Model Engineering: From Principles to Platforms

Jean Bézivin

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

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

The industrial evolution (OMG, IBM, Microsoft)
"OMG is in the ideal position to provide the model-based standards that are necessary to extend integration beyond the middleware approach... Now is the time to put this plan into effect. Now is the time for the Model Driven Architecture."

Richard Soley and the OMG staff,
MDA Whitepaper Draft 3.2
November 27, 2000
Write Once, Run Anywhere
Model Once, Generate Anywhere

Model Engineering

- Multi-target code generation
- Platform-Independent Model vs. Platform-Specific Model
- CORBA
- Java/EJB
- C#/DotNet
- Web/XML/SOAP
- SMIL/Flash
- Data grid computing
- Service based computing
- Pervasive computing
- Cluster computing
- P2P computing
- + SVG, GML, Delphi, ASP, MySQL, PHP, etc.

etc.

© 2005 Jean Bézivin
Model Engineering: From Principles to Platforms, March 2005, Vienna
But also backward (Architecture-driven Modernization)

Platforms of the past
- Legacy, Cobol, ADA, etc.

Platforms of the present
- Java, EJB, J2EE, etc.

Platforms of the future
- Grid, Cluster, P2P architectures

PIM

ADM

MDA
Suggests that MDA proponents fall into the following three camps:

1. The UML PIM camp: MDA involves the use of UML to build Platform Independent Models (PIMs) which are transformed into Platform Specific Models (PSMs) from which code is generated.

2. The MOF camp: MDA does not involve the use of UML, but instead the crucial technology is MOF, and the definition of modelling languages and language transformations using MOF.

3. The Executable UML camp: MDA involves building a UML compiler, making it a first class programming language.

Ref: Steve Cook Blog @: http://blogs.msdn.com/stevecook
Coordinated metamodels (conformance)

The MOF — The UML metamodel — Some UML Models

Various usages of these models
The metamodelling stack

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

\[ \text{M}_0 \rightarrow \text{M}_1 \rightarrow \text{M}_2 \rightarrow \text{M}_3 \]

- The MOF (some kind of "representation ontology")
- The UML metamodel
- Some UML Models
- Various usages of these models
- "the real world"

and other MMs
and other Ms
MDA general organization

Analysis M. -> UML -> MOF

Design M. -> ????

Code M. -> Java

Simplified vision of the world

- Analysis M.
- Design M.
- Code M.
- EIFFEL
General case

**PIM**
- **Entreprise model** → **MM1**

**PSM**
- **System model** → **MM2**

**PDM**
- **Platform model** → **MM3**

© 2005 Jean Bézivin
Model Engineering: From Principles to Platforms, March 2005, Vienna
Model Engineering

PIM=EDOC; PSM=WSDL
EDOC meta-model

Mapping

WSDL meta-model

Source association composition

© 2005 Jean Bézivin
Model Engineering: From Principles to Platforms, March 2005, Vienna - 13 -
The initial Y conjecture: \( \text{PSM} = f(\text{PIM}, \text{PDM}) \)

\( \text{PSM} = f(\text{PIM}, \text{PDM}) \) or by currying: 
\( \text{PSM} = f_{\text{PDM}}(\text{PIM}) \)

PIMs (Platform Independent Models)

PSMs (Platform Specific Models)

PDMs (Platform Description Models)

Merging phase

binding

M
Abstract models of Platforms and Enterprises

- **MMe**: A model of enterprise E
- **MMs**: A model of system S
- **MMp**: A model of platform P

- enterprise E
- system S
- platform P
**MDA in a nutshell**

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

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

• MDA is an OMG initiative that proposes to define a set of non-proprietary standards that will specify interoperable technologies with which to realize model-driven development with automated transformations. Not all of these technologies will directly concern the transformations involved in MDA.

• MDA does not necessarily rely on the UML, but, as a specialized kind of MDD (Model Driven Development), MDA necessarily involves the use of model(s) in development, which entails that at least one modeling language must be used.

• Any modeling language used in MDA must be described in terms of the MOF language, to enable the metadata to be understood in a standard manner, which is a precondition for any ability to perform automated transformations.
"MDA™ is a specific MDD™ deployment effort around such industrial standards as MOF, UML, CWM, QVT, etc." (from OMG/MDA guide).

Metamodels  DSLs

Model-Driven Engineering (MDE)

MDA™
Model-Driven Architecture (OMG)

MIC
Model Integrated Computing

MS:
Whitehorse

Other subfields of MDE

Common set of principles

Terminological note: MDA and MDD are trademarks of OMG; MDE is not.
Principles, standards and tools

Model-Driven Engineering (MDE)

Principles
- MDA™ Model-Driven Architecture (OMG)
- MIC Model Integrated Computing

Standards
- Software Factories (MS)
- Other Standards

Tools
- AMMA
- Eclipse
- EMF
- GME
- Microsoft Visual Studio Team system
- Other Tools
EMF: The ECORE metamodel

http://www.eclipse.org/emf/
A very active industrial development area

**Principles**

*Model-Driven Engineering (MDE)*

**Standards**

- MDA™ Model-Driven Architecture (OMG)
- MIC Model Integrated Computing
- Software Factories (MS)
- Other Standards

**Tools**

- Eclipse EMF
- GME
- Microsoft Whitehorse Visual studio Support for DBs
- Other Tools
The mission of the Eclipse Technology Project is to provide new channels for open source developers, researchers, academics and educators to participate in the on-going evolution of Eclipse. It is organized as three related project streams, namely Research, Incubators and Education. Research projects explore research issues in Eclipse-relevant domains such as programming languages, tools, and development environments. Incubators are small, informally structured projects which add new capabilities to the Eclipse software base. Education projects focus on the development of educational materials, teaching aids and courseware.
The goal of the Eclipse Community Education Project (ECESIS) is to promote the creation, improvement and distribution of commercial and academic quality Eclipse courseware, education and training technologies, and resource material. ECESIS’s basic objectives are as follows:

- To develop a starting set of high quality Eclipse courseware, tools, and non-course educational resources
- To distribute freely all educational material under an open source license
- To encourage the production and development of further material
- To encourage and promote the widest possible use of such material
The goal of this technology project is to construct/assemble a set of tools for the Eclipse platform which support model driven software development with fully customisable Platform Independent Models, Platform Description Models, Texture Mappings, and Refinement Transformations.
Eclipsecon’2005
Big Announcements from the Eclipse Foundation

- Four new Strategic Developer partners: BEA, Scapa, Sybase, and Borland.
  To become a Strategic Developer, a company pays yearly dues of 250,000, occupies a seat on the board, leads products, and devotes developers to Eclipse projects.

Eclipse announcement from IBM

- The Model Transformation Framework (MTF) -- a set of tools to help developers make comparisons, check consistency and implement transformations between Eclipse Modeling Framework (EMF) models.

- The Emfatic Language for EMF Development -- an Eclipse Modeling Framework which models in textual form. Emfatic offers a very simple and direct way to create and edit Ecore models with its syntax very intuitive to programmers familiar with Java.

Sybase

Sybase is proposing a new Data Tools Project at Eclipse. The goal of the project is to work with the Eclipse community in developing a comprehensive data management tooling framework.

Borland

As part of this increased investment, Borland will expand its use of Eclipse as a platform across its Application Lifecycle Management (ALM) product line. In an effort to drive ALM advancements on the Eclipse platform, Borland will also lead the proposal for a new graphical modeling framework that would build upon, bridge and extend the existing modeling technology within the Eclipse community.

Modeling continues to gain traction as enterprises look to reduce IT complexity, improve team efficiency and collaboration and more closely connect IT and business requirements. It is an important element of Borland’s Software Delivery Optimization strategy and Borland will make a significant investment in expanding Eclipse to better address this area.
iMDD (integrated Model Driven Development)

- The Eclipse Integrated Model Driven Development (iMDD) project is dedicated to the realization of a platform offering facilities needed for applying a Model Driven Development (MDD) approach. This platform will provide a consistent set of tools for modeling Domain Specific Languages (DSL) and for generating a dedicated environment (GUI, checks, generators, compilers, simulators...) for these specific DSL.
- The Eclipse platform is an excellent basis on which to build and integrate Model Driven Development tools. The iMDD platform will extend the Eclipse platform with an additional level of integration for MDD tools providing and requiring services that operate over models or model elements.

GMF (Graphical Modeling Framework) (GMF =

- Borland is taking a bigger stake in the organization it helped found, joining the Eclipse Foundation's board of directors as a strategic developer, officials will announce in a press conference Monday at EclipseCon 2005 in Burlingame, California.
- Raaj Shinde, Borland vice president of product strategy and architecture, said the proposal, which hasn't been formally submitted to the Eclipse Foundation, builds on an area they have some expertise with, he said, as Together has been running on Eclipse since 2002. The project has a good chance of looking like a port of its commercial product.
- Skip McGaughey, an Eclipse spokesman, said Borland's installment on the board of directors will add significant expertise and energy to the Eclipse Foundation, and doesn't expect the company to have any difficulties getting its project accepted within the organization. It's too early to tell, he said, whether the project will remain a sub-project or become a top-level project down the road.
Microsoft Whitehorse, etc.

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

... Modeling is the future ...

Bill Gates
... Modeling is the future ...
You know UML [Unified Modeling Language] made the meta-models a little complex, so I don't think UML alone is the answer ...
And the promise here is that you write a lot less code, that you have a model of the business process. And you just look at that visually and say here is how I want to customize it ...
So even a business could express in a formal, modeled way, not just scribbling on paper, how the business process is changing over time or how it's different from other companies. So instead of having lots of code behind that, you just have visual, essentially model, customization ...
So, I think we believe that. There are certainly some people from IBM who have that same vision, and I think it'll be healthy competition between the two of us because today's modeling products fall short. That's one part of Visual Studio 2005, that we do have some neat things coming along that will be part of it that we haven't shown completely ...
So, modeling is pretty magic stuff, whether it's management problems or business customization problems or work-flow problems, visual modeling. Even the Office group now really gets that for document life-cycle rights management, that this visual modeling will be key to them. Business intelligence, where you let people navigate through things, is another area where modeling could be used. It's probably the biggest thing going on. And both Visual Studio and Office need to be on top of that ...
• Microsoft is releasing a suite of tools to make it easy to construct graphical designers hosted in Visual Studio for editing domain specific languages (DSL).
IBM on MDA: Three complementary ideas

1. Direct representation
2. Automation
3. Standards

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

The model used to represent models in EMF is called Ecore. Ecore is itself an EMF model, and thus is its own metamodel. You could say that makes it also a meta-metamodel. People often get confused when talking about meta-metamodels (metamodels in general, for that matter), but the concept is actually quite simple. **A metamodel is simply the model of a model**, and if that model is itself a metamodel, then the metamodel is in fact a meta-metamodel.4

Got it? If not, I wouldn’t worry about it, since it’s really just an academic issue anyway.

4. This concept can recurse into meta-meta-metamodels, and so on, but we won’t go there.
Enter the "metamuddle"

A metamodel is a model of a model. The concept can be recursively applied to itself, e.g., a meta-metamodel is a model of a metamodel and so on. A metamodel is a reusable process component, the notion of a model of a model is important in the field of Model Management Systems (MMS). As illustrated in Meta Object Facility (MOF) [2], as a language for specification, as a model of model defining some aspects of the structure of the model, the metamodel itself is formulated re-using the language it describes.

Diagrams, similar to UML models, may be described by means of a mechanism such as the term "metamodel" here is used in the same way the UML does it. A metamodel is a key facility in this new era, it provides an abstraction, it is often called "a model of a model". It provides a rules/grammar for the modelling language (ML). The ML consists of instances of concepts in the metamodel.

Glossary

- Metamodel: model of a model
- Metaclass: an element of a metamodel
- Metaobject: a model of a model
- Metaclass: a class which represents an element of a metamodel. Every component in a UML model is an instance of this metaclass. Metaclasses are defined using meta-attributes, meta-

© 2005 Jean Bézivin

Model Engineering: From Principles to Platforms, March 2005, Vienna
The standard "official" OMG stack (mimicking OT)

- **M3 Layer**: Specifies meta-metaclasses for the UML metamodel.
- **M2 Layer**: Specifies metaclasses for the UML metamodel, such as Class.
- **M1 Layer**: Specifies classes for the UML user models, such as Passenger, Ticket, TravelAgency.
- **M0 Layer**: User objects that are instances of UML user model classes, such as instances of Passenger, Ticket, TravelAgency.
The metamuddle

- A very rapidly growing industrial application field since November 2000,

- ... but ...

- We badly need a unifying theory of models
In search of sound principles for MDE
The basic assumptions

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

Model Engineering

Language engineering

Ontology engineering

MDE

MetaModel

conformsTo

Model

isRepresentedBy

System

The modelling world

The real world

© 2005 Jean Bézivin

Model Engineering: From Principles to Platforms, March 2005, Vienna
Definitions

• **A model** is the simplified **image of a system**
  • This short definition should be completed

• **What is a system?**
  • "A system is a set of elements in interaction" (von Bertalanffy)
  • The word system comes from the Greek "sun-istémi" (I compose)

• **Model** comes from the Latin "modullus", diminutive of "modus" (measure)
  • Initially it was an architectural term meaning an arbitrary measure used for establishing various ratios between different parts of a building in construction.

• Two importations in **English**:

The word is recent, the idea is old

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

This idea will be used again later by the English physiologist William Harvey (1578–1657) who will discover the blood circulation principle: "de ce que, dans le cœur des vivants, les valvules semblent être des soupapes ou des portes d'écluse".
<table>
<thead>
<tr>
<th><strong>mod·el</strong></th>
<th><strong>noun (plural mod·els)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. copy of an object:</td>
<td>a copy of an object, especially one made on a smaller scale than the original (often used before a noun)</td>
</tr>
<tr>
<td>2. particular version of manufactured article:</td>
<td>a particular version of a manufactured article</td>
</tr>
<tr>
<td>3. something copied:</td>
<td>something that is copied or used as the basis for a related idea, process, or system</td>
</tr>
<tr>
<td>4. somebody paid to wear clothes:</td>
<td>somebody who is paid to wear clothes and demonstrate merchandise as a profession, for example, in fashion shows and photographs for magazines and catalogues</td>
</tr>
<tr>
<td>5. simplified version:</td>
<td>a simplified version of something complex used, for example, to analyze and solve problems or make predictions a financial model</td>
</tr>
<tr>
<td>6. perfect example:</td>
<td>an excellent example that deserves to be imitated</td>
</tr>
<tr>
<td>7. artist’s subject:</td>
<td>somebody who poses for a painter, sculptor, photographer, or other artist</td>
</tr>
<tr>
<td>8. zoology animal copied by another animal:</td>
<td>an animal species repellent to predators which another animal mimics for protection</td>
</tr>
<tr>
<td>9. logic interpretation:</td>
<td>an interpretation of a theory arrived at by assigning referents in such a way as to make the theory true</td>
</tr>
<tr>
<td>10. U.K. fashion exclusive garment:</td>
<td>the first sewn example of a couturier’s or clothing manufacturer’s design, from which a new line of garments is produced</td>
</tr>
</tbody>
</table>
Differents kinds of models

- Numerous examples
  - Mathematical models
  - Hydrological models
  - Biological models
  - Ecological models
  - Economical models
  - Meteorological models
  - Simulation models
  - Descriptive or predictive models
  - etc.

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

A system

Several models of this system (partial views)

Don't confuse the model and the system

This is not a pipe by Magritte

Ceci n'est pas une pipe.
Don't confuse the model and the system

The system

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]
The globe is a model of the earth

System
+ask()

repOf

Model
+ask()
A very popular model: geographical maps

- France in 1453
- The cheese french map
- Percentage of termite infestation in France:
  - de 75 à 100
  - de 50 à 75%
  - de 25 à 50%
  - de 10 à 25%

The System

- Railroad map in Western France

- Models

System  repOf  Model
Every map has a legend (implicit or explicit)

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

The legend is the metamodel
A model has no meaning when separated from its metamodel.

First round of political election in France in 2002.

Percentage of places infested by termites in France.
Lewis Carroll and the 1:1 map

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

“About six inches to the mile.”

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

“Have you used it much?” I enquired.

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

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

See also J.-L. Borges, J. François, and more recently Umberto Eco.
Lewis Carroll and the blank map

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

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

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

THE HUNTING OF THE SNARK
an Agony in Eight Fits by Lewis Carroll
A "lattice" of metamodels

The system

Nothing: Lewis CARROLL
white map

Top

A model

Everything: Lewis CARROLL
1:1 map

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

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

Paraphrasing Tony Hoare

[Compare to UML 2.0 approach]
Research agenda: Everything is a model

- What is a model?
  - A model is a representation of a system
  - A model is written in the language of its unique metamodel
  - A metamodel is written in the language of its unique metametamodel
    - The unique MMM of the MDA is the MOF
  - A model is a constrained directed labeled graph
  - A model may have a visual representation

- Where do models come from?

- What are the various kinds of models?
A classification of explicit models (incomplete)
Various kinds of models

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

```
System
  ask()

Model
  ask()
```

```
Dynamic System
StaticSystem
Dynamic Model
StaticModel
```
MDA proposed R&D Agenda: "Everything is a model ..."

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

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

A given system may have plenty of different models. Each model represents a given aspect of the system.
Transformations as models
Uniform access to models and metamodels

MMa
Ma

MMt
Mt

MMb
Mb

Transformation Engine
**ATL: a model transformation language, engine and IDE**

- ATL: a MOF/QVT compliant model transformation language
- For more info see: [http://www.sciences.univ-nantes.fr/lina/atl/](http://www.sciences.univ-nantes.fr/lina/atl/)
- or [http://eclipse.org/gmt/](http://eclipse.org/gmt/)
ATL Editor with its content outline
ATL Debug perspective with the ATL Editor, Debug, Variable and Outline view
A platform is a model

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

2.2.7 Platform

Examples:

Generic platform types

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

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

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

Technology specific platform types

CORBA: An object platform that enables the remote invocation and event architectural styles. [formal-01-12-33]

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

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

Vendor specific platform types

CORBA: Iona Orbix, Borland VisiBroker, and many others

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

Microsoft .NET

Figure 2-1: A platform

Many of the illustrations in this Guide use this icon to represent a platform.
A correspondence is a model

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

Diagram:

- PIMs (Platform Independent Models)
- PSMs (Platform Specific Models)
- Merging/binding phase
- Hidden weaving model here?
- PDMs (Platform Description Models)
Assigning meanings to models

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

Robert W Floyd
"Assigning meanings to programs"
Symposia in applied mathematics, 1965
The "representation" relation

Simple set interpretation of the \texttt{repOf} relation is probably as correct as simple set interpretation of the \texttt{instanceOf} relation in object technology.
"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."

Technical spaces
Technical Spaces and Working Contexts

**Technical Spaces**
- Examples: MDE, EBNF, XML, DBMS, ontologies, etc.
- Conjecture:
  - Each TS is represented by a metametamodel
  - Each TS is organized in a 3 metalevel architecture

**Working contexts**
- Local
  - MM specific, e.g. UML
- Global
  - TS specific, MM independent, e.g. MOF
- Universal
  - Across several TSs
The notion of TS (Technology Space) as a tool for collaboration

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

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

Figure 2.2 EMF unifies Java, XML, and UML.
Abstract Syntax Systems Compared

Technology #1  (formal grammars, attribute grammars, etc.)

Technology #2  (MOF + OCL)

Technology #3  (XML Meta-Language)

Technology #4  (Ontology engineering)

M³

EBNF

M²

Pascal Language Grammar

The UML meta-Model

A XML DTD or Schema

KIF Theories

+Description Logics
+Conceptual Graphs +etc.

M¹

A specific Pascal Program

A Specific UML Model

A XML document

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

[XMI=MOF+XML+OCL]

Model serialisation

…DBMS not shown here

A specific execution of a Pascal program

A Specific phenomenon corresponding to a UML Model

A XML DTD or Schema

Representation issues

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

Fragment of the XML representation ontology
A technology space is organized around a set of concepts
Spaces may be connected via bridges
Spaces are often similarly organized

Grammar
- Syntax
- Program

Schema
- XML
- Document

Meta-Model
- MDA
- Model

Schema
- DBMS
- Data

Ontology engineering
- Ontology
- Top Level O.
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>Good</td>
<td>Good</td>
<td>Poor</td>
<td>Poor</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)
Three representations for the same program

Program TS

Java Grammar

Java Program

MDA TS

Java Meta model

Java Model

XML TS

JavaML DTD

JavaML Document
Three representations for the same program

Each of these representations may be more convenient to perform some operation on the program.
Collaborations between TSpaces: Projectors

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

- Everything is a model
  - A $\lambda$-model
  - $\lambda$ meaning the specific TS
  - An XML document is an XML-model
  - A Java source program is a Java-model
  - An UML model is a MDA-model
  - etc.

- Each TS is rooted in a metametamodel defining its representation scheme

- Distinguish between intra-space and inter-space operations
“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.”

What we learn from OT and why it is not the solution
“...”

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

“...”

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

“...”

J. Greenfield and K. Short
From objects to models

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

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

```
Super-class
  inherits

Class
  instanceof

Instance

MetaModel
  conformsTo

Model
  isRepresentedBy

System
```
Unification principle

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

Both assertions are "basic engineering postulates".

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

The same applies now to models.
Summary

- Object technology realized some promises but failed to achieve others
  - Stopping the search for generality by unification may be one of the causes for this
- Model engineering is making many promises today
  - Will it be able to deliver correspondingly?
  - Sticking with the principle that "everything is a model" seems a good way to make progresses
Towards an open MDE platform?
AMMA: A Lightweight Architectural Style for Generic Model Management Platforms

- ATLAS Model Management Architecture
- Build around a minimal set of sound principles
- Defines the conventions for the various connected tools to interoperate
- Lightweight: Not reinventing CORBA
- Model-based interoperability and not Middleware-based interoperability
- Four basic blocks:
Many groups building Open MDE platforms

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

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

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

- Fixed mapping metamodels are not sufficient
- We need support for extensible variable metamodels
The Model Weaver: first prototype
AM3: ATLAS MegaModel Management Tool

- Megamodel: a model with elements corresponding to models, metamodels, services, tools and more generally to any global resource available in the scope of an AMMA session.
- A registry for model engineering resources as well as a metadata repository
- Megamodel with different metamodels
- Megamodels beyond typing systems
Megamodel Resource Navigator

- Extension of Resource View
- According to metadata, tools may be available for an element

Megamodel Browser

- Quick access to elements
- Four views:
  - List of artifacts
  - List of transformations (MetamodelIn - Transformation - MetamodelOut)
  - List of injectors (From - Injector - MetamodelOut)
  - List of extractors (MetamodelIn - Extractor - To)
ATP: TS and projectors

XMI  JMI  CMI  etc.

ATLAS  Technical  Projectors

TSpace #1 MDA

mmodel  model

π₁

TSpace #2 EBNF

grammar  program

π₂

TSpace #3 XML

XMLschema  document

π₃

TSpace #4 SQL

dataSchema  data

Java  Corba  SVG  NLP,  etc.
Conclusions and perspectives
Conclusions

Model engineering is the future of object technology

- As object and classes were seen in the 80's as "first class entities", with libraries of several hundred of classes hierarchically organized, models and metamodels are beginning to be considered alike in the 2000's.

- Libraries of hundreds of domain specific models and metamodels of high abstraction are beginning to appear. Each such metamodel contains a number of concepts and relations.

- Tools will be needed to work with these vast libraries of models and metamodels. These tools will be much different of present CASE tools and class browsers. They will interoperate on open platforms (EMF or VisualStudio based). Many of them will be partially automatically generated from metamodels.
Conclusions

• The problems of understanding what is a good metamodel for a PIM, a PSM or a PDM may take longer to settle than initially planned. The concept of PSM itself may be questionable and is probably still ill-defined.

• The model transformation operation is beginning to be understood (QVT++). Other operations like model weaving still need research work. The global relations between models should be considered.
Conclusions

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

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

Jean Bézivin
Jean.Bezivin{noSpamAt}lina.univ-nantes.fr
Equipe ATLAS, INRIA & LINA, Nantes, France