informatics

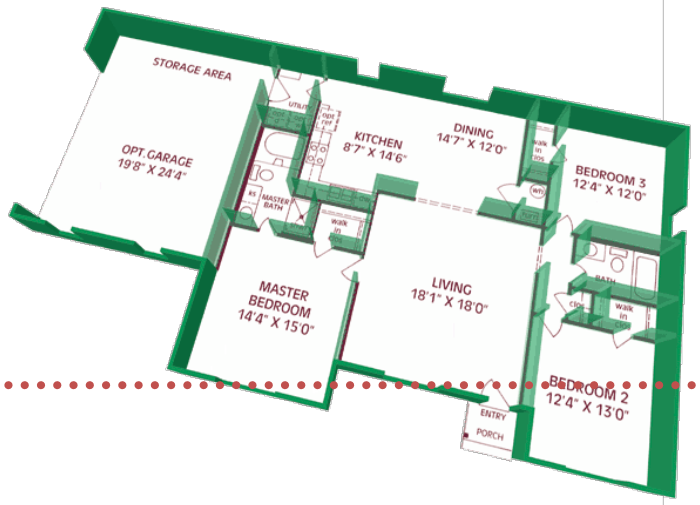
Production of a system from a model

S

M

Production of a system
from a model.
For example construction a building
from its floor plans.





S

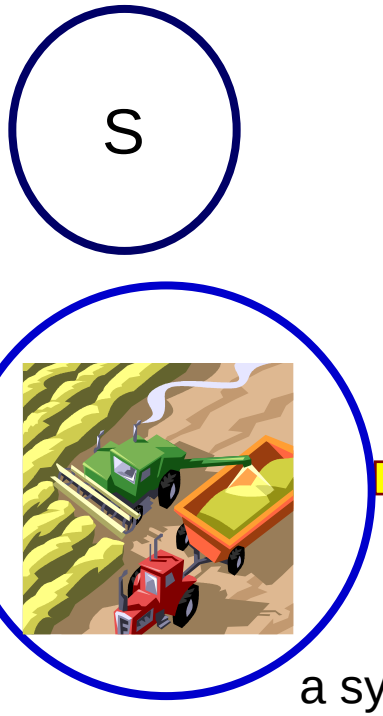
repOf

M

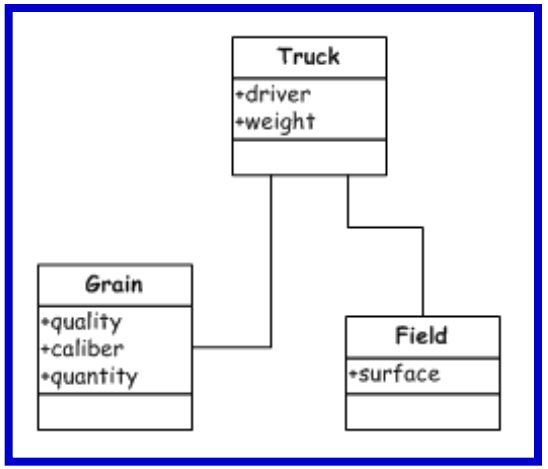
{postcondition}

Production of a model from a system

Production of a model from a system. For example production of the business model of a farm in the UML language.



a system S
e.g., agricultural enterprise



a business model M
of the agricultural enterprise

{postcondition}

Prescriptive or descriptive modeling

[M => S] Prescriptive

[S => M] Descriptive

[S <=> M] Synchronized

Possible combinations like in code production or refactoring: [S=>M=>S]

In all these situations,
the relation *repOf(M,S)* applies,
as a postcondition or invariant

Systems belong to reality

The real world (or reality)
is not only
the present physical world.

Descriptive:
A model may be built to understand the real world.
Prescriptive:
A model may be built to help construct a new system
And thus contribute new items (systems) to the real world

repOf

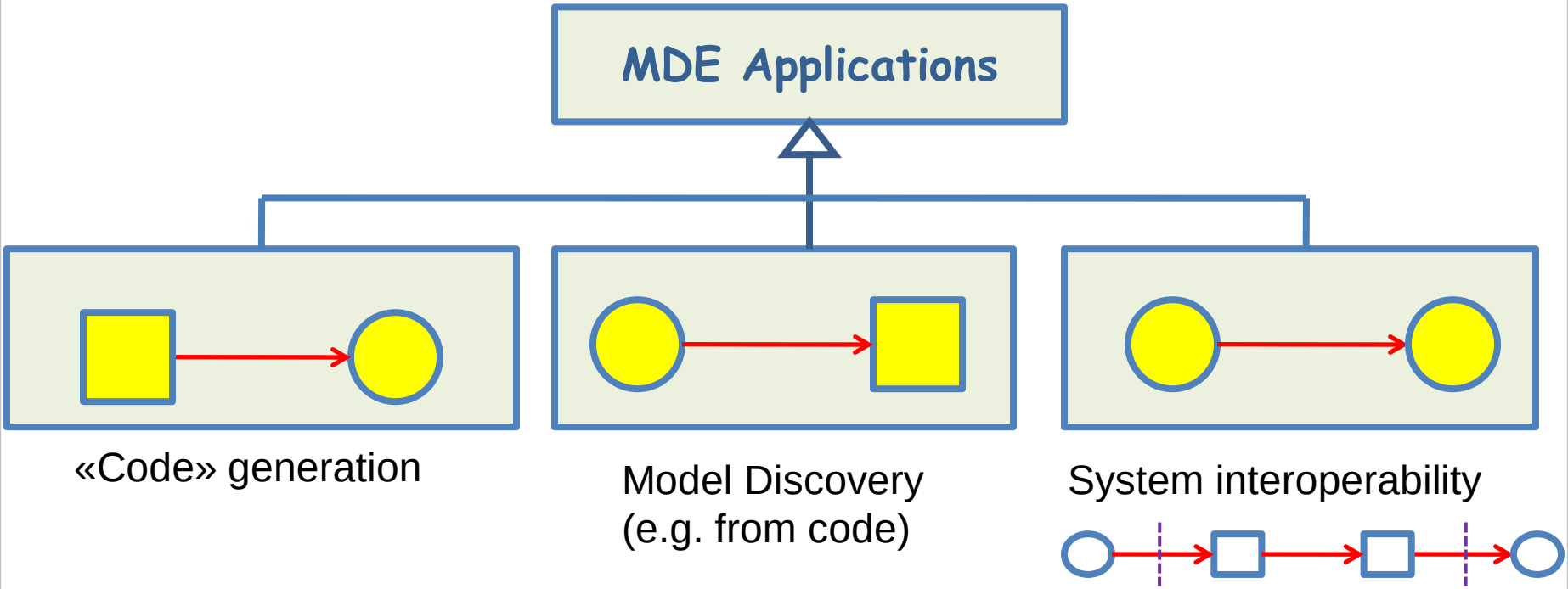
The Pharos (Lighthouse) of Alexandria

System

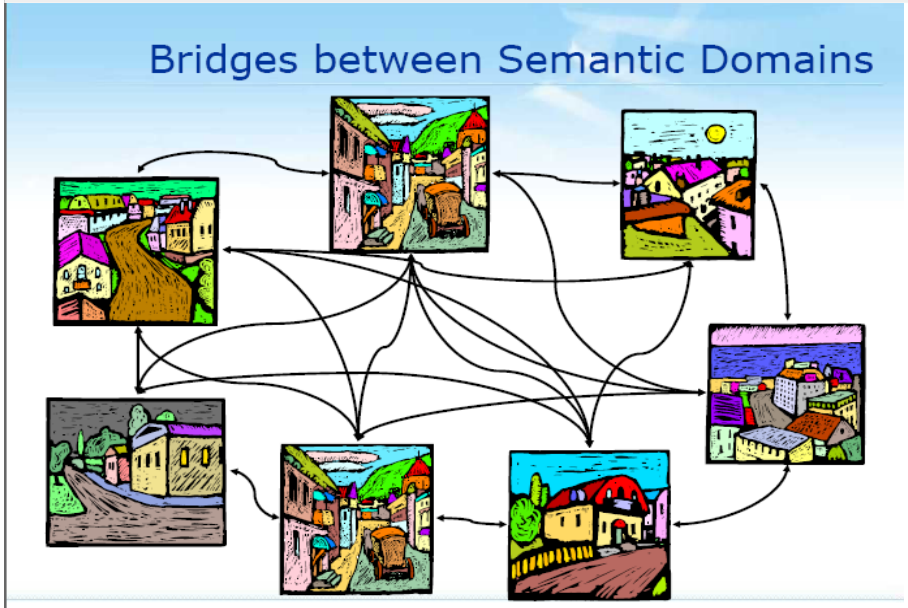
Model

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)



The «Village metaphor» by Antonio Vallecillo



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

"Men build too many walls and not enough bridges."

Isaac Newton

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

Main messages of the lecture

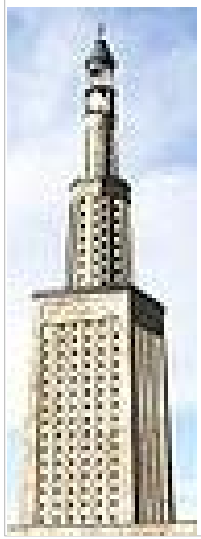
- Model Driven Engineering (i.e. the consideration of models as first class entities and the representation of all types of artifacts by models) and its multiple variants are changing the landscape of software engineering, system engineering , data engineering and business engineering.

- Software Modeling is nearly as old as programming. Initially software modeling was mainly focused on non-executable requirement engineering.

- A lot of insight has been accumulated through the various periods of modeling.
  - Plenty of specific modeling point of views, from PNs to DFDs
  - The AD/Cycle period introduced metamodels and model repositories in the 90's
  - The UML proposal unified the visual concrete syntaxes
  - The MDA OMG initiative did put the focus on transformations
  - The recent evolution of MDE is mainly centered on DSLs

Thanks

- Questions?
- Comments?



JBezivin@gmail.com