Advances in Model Driven Engineering
Achievements and challenges

Jean Bézivin
Common work with Frédéric Jouault
{Jean.Bezivin}{noSpamAt}{inria|emn}.fr
AtlanMod Team (INRIA & EMN),
Nantes, France

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

Powerpoint version of these slides available from author
Nantes is a city in Western France, near the Atlantic coast, with 750,000 inhabitants in the metropolitan area. Nantes is the most important city of Brittany and the 6th town in France.

AtlanMod
A common INRIA and EMNantes research team focusing on Model Driven Engineering

Modeling Technologies for Software Production, Operation and Evolution
Achievement and Challenges in Model Driven Engineering

1. Introduction
2. Basic Mechanisms
3. Technical Spaces
4. Applications
5. Model Taxonomy
6. Conclusions
INTRODUCTION
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.)
EMP is a MDE technical space organized on the ECORE metameta 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

Modeling at large

S \text{ repOf } M
Modeling at large
First 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”
Multiple Acronyms

- **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.
A definition of MDA

OMG/ORMSC/2004-06-01 (The OMG MDA Guide): A Definition of MDA (The following was approved unanimously 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.
IBM MDA manifesto: Three complementary ideas

1. Direct representation
2. Automation
3. Standards

MDE vs DSLS

Language Engineering
Ontology Engineering
The impossible equation

NB:
Unfortunately no much relations between the research communities of MDE and End-User programming.

USA:
90 Millions computer users;
50 Millions Spreadsheet & DB users;
12 Millions self described programmers;
3 Millions professional programmers;
A definition of MDE?

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

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

- Agile metamodelling (working with precise, open and explicit metamodels)

- Three main operations on models: create/delete, store/retrieve, transform.
Overview of Model-Driven Engineering (MDE)

Principles

Standards
- MDA™ Model-Driven Architecture (OMG)

Tools
- Eclipse
- EMF
- GMF
- OSLO
- Other Tools
- Other Standards
Influencing parties (some)

- **SME Techno-providers**
  - Research conferences
    - ICMT, MODELS, ECMODA, ACM/SAC, ICMT, ECOOP, TOOLS, ICSE, ETAPS, MOMPES
  - International projects
    - ModelWare, ModelPlex
  - Open source groups
    - Eclipse

- **Normalization organizations**
  - OMG, W3C

- **Large companies**
  - MS, IBM & consultants

- **End user companies**
  - Airbus, Boeing, Ericsson

- **CASE Tool vendors**
One observation:
The lack of grounded theories relating to models (ModelWare) is obvious, which can be seen in the road map for object-oriented models (ref...). Hence, the current modeling practice, UML, doesn’t have a grounded theoretical foundation, although it is still widely used in software development world-wide.
My cat is Model-Driven

• Everything is nice with models, but …
  – Criticisms on MDE
  – Exaggerated hype
  – Limited tooling
  – Needs to be precise on the technology
  – Needs to be precise on the applicability scope
  – OVERSELLING

BASIC CORE MECHANISMS
Systems and Models

a system $S$

repOf

a model $M$
System and Model

**Caution:** These are only plastic food models, don’t eat them.
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 representation systems.
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.
Abstract Models

Abstract model

Metamodel

conformsTo

Terminal model
Representation and Conformance

System \( \text{repOf} \) Metamodel \( \text{conformsTo} \) Model
MetaMetaModels

- Metametamodel
  - Metamodel
    - Model
Abstract Models

Abstract model

Reference model

Terminal model

MetaMetaModel

MetaModel
Transformations as Models

- Each model conforms to a metamodel.
- A transformation builds a target model \((M_b)\) from a source model \((M_a)\).
- A transformation is a model \((M_t)\) conforming to a metamodel \((M_{Mt})\).

\[ \text{MOF} \quad \text{MMa} \quad \text{QVT} \quad \text{MMb} \]

\[ \text{Ma} \quad \text{Mt} \quad \text{Mb} \]
Transformations as models

- Treating everything as a model leads not only to conceptual simplicity and regular architecture, but also to implementation efficiency.

- ATL is composed of a transformation virtual machine plus a metamodel-driven compiler.

- The transformation VM allows uniform access to model and metamodel elements.

- Three generations of VMs:
  - Procedure oriented (Wirth's P-machine)
  - Object oriented (Smalltalk bytecode, Java VM)
  - Model oriented (ATL VM, uniform access to models and model elements)
Correspondences as models (Weaving)
Correspondences as models (Weaving)
Relationships as a model: the weaving technique

- To capture relationships between model elements
- Relationships are "reified" in a **weaving model**
  - The model elements represent the relationships and the related elements
  - As any kind of model, the weaving model can be saved, stored, transformed, modified, etc.

![AMW](image)

Ma

<table>
<thead>
<tr>
<th>a₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₂</td>
</tr>
<tr>
<td>a₃</td>
</tr>
</tbody>
</table>

Weaving model

<table>
<thead>
<tr>
<th>a₁</th>
<th>r₁</th>
<th>b₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₂</td>
<td>r₂</td>
<td>b₂</td>
</tr>
<tr>
<td>a₃</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mb

<table>
<thead>
<tr>
<th>b₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₂</td>
</tr>
</tbody>
</table>

Pollution-free traceability
Petstore Application Navigability
General annotation/decoration scheme

Model Md decoration

Model Mw mapping

Model Ma original
Abstract Models

Abstract model

Reference model

Terminal model

MetaMetaModel

MetaModel

Transformation Model

WeavingModel
Assigning meanings to models?

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

Robert W. Floyd

"Assigning meanings to programs" Symposia in applied mathematics, 1965
A model of a model

An Enterprise E

repOf

a Cobol Program

apof

A UML model
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 the other 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]
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.
Naïve illustration of multimodeling

The System

Models

System repOf repOf Model

France in 1453

The cheese french map

Percentage of termite infestation in France.

Railroad map in Western France

repOf System
Simple example

MM1: Static Structure

MM2: Dynamic Behavior (event trace)
Simple example

MW : Weaving Metamodel
Megamodeling

• With megamodeling, a given model may describe a set of other models and mutual relationships between them.
• Since a megamodel is itself a model, this allows to represent deeply nested systems of systems.
Summary

- Systems
- Models
- Technical Spaces
- Abstract models
- Metamodels
- Metametamodels
- Transformations
- Correspondence (Weaving)
- Megamodels
- Multimodeling
Structural definition of a model


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

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

```
context MetaMetaModel inv: self.conformsTo = self
context MetaModel inv: self.conformsTo.oclIsKindOf(MetaMetaModel)
context TerminalModel inv: self.conformsTo.oclIsKindOf(MetaModel)
```
These definitions are compatible with OMG view

A Proposal for an MDA Foundation Model

An ORMSC White Paper
V00-02
ormsc/05-04-11

Object Reference Model SubCommittee (ORMSC )
"The MDA guide"

"MDA is an approach to system development...[that]... provides a means for using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification." [MDA Guide omg/03-06-01]
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. Figure 1 shows the relationships between these major concepts.
Utilization definition

After the language engineering part (conformsTo), we also need to cope with the ontology engineering part (representationOf).

This is more difficult.
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.
"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

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".


See also “on the origin of objects”
The "representation" relation

System and System elements (after discretisation)

Model and Model elements

Simple set interpretation of the repOf relation is probably as correct as simple set interpretation of the instanceof relation in object technology.
Where are models coming from?
TECHNICAL SPACES
**Basic entities**

**Technical Space**: a model management framework usually based on some algebraic structures (trees, graphs, hypergraphs, etc.).

**System**: a group of interacting, interrelated, or interdependent elements forming a complex whole.

**Model**: an abstract representation of a system created for a specific purpose.
XML Technical Space

Metametamodel: XML Schema for XML Schema

Metamodel: a Petri Net XML Schema

Model: an XML document

conformsTo

conformsTo

conformsTo

repO:

ψ_{alive}

ψ_{dead}
EBNF Technical Space

**Metametamodel:**
EBNF grammar of EBNF

```
productionRule := IDENT "::=" seq ";";
seq := alternative seq?;
alternative := rep ("|"alternative)?;
rep := atom ("?" | "*" | ")")?;
atom := terminal | ("seq ");
terminal := STRING | IDENT;
```

**Metamodel:**
a Petri Net Grammar

```
net := "petrinet" "{"
  place* transition* 
  arcPT* arcTP* 
"}"
place := "place" IDENT ";"
transition := "transition" IDENT ";"
arcPT := "arcPT" IDENT "->" IDENT;
arcTP := "arcTP" IDENT "->" IDENT;
```

**Model:** a string

```
petrinet {
  place P1;
  place P2;
  transition T1;
  arcPT P1 -> T1;
  arcTP T1 -> P2;
}
```
RDF Technical Space

Metametamodel: RDF Schema

Metamodel: RDF Schema

Model

conformsTo

Classical representation

P1

T1

P2

System
The influence map

- No technology is an island
- No technology is uniformly superior to others
- Technologies are active and evolving
  - Even if sometimes they may stay idle for long periods (expert systems, etc.)
- Technologies never die: they just hide in deep software layers
  - e.g. RPG, Cobol, etc.
  - Today edge cut technologies are tomorrow legacy
- Technologies are mutually influencing
## Comparing spaces

<table>
<thead>
<tr>
<th></th>
<th>XML</th>
<th>MDA</th>
<th>Grammarware</th>
<th>Ontologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executability</td>
<td>Poor</td>
<td>Poor</td>
<td>Excellent</td>
<td>Poor</td>
</tr>
<tr>
<td>Aspects</td>
<td>Good</td>
<td>Excellent</td>
<td>Poor</td>
<td>Fair</td>
</tr>
<tr>
<td>Formalization</td>
<td>Poor</td>
<td>Poor</td>
<td>Excellent</td>
<td>Fair</td>
</tr>
<tr>
<td>Specialization</td>
<td>Fair</td>
<td>Good</td>
<td>Poor</td>
<td>Fair</td>
</tr>
<tr>
<td>Modularity</td>
<td>Fair</td>
<td>Fair</td>
<td>Poor</td>
<td>Fair</td>
</tr>
<tr>
<td>Traceability</td>
<td>Good</td>
<td>Fair</td>
<td>Poor</td>
<td>Excellent</td>
</tr>
<tr>
<td>Transformability</td>
<td>Excellent</td>
<td>Fair</td>
<td>Fair</td>
<td>Fair</td>
</tr>
</tbody>
</table>

(NB: marks are indicative) + stability in time
The «Village metaphor» by Antonio Vallecillo

Bridges between Semantic Domains

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

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.
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.
APPLICATIONS
Example: UML to Java
Example: Java to Excel

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

public class SecondClass {
    public void sc_m1() {
        FirstClass a = new FirstClass();
        a.fc_m1();
    }
    public void sc_m2() {
        this.sc_m1();
    }
}
```
Example: SPL to CPL
Example: XSLT to XQuery

**XSLT**

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

**XQuery**

```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)}</emps>
```
Metamodel-driven transformation in ATL: XSLT & XQuery metamodels
Example: Bugzilla to Mantis

About 30 open source tools to choose among for bug tracking

Different data models and functionalities
Bugzilla metamodel (simplified)
Mantis metamodel (simplified)
Bug control metamodel (pivot)

Several other types of control could be added by creating new classes that inherit of the abstract class "ControlType"...
Excel-to-Bugzilla and Excel-to-Mantis ATL bridges

Harnessing the additional complexity (accidental): global model management
Example: Complex to Simple

What is a Complex System?

- CBCS: Computer-Based Complex System
  - A complex system with a significant number of hardware/software components
  - Compare with de Rosnay's biological or ecological complex systems
- A CBCS is composed of a large number of components
- A CBCS is constantly in evolution
  - Past, present, future
  - No stops when parts are added, removed or under maintenance
- A CBCS has a structure (static architecture) and a dynamic behavior
- A CBCS is composed of components that may be also CBCSs (no limit in nesting)
- A CBCS has a goal defining its purpose in the context in which it is operating
  - The goal of a CBCS is part of its metadata
- A CBCS has a heterogeneous-based engineering
- A CBCS is a distributed system
- A CBCS may not be understood by one unique human operator
- The interactions between different parts of a CBCS follow specific patterns, implicit or explicit
- Other properties of CBCS like behavior emergence
Metamodels as lenses
Example: Autosar 2.0 to Autosar 2.1

1) Two versions of similar metamodels

Scade(v1)  Scade(v2)
\[c2\]  \[c2\]
Source model  Target model

2) A set of transformations produces a weaving model between the metamodels

Scade(v1)  MMw  Scade(v2)
\[c2\]  [c2]
Source model  Transforms  Target model

3) The weaving model is translated in a model transformation

MMw  \[c2\]
\[c2\]
Scade(v1)  MMt  Scade(v2)
Source model  Transforms  Target model

4) The source model is transformed into the target model.

"Metamodel comparison" Use Case's Overview

The comparison weaving model conforms to a weaving metamodel that is an extension of the core weaving metamodel. The metamodel extension contains an Equivalent link. This link contains a similarity attribute that saves the similarity estimation between a left and a right element. This link is extended by different kinds of links, depending on the type of elements that are being compared, for example AttributeEqual and ReferenceEqual. The relations between ElementEqual and AttributeEqual links are created according to the containment relations between Classes and Attributes. For the elements that have similarity value lower than a given threshold, the Equivalent links are rewritten into NotEquivalent links.
A typical problem: Version evolution in the information system jigsaw
Example Old to New

- **MoDisco for “Model Discovery”**
- Mining legacy to discover EMF/Ecore based models
- Extraction of models from legacy systems
  - Multiple types of such legacy systems
- A generic and extensible metamodel driven approach to model discovery

![Diagram showing MoDisco process]

- Legacy Systems
  - Source code
  - Databases
  - Configuration files
- MoDisco
- Modernization helpers
  - Documentation
  - Impact analysis
  - Models
  - Viewpoints
  - Restructured code
  - Migrated code
  - Metrics
  - Norms checking

Eclipse Modeling
The Modisco Component (Model Discovery)

Raw data → Model Discovery → Model Understanding → Interpretation

MoDisco Toolbox

Transformation or Weaving Libraries

ATL eclipse
AMW eclipse
AM3 eclipse

San Sebastián, 8-11 de septiembre de 2009
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

A Metamodel for Unix Systems

Event

User

Login

Logout

Discoverer

System S

Discovery

Model M_i

Dynamic behavior of a Unix system

who, login, logout, etc

Event time

0..* 1

 conforms To

c2
# Will MDE scale up?

<table>
<thead>
<tr>
<th></th>
<th>Terminal Models</th>
<th>Metamodels</th>
<th>Others (e.g. transf.)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size</strong></td>
<td>&lt;5 000 el.</td>
<td>&lt;50 el.</td>
<td>&lt;500 rules</td>
</tr>
<tr>
<td></td>
<td>&lt;50 000 el.</td>
<td>&lt;5 000 el.</td>
<td>&lt;5 000 rules</td>
</tr>
<tr>
<td></td>
<td>&lt;500 000 el.</td>
<td>&lt;50 000 el.</td>
<td></td>
</tr>
<tr>
<td><strong>Evolutivity</strong></td>
<td>Every second</td>
<td>Every second</td>
<td>Every second</td>
</tr>
<tr>
<td></td>
<td>Every minute</td>
<td>Every minute</td>
<td>Every minute</td>
</tr>
<tr>
<td></td>
<td>Every hour</td>
<td>Every hour</td>
<td>Every hour</td>
</tr>
<tr>
<td></td>
<td>Every day</td>
<td>Every day</td>
<td>Every day</td>
</tr>
<tr>
<td></td>
<td>Every year</td>
<td>Every year</td>
<td>Every year</td>
</tr>
<tr>
<td></td>
<td>Never</td>
<td>Never</td>
<td>Never</td>
</tr>
<tr>
<td><strong>Heterogeneity</strong></td>
<td>XMI, native data, etc.</td>
<td>MOF, DSL, KM3, etc.</td>
<td>QVT, Viatra, ATL, XSLT, etc.</td>
</tr>
<tr>
<td><strong>Quantity</strong></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>1 000</td>
<td>1 000</td>
<td>1 000</td>
</tr>
<tr>
<td></td>
<td>10 000</td>
<td>10 000</td>
<td>10 000</td>
</tr>
</tbody>
</table>
MODEL TAXONOMY
Process and Product Models

Who's doing what, when, how and why?
Static and Dynamic Systems/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 (rare)**
  - Static system: Census results
  - Dynamic model: Simulation program
**Code and Data Models**

- Generating code from hand-coded models
- Generating models from legacy code (code as data)
- Models of data

![Diagram](image)
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 others of a wonderful discovery he made while twittering in the weekend: a very powerful tool to generate UML diagrams from UML code.
- The decision was rapidly taken and all three of them 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.
MDE is a unique chance to achieve the goal of separation of the what and of the how, for example in the educational context.
CONCLUSIONS
Main messages of the presentation

- What is a model?
  - A model is a representation of a system
  - A model is written in the language of its metamodel
  - A metamodel is written in the language of its metametamodel
  - A model is a typed graph

- Where do models come from?

- What are the various kinds of models?
MDE fulfilling the promises?

But 100% of what?
Of code generation?
But then of code generation from which model? Design? Requirements?
If MDE is the solution, then what is the problem?
We still need to make precise the MDE promises.
Main messages of the presentation

- 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 various variants are changing the landscape of software engineering, system engineering and data engineering.

- Foundations are important: Principles are beginning to be consensually identified (relations of representation and conformance)

- MDE has evolved a lot since its inception in 2000
  - PSM generation (Java) from a PIM (UML)
  - Generating tests or other software artifacts (conf. files)
  - Separation and combination of aspects
  - Interoperability (tool, data, enterprise, etc.)
  - Using MDE to manage complex systems: a big challenge
Main messages of the presentation

- **MDE is about working with a lot of open and explicit metamodels (precise metamodeling).**

- **Model transformation, model weaving and global model management are the typical facilities needed in a practical MDE framework.**

- **Three levels of complexity**
  - ✓ $S \leftarrow M$ (MD Software Development)
  - ✓ $S \Rightarrow M$ (MD Reverse Engineering)
  - ✓ $S \leftrightarrow M$ (Run Time Modeling)
    - External reflection
    - Model-Based introspection and intercession
    - Descriptive/Predictive/Prescriptive
Main messages of the presentation

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

- Model Driven Testing
- Model Driven Validation and Verification
- Model Driven Requirement Engineering
- Model Driven Web Engineering
- Model Driven Process Engineering
- Model Driven Interoperability
- Model Driven Integration of Product Lines
- Model Driven Reusability
- Model Driven Traceability
- Model Driven Data Stream Processing
- Model Driven System Architecture
- Model Driven Data Engineering
- Model Driven Reverse Engineering
- Model Driven Software Modernization
- Model Driven Software Evolution
- Model Driven System Administration
- etc.
The “Towers of Models” Grand Challenge
(Robin Milner)

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.
The importance of teaching

"Teaching reduces the gap and research increases it again"

(C.A.R. Hoare, ICSE-18)
The MDE Diploma

- **Module 1: Prerequisites (60h)**
  - Software Development with Eclipse
  - Free and Open Source Models for software development
  - Software Modeling

- **Module 2: Fundamentals (120h)**
  - Fundamentals of Metamodelling and DSLs
  - Theory and Practice of Model Transformation
  - Advanced Model Management: repositories & collaborative development
  - Basic Model Driven Software Development

- **Module 3: Applications of MDE (120h)**
  - Information Systems
  - Embedded Systems
  - Data Engineering
  - Web Engineering
  - Graphical User Interfaces
  - Legacy Reverse Engineering
  - Process Engineering
  - System Engineering

- **Module 4: Management (60h)**
  - Management of MDE Projects
  - Alignment of Business Needs with Technical Platforms
  - Cartography of Information Systems
  - Strategies for Information System Evolution and Modernization
  - Human and Organizational Factors in Transitioning from Previous Technologies

- **Module 5: Internship (6 months)**
  - A co-op stay in a company or in a lab to work on a MDE project.

---

- **Move from object technology to model engineering:**
  - Probably much more difficult than the migration from procedural technology to object technology in the 80's
Thanks

✓ Questions?
✓ Comments?

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

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