

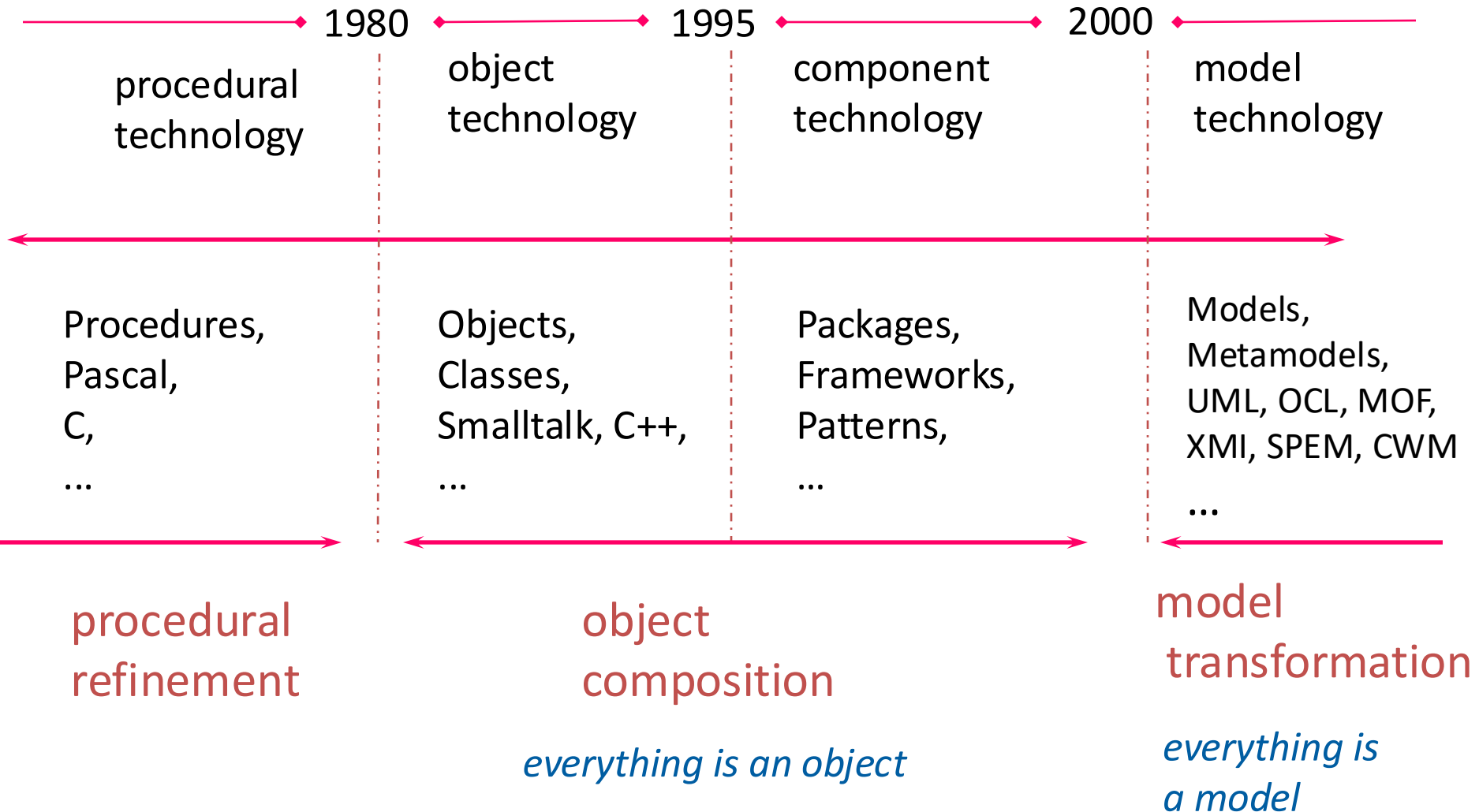
# Model Driven Engineering (MDE)

## *Principles, Standards & Technologies*

# Building SW is an *effortful* activity

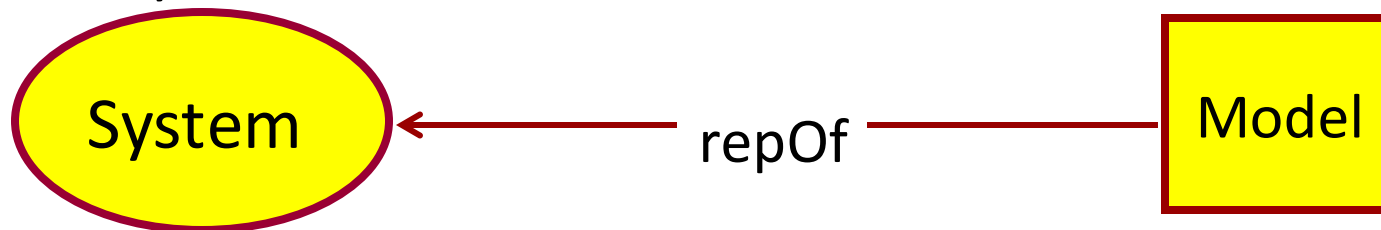
- Modern distributed software systems:
  - exploit various technologies (programming languages, scripting languages, middleware libraries, interface definition languages, database definition languages, query languages, etc.)
  - are expected to provide given levels of quality of service in terms of efficiency, reliability, scalability, security, etc., by taking into account highly variable *workloads* and *best-effort* network infrastructures
- Translating from the requirements of a business problem to a solution using these technologies requires a deep understanding of the many architectures and protocols that comprise a distributed solution
- To deal with such a complexity a paradigm shift is necessary, from *low-level coding* to *high-level modeling*

# From procedures to models



# The essential role of modeling

- Modeling is essential to human activity because every action is preceded by the construction (implicit or explicit) of a model
- What is (in short) a model?
  - a model is an abstract (i.e., simplified) representation of a system
  - and what is (in short) a system?
    - a system is a set of elements in interaction



# 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.*

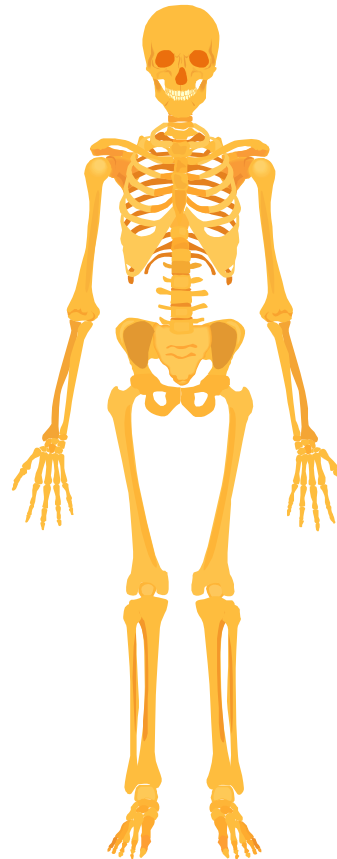
"The Nature of Modeling."  
Jeff Rothenberg

# A model is a partial view of a system

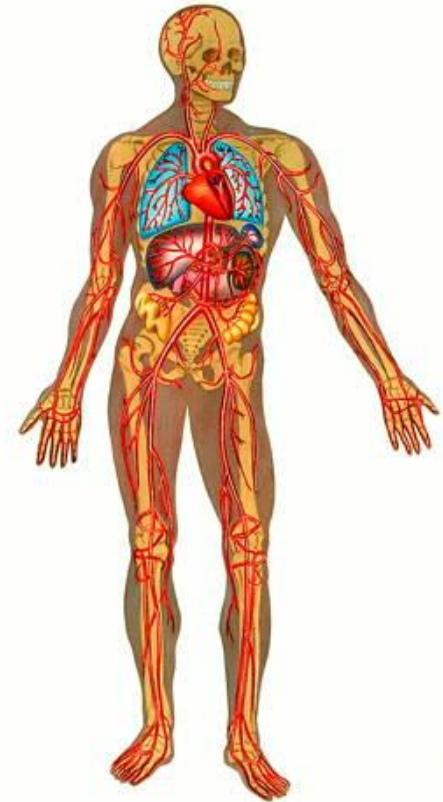
A system

represent

Several models  
(partial views)

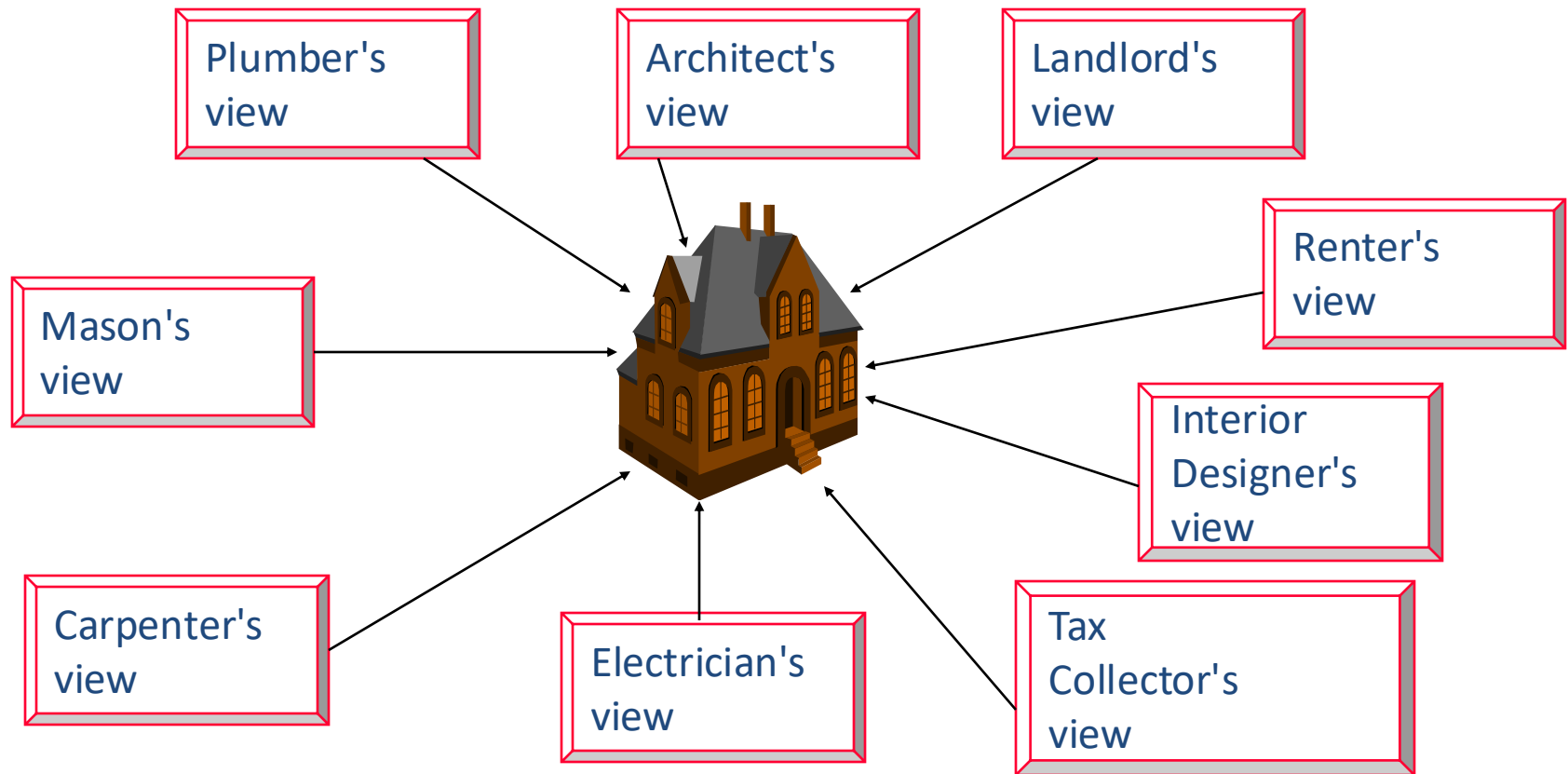


Skeleton  
model



Circulatory  
model

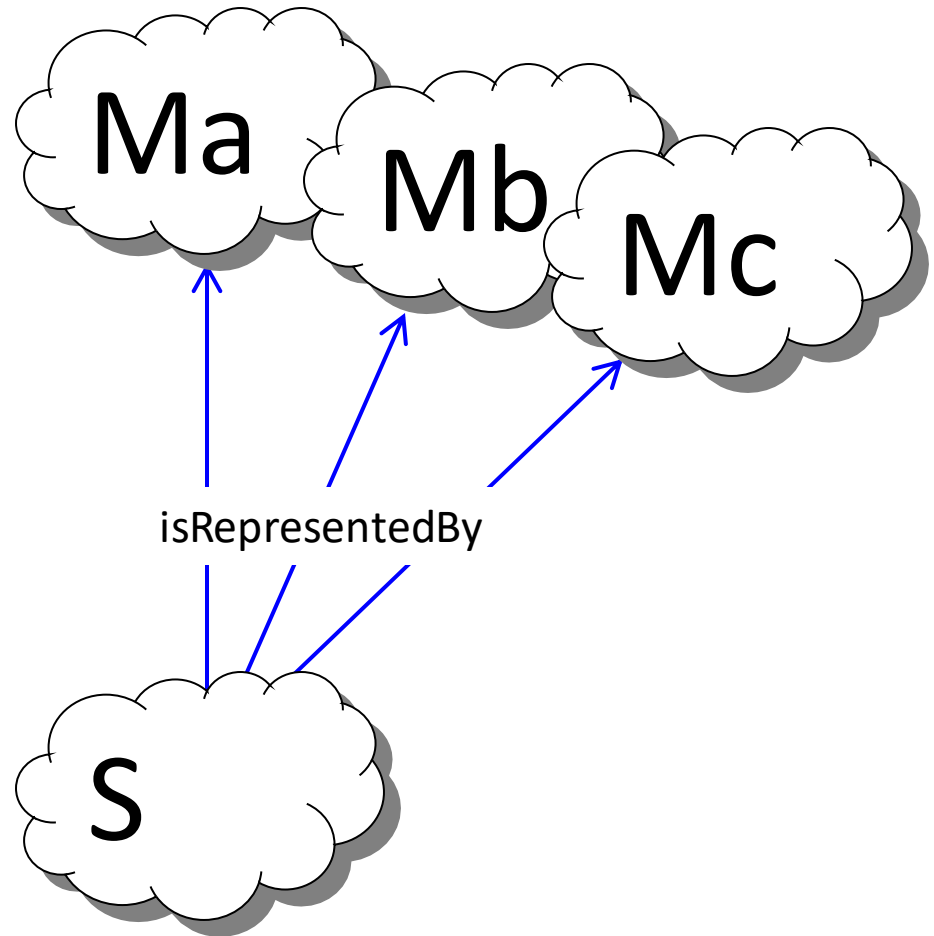
# Multiples views and coordinated DSLs



- Each view is expressed in a given domain language (DSL)
- Vocabularies of different corporations are different
- However they allow talking about a common building

# A model for each aspect

- A given system may have plenty of different models
- Each **model** represents a given aspect of the system



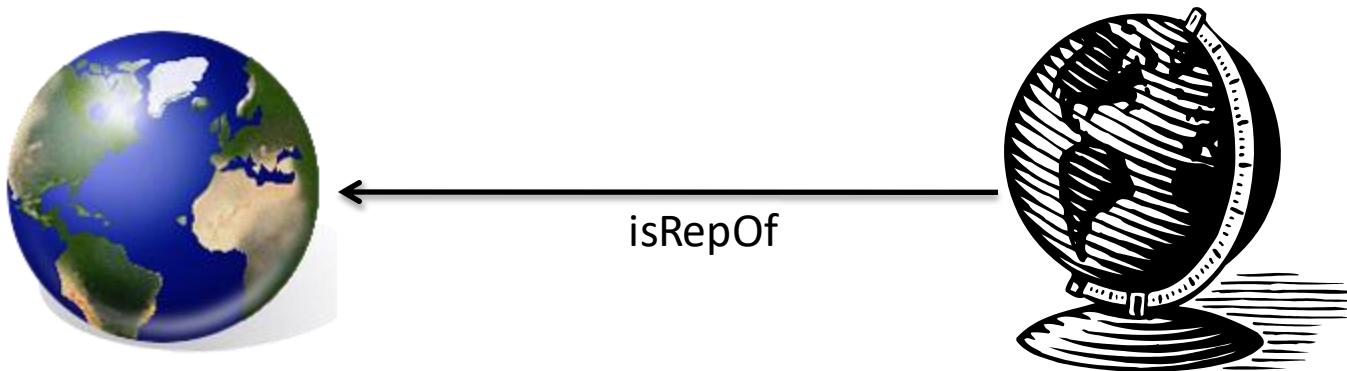
# Principle of *limited substitutability*

- 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

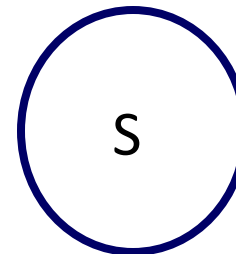
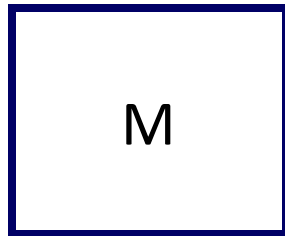
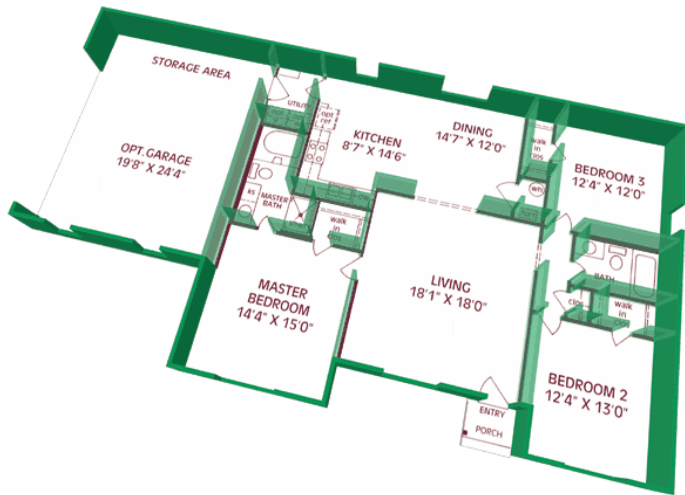


## As an example...

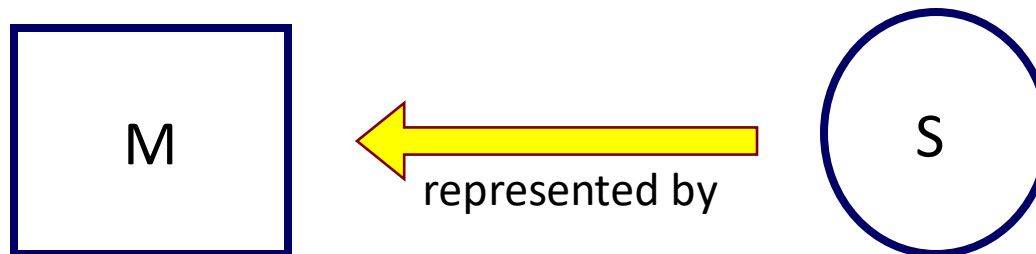
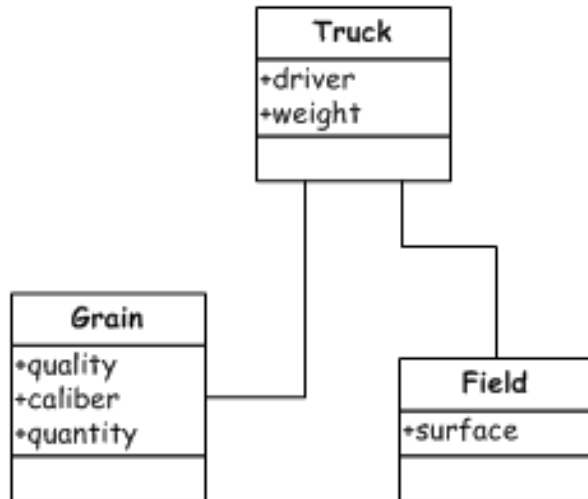
- A globe (model of the world) can provide an answer to the question: *“may I reach Tallin from Rome without using a boat?”*
- but wouldn't be able to provide an answer to the question: *“what is the temperature in Rome?”*



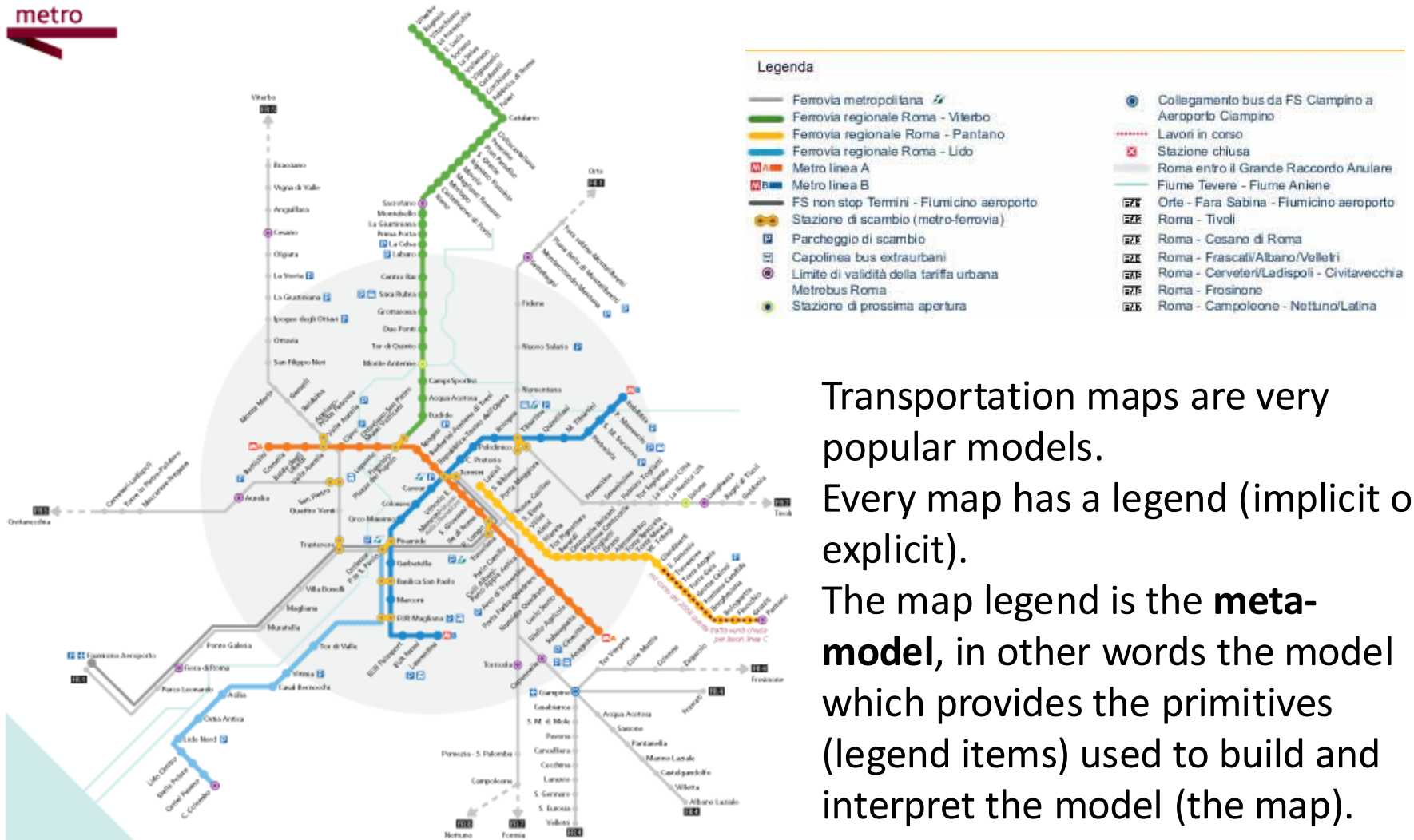
# Production of a system from a model (to build the system)



# Production of a model from a system (to describe/analyze a system)

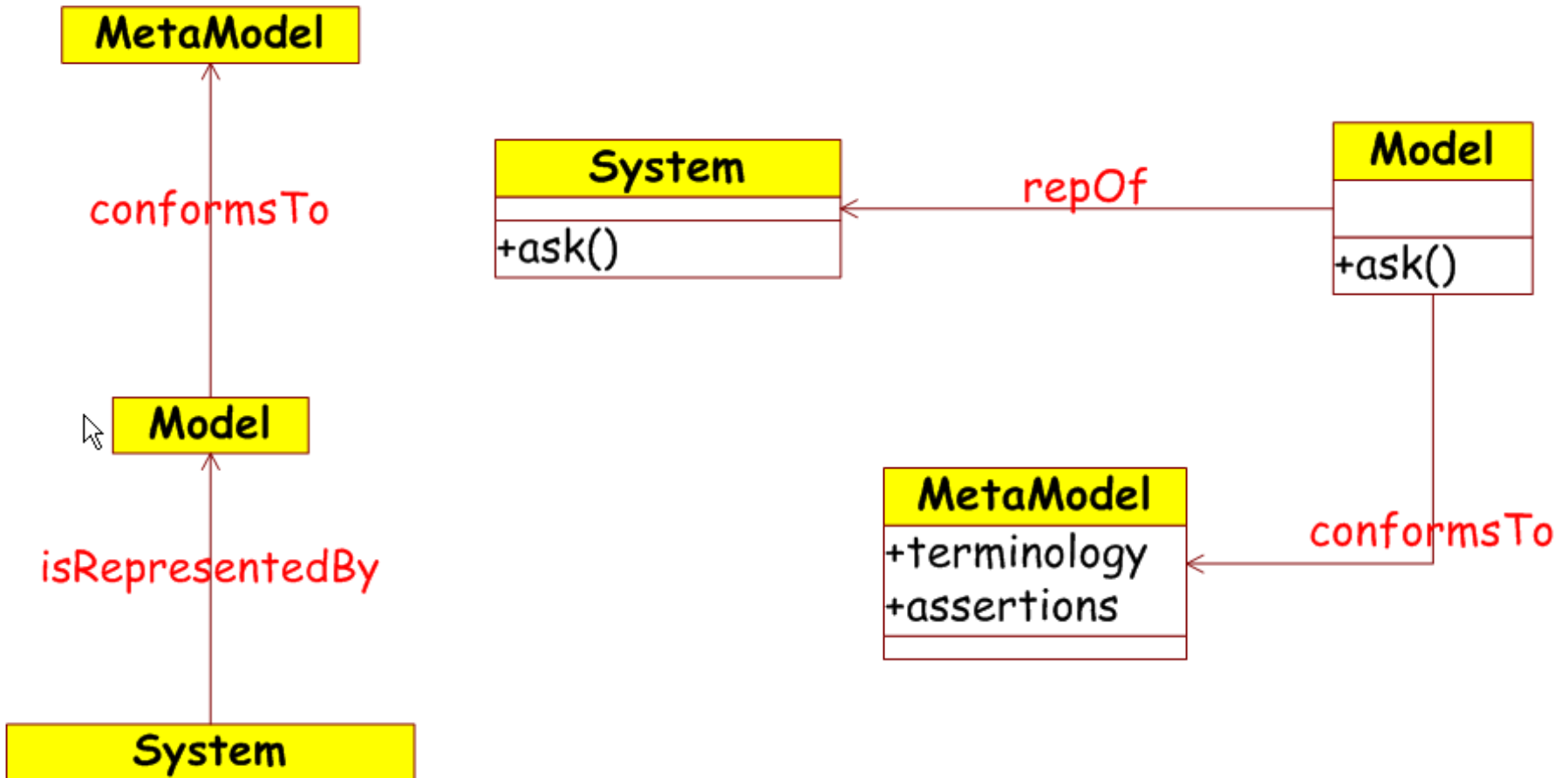


# How to build and interpret a model?

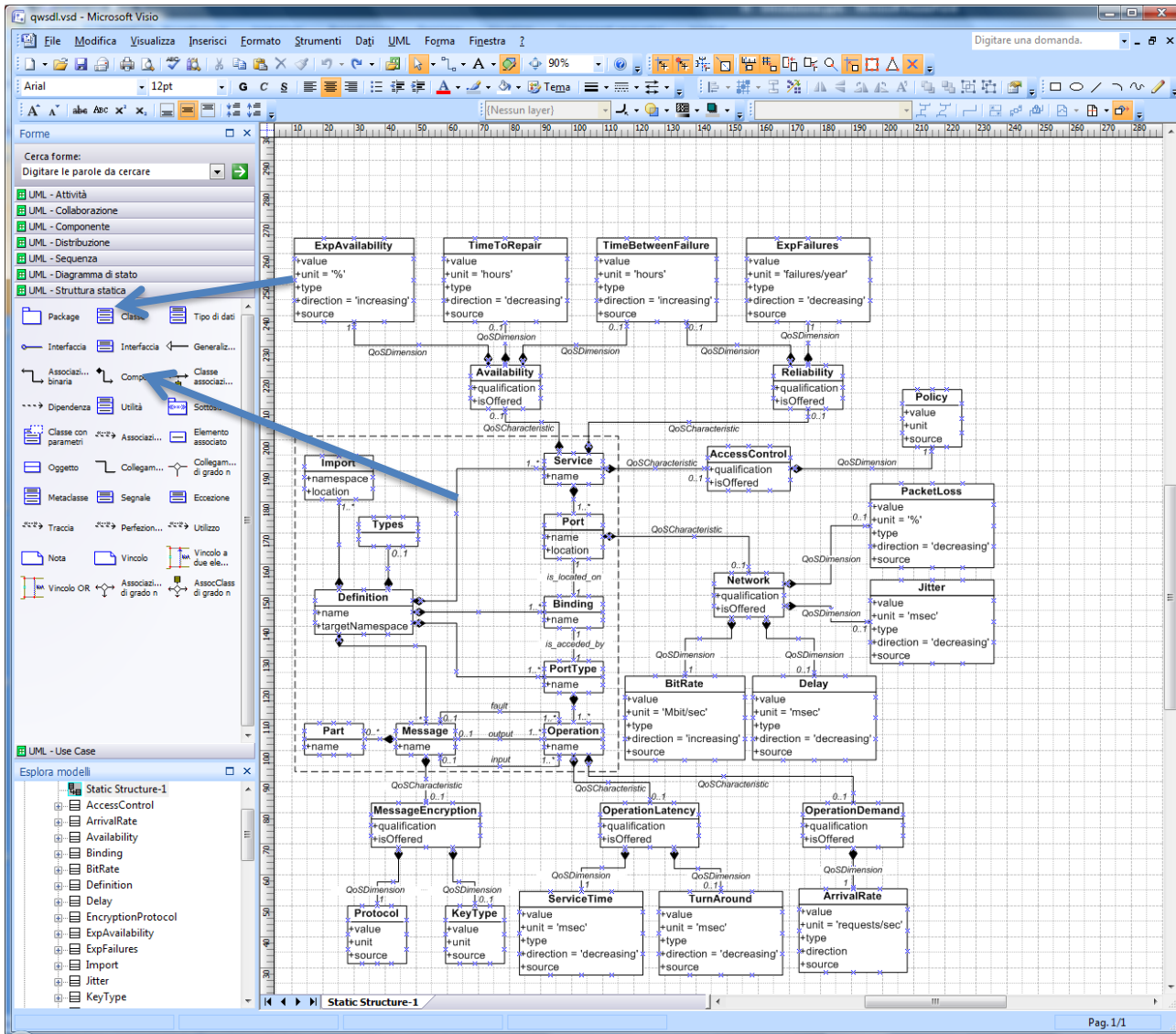


Transportation maps are very popular models. Every map has a legend (implicit or explicit). The map legend is the **meta-model**, in other words the model which provides the primitives (legend items) used to build and interpret the model (the map). Each map has to conform to its legend.

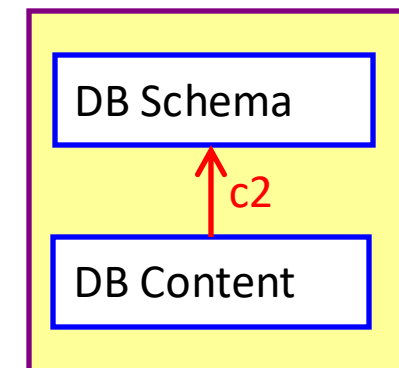
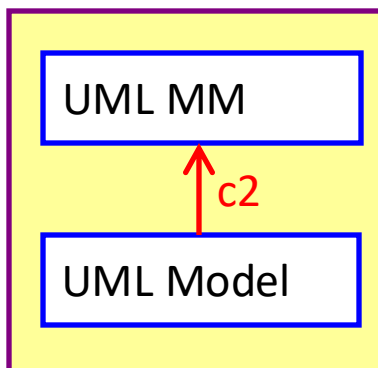
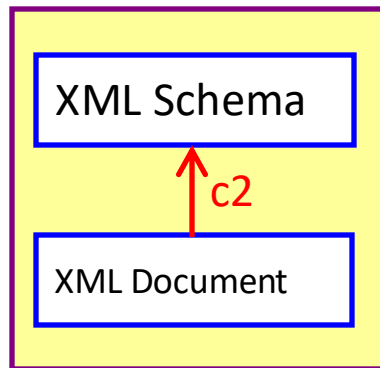
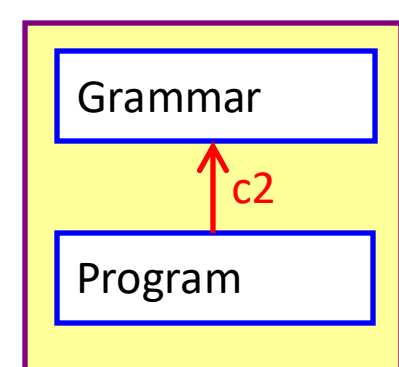
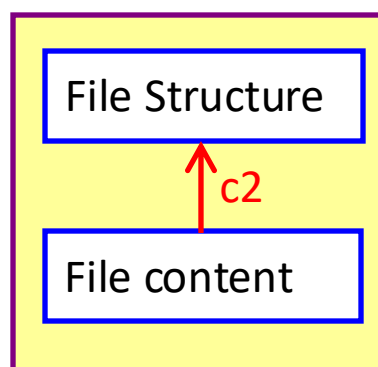
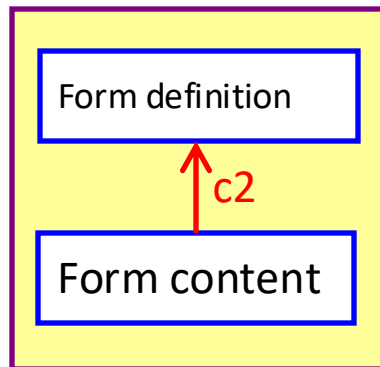
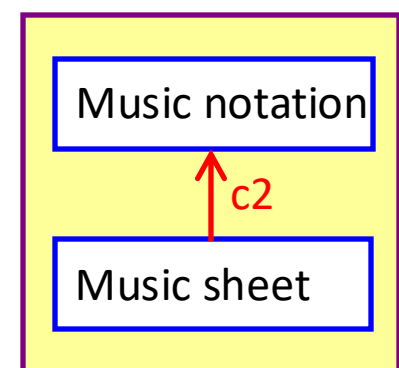
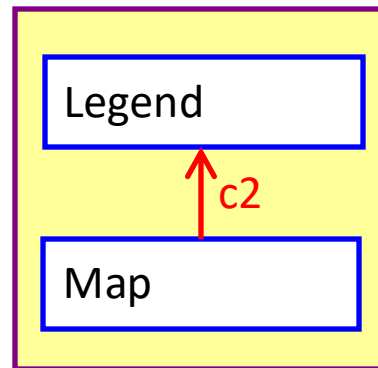
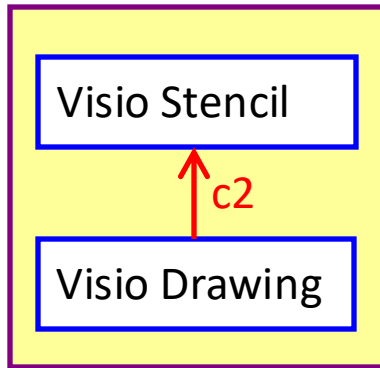
# System– Model - MetaModel



## Another example (MS Visio)

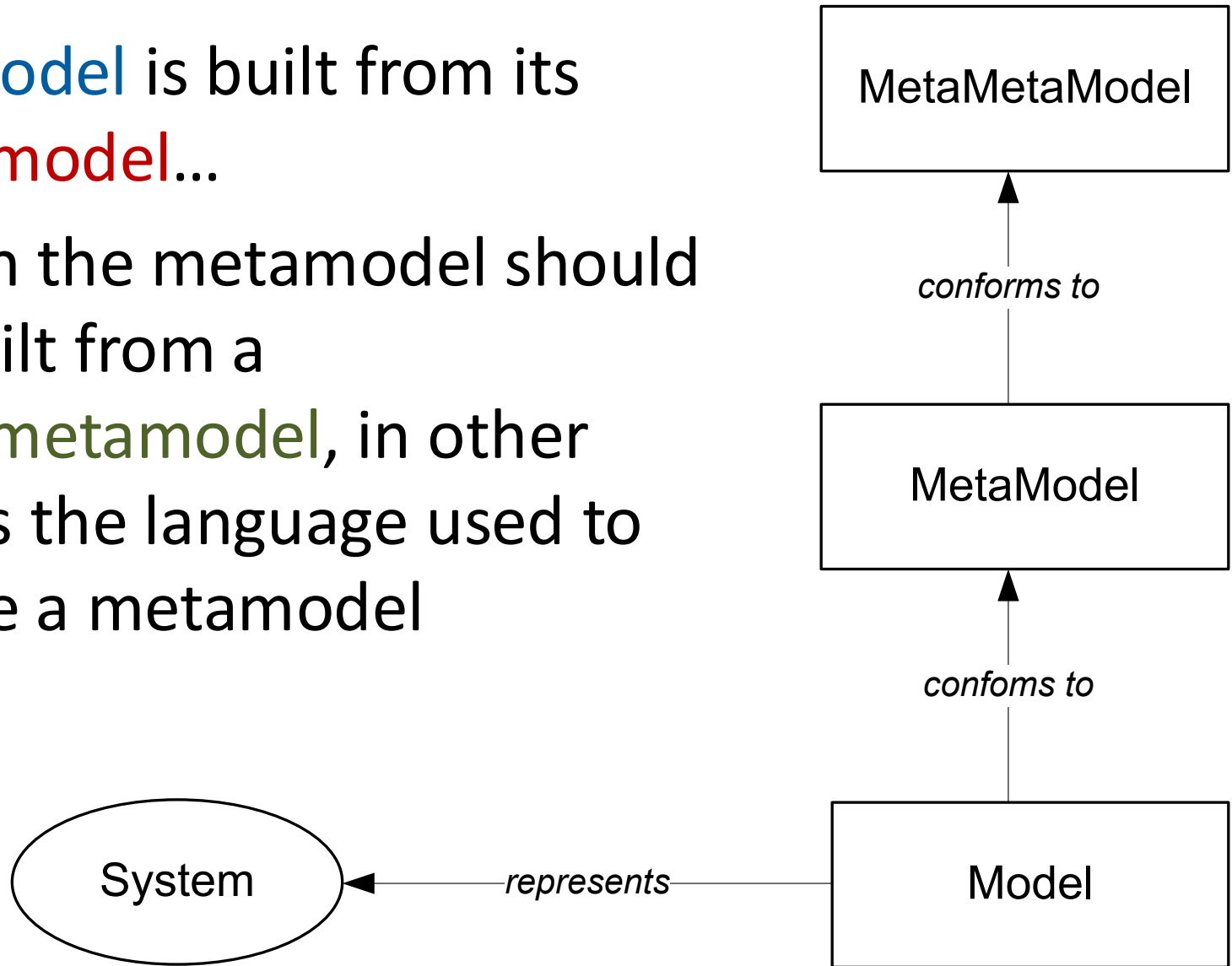


# Additional examples...



# How to define a metamodel?

- If a **model** is built from its **metamodel**...
- ...then the metamodel should be built from a **metametamodel**, in other words the language used to define a metamodel



# The metamodel is a model itself...

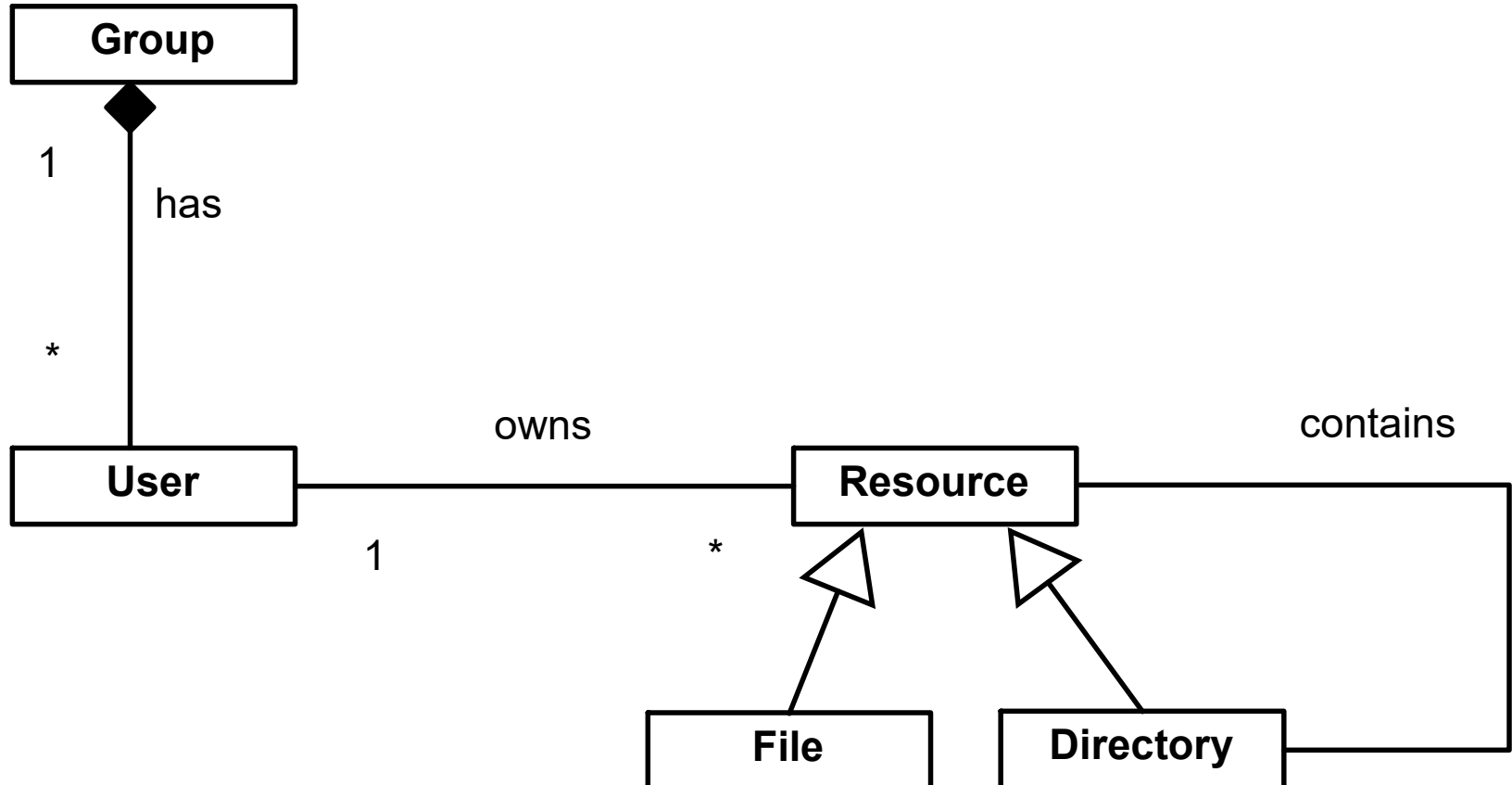
- ...and then has a metamodel, that is a **metametamodel**
- How to stop such a *meta*- $\infty$  iteration?
- Let's reason about that by use of an example...

# From the system...

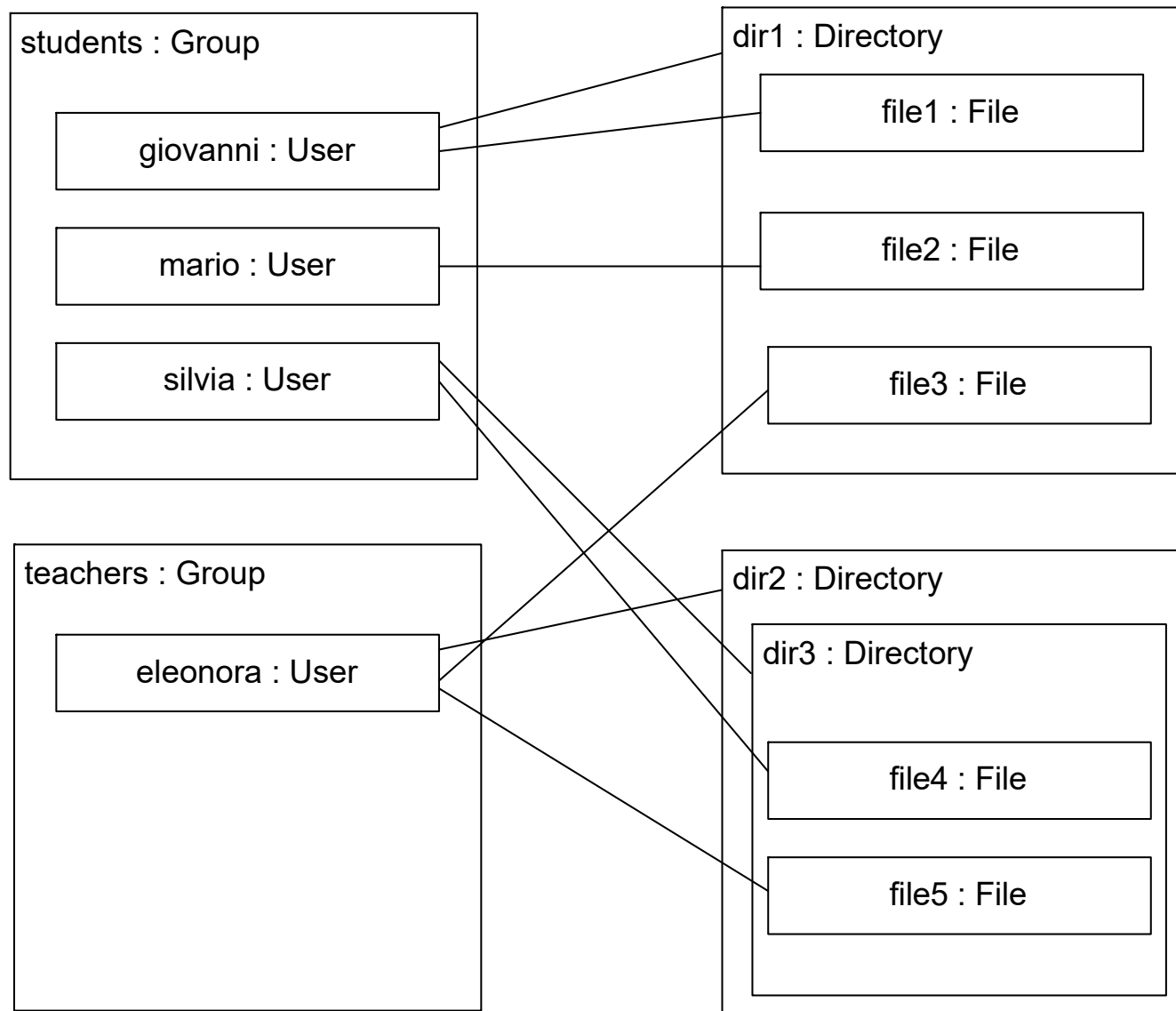
- A group of users (students and teachers) who manage a set of resources (files and directories) of a PC
- We would like to represent students, teachers, resources and the administration rights of each user



...by using the metamodel (visual notation)...



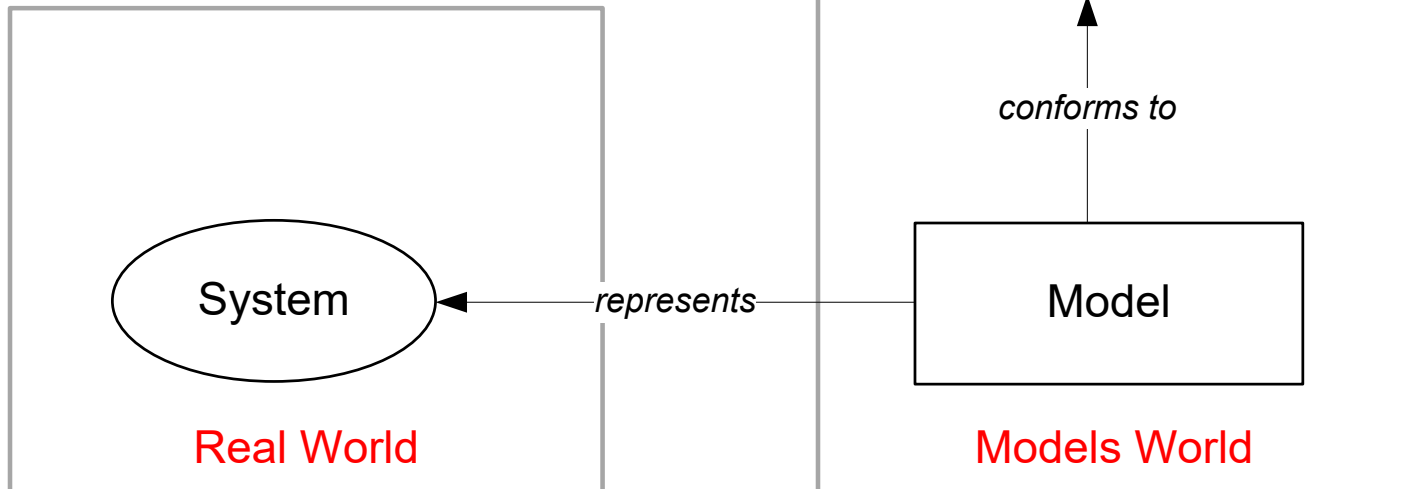
# ...to a model (visual notation)



# Metametamodel for the example

- Which is the language (*metametamodel*) used to define the metamodel
  - a simplified version of a *class diagram*
- May we use such a language to define all possible metamodels?
  - so as to guarantee uniformity of interpretation for each model specified by use of a metamodel defined by a single metametamodel?
- Good...but what is the language to build the metametamodel?
  - it is the metametamodel itself!

# The complete picture



# 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  
GMF**

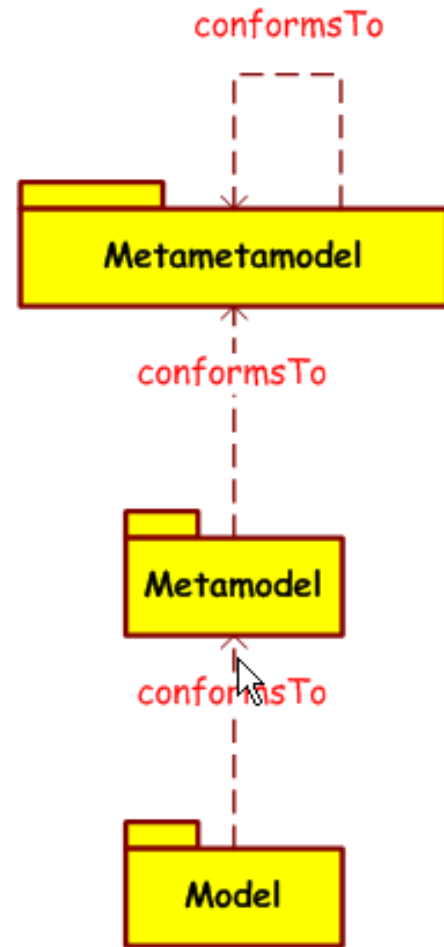
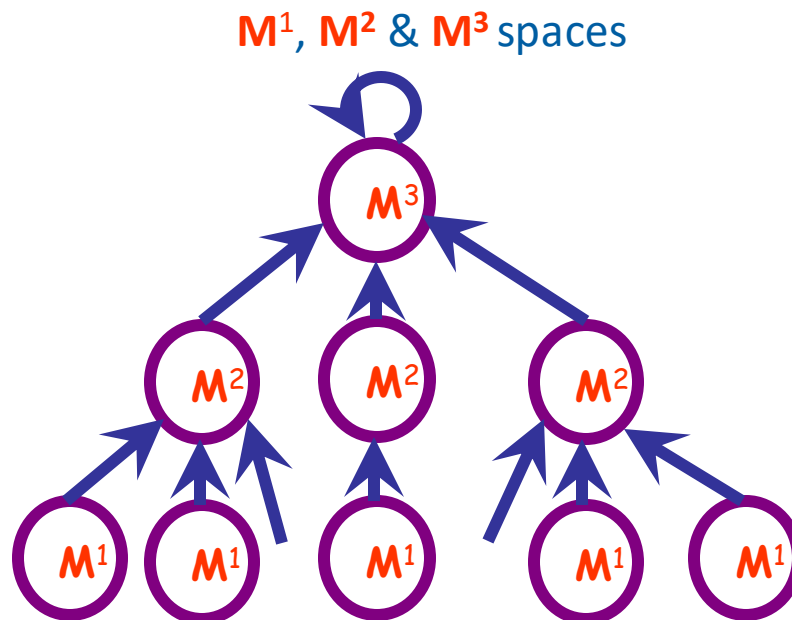
**GME**

**Microsoft  
Visual Studio  
Team system  
DSL Tools**

**Other  
Tools**

# MDA in a nutshell

- A metamodel (MOF)
- A library of MOF-based metamodels that define *domain-specific languages*
- Languages to define model transformations (*QVT*, *MOF2Text*)



# Some informal definitions of 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.

- Chris Raistrick et al., “Model Driven Architecture with Executable UML”

A formal representation of the function, behavior, and structure of the system we are considering, expressed in an unambiguous language.

- J. Bézivin & O. Gerbé, “Towards a Precise Definition of the OMG/MDA Framework”

A simplification of a system built with an intended goal in mind; The model should be able to answer questions in place of the actual system.

- ✓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”

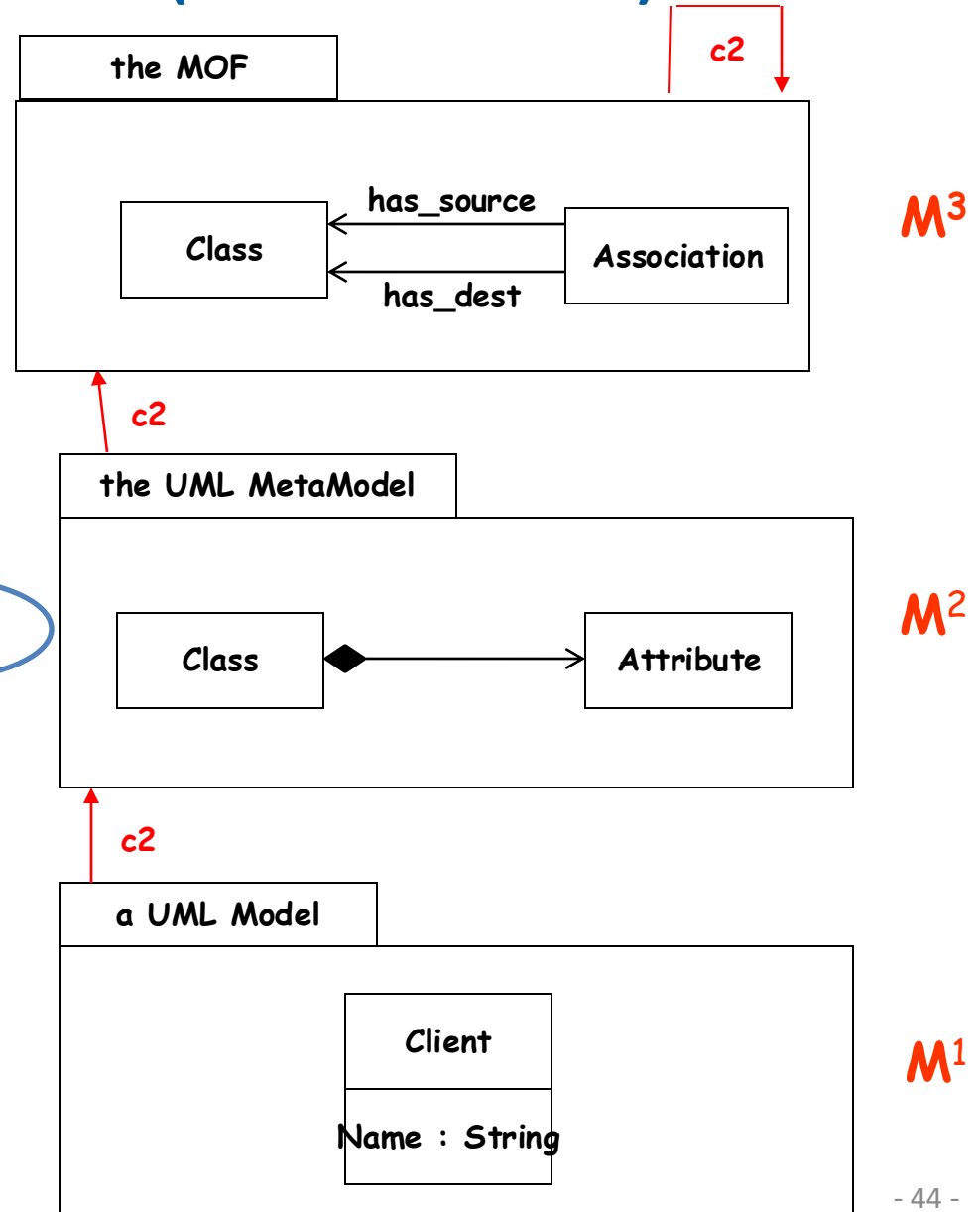
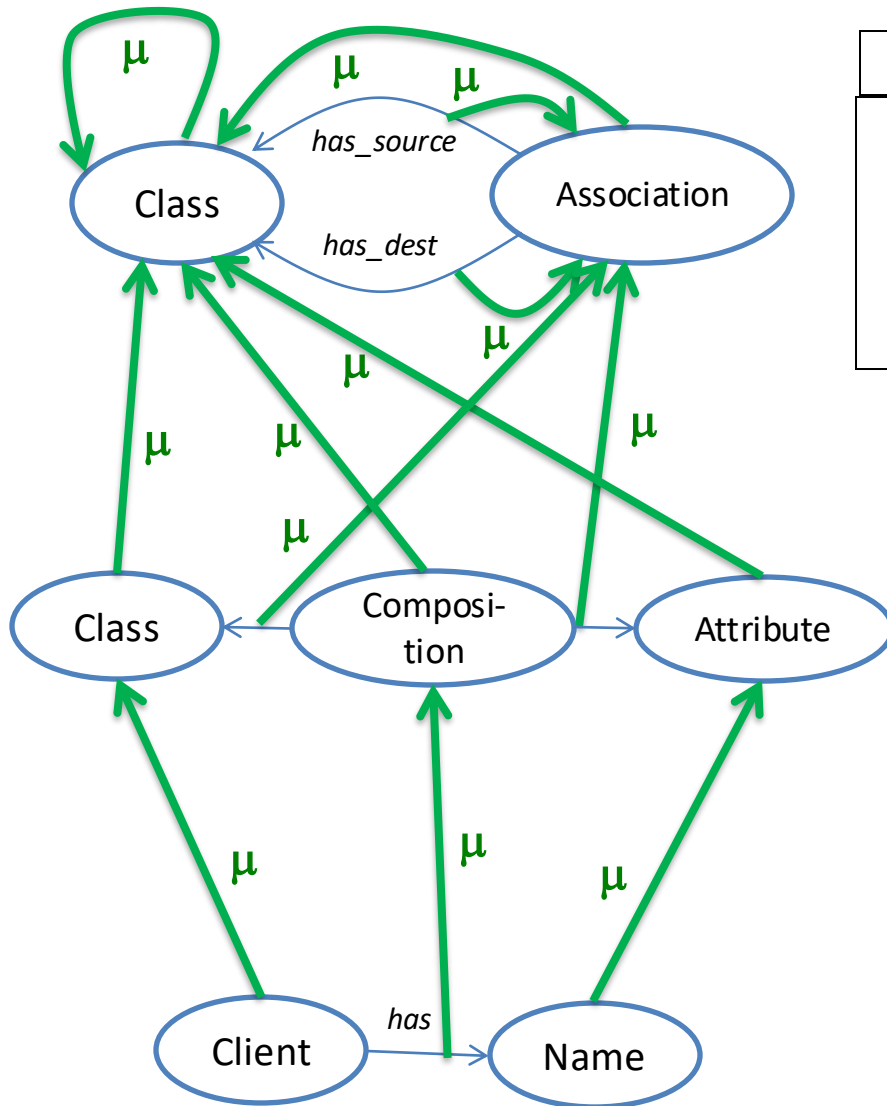
# Formal (*structural*) definition of model

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

**Definition 2.** A **model**  $M = (G, \omega, \mu)$  is a triple where:

- $G = (N_G, E_G, f_G)$  is a directed multigraph
- $\omega$  is itself a model, called the reference model of  $M$ , associated to a graph  $G_\omega = (N_\omega, E_\omega, f_\omega)$
- $\mu: N_G \cup E_G \rightarrow N_\omega$  is a function associating elements (nodes and edges) of  $G$  to nodes of  $G_\omega$  (metaElements)

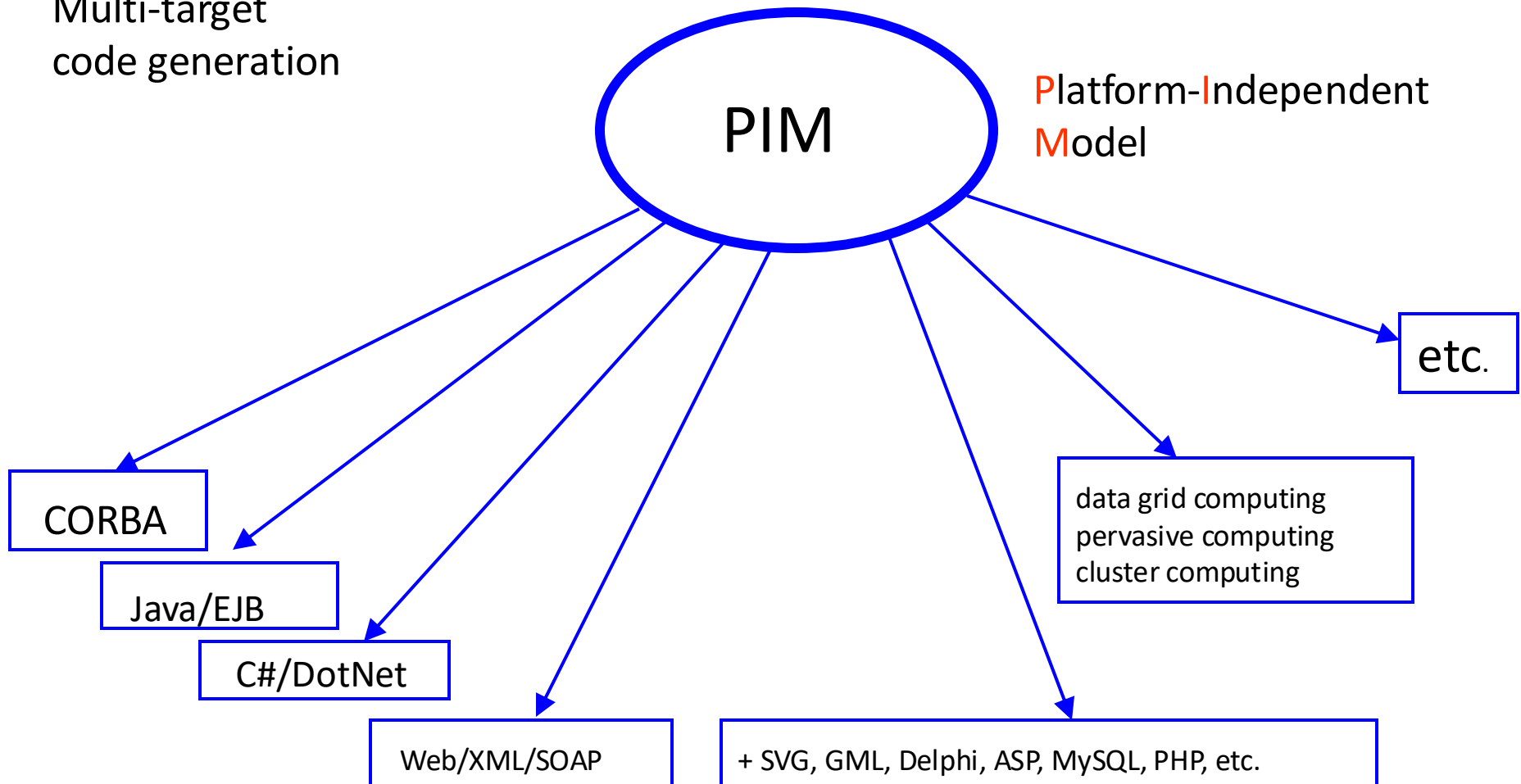
# The UML stack (revisited...)



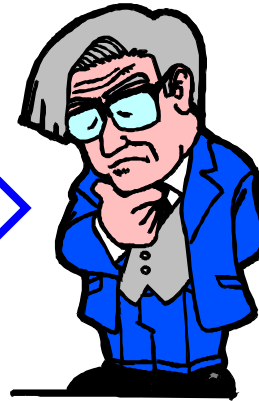
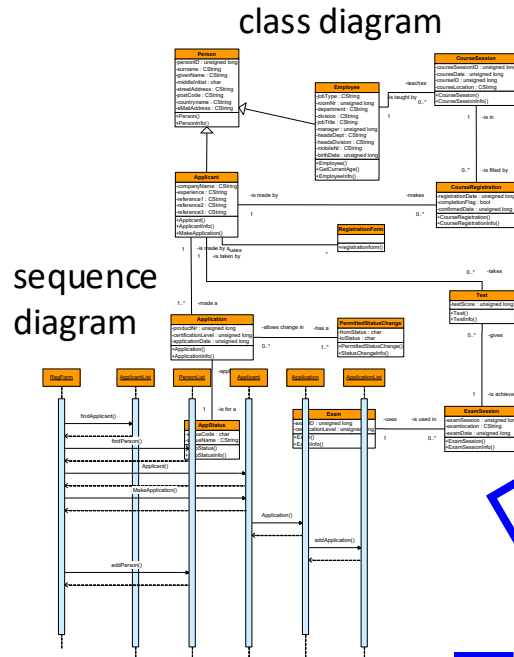
# ~~Write Once, Run Anywhere~~

## Model Once, Generate Anywhere

Multi-target  
code generation



## From contemplative to productive approaches

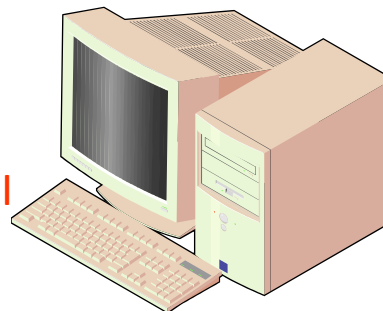


```

* @(#)Blah.java      1.82 99/03/18
*
* Copyright (c) 1994-1999 Sun Microsystems, Inc.
* 901 Sun Antonio Road, Palo Alto, California, 94303, U.S.A.
* All rights reserved.
*
* This software is the confidential and proprietary information of Sun
* Microsystems, Inc. ("Confidential Information"). You shall not
* disclose such Confidential Information and shall use it only in
* accordance with the terms of the license agreement you entered into
* with Sun.
*/
package java.blah;
import java.blah.blahdy.BlahBlah;
/*
 * Class description goes here.
 */
@version      1.82 18 Mar 1999
@author       Firstname Lastname
*/
public class Blah extends SomeClass {
    /* A class implementation comment can go here. */
    /* classVar1 documentation comment */
    public static int classVar1;
    /*
     * classVar2 documentation comment that happens to be
     * more than one line long
     */
    private static Object classVar2;
    /* instanceVar1 documentation comment */
    public Object instanceVar1;
    /* instanceVar2 documentation comment */
    protected int instanceVar2;
    /* instanceVar3 documentation comment */
    private Object[] instanceVar3;
    /*
     * ...constructor Blah documentation comment...
     */
    public Blah() {
        /* ...implementation goes here... */
    }
    /*
     * ...method doSomething documentation comment...
     */
}

```

Java  
code



XM

- From human-readable to computer-understandable
- From manual model building (e.g., Design Patterns) to automated approaches (e.g., Model Transformations)