



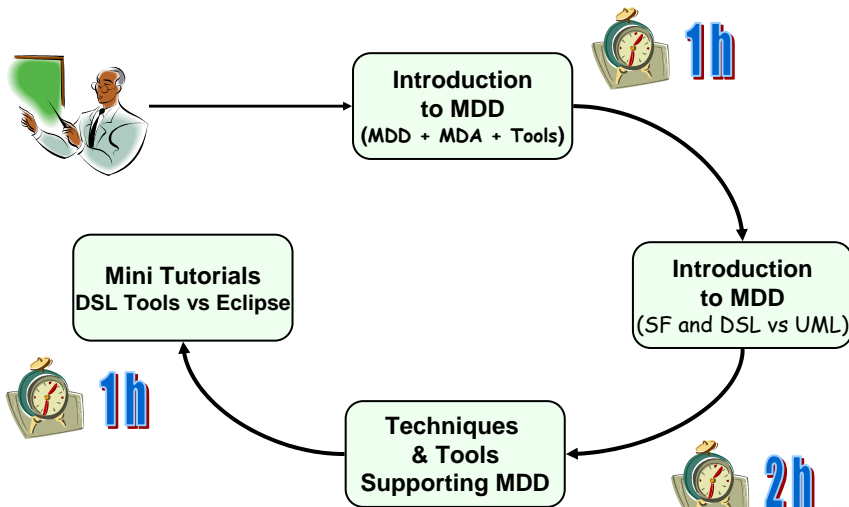
Model Driven Development

Building Automated Code Generation Methods with Eclipse and DSL Tools

Vicente Pelechano

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

Course Overview



Model Driven Development

Vicente Pelechano

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia



Contents

- Model Driven Development
- MDA
- MDD vs. MDA
- Software Factories
- MDA vs. Software Factories
- UML vs. Domain Specific Languages



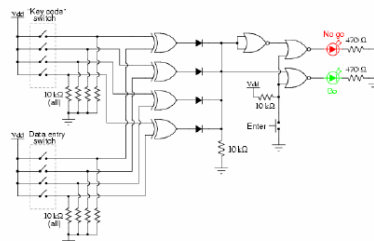
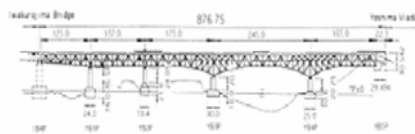
Model Driven Development

- Models in Engineering
- What are Models for?
- Characteristics of the Models
- What is a (Software) Model?
- Current Limitations of the Models
- What is the MDD?



Models in Engineering

- As old as Engineerings
- Engineers always build models before the construction of their works and artifacts





What are Models for? ...

- **Specify the System**
 - Structure, behavior,...
 - Communicate with stakeholders
- **Understand the System (if it exists)**
- **Validate and Reason about the System**
 - Identify errors and missing data and/or functionality
 - Prototyping (model execution)
 - Infer and prove properties
- **Guide the Implementation**



Characteristics of the Models

- **Abstract**
 - Emphasizing some concerns, while hiding others
- **Understandable**
 - Expressed in a language understandable by users and stakeholders
- **Precise**
 - Accurate representation of the object or modeled system
- **Predictive**
 - Should be used to infer correct conclusions
- **Cheap**
 - Easier and cheaper to build than the software system

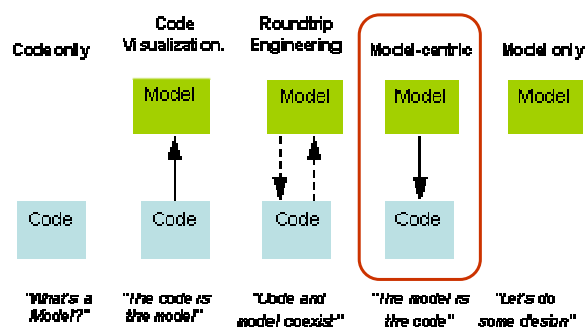


What is a (Software) Model?

- A description of (part of) a system written in a well-defined language. (Equivalent to specification.) [Kleppe, 2003]
- A representation of a part of the function, structure and/or behavior of a system [MDA, 2001]
- A description or specification of the system and its environment for some certain purpose. A model is often presented as a combination of *drawings and text*. [MDA Guide, 2003]
- A set of *statements* about the system. [Seidewitz, 2003] (Statement: expression about the system that can be considered true or false.)
- A model is 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. [Bezivin, ASE 2001]



What is a (Software) Model?





Current Limitations of the Models

- Only used as documentation
 - Most times not updated
- “Gap” between the model and the system implementation
 - Semantic differences between the respective languages
 - Absence of automatic propagation of changes
 - Changes in the Model are not reflected in the code
 - Changes in the Code are not reflected in the model
- The distinct models are not harmonized
 - Different views of the same system, but difficult to relate them
 - Lack of tools for model integration
 - Each view (language) has a semantics that is different from the rest
- Lack of tools and languages for model management
 - Only graphical or textual editors, but “compilers”, “optimizers”, “validators”, “model transformers”, etc. are missed.



What is the MDD?

- It is a software development approach where the first class citizens are the **models** and the **model transformations**.
 - ...Versus **programs** and **compilers** that were the analogous paradigm 30 years ago
- MDD implies the (almost) automatic generation of code from models.
- Languages (for modeling and also for transforming models) are key factors in the MDD.
- MDA is the proposal of the OMG for supporting the MDD approach using its own standards.



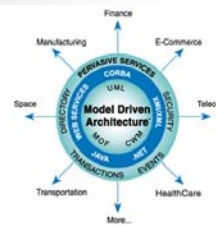
Model Driven Architecture

- MDA
- MDA: Beyond Technology
- (Expected) Advantages
- Basic Concepts. Examples
- Model Transformation
- Applying MDA to build a SW Application
- Advantages
- Some Questions
- Tool Support. Industrial CASE Tools



Model Driven Architecture

- MDA is an initiative of the OMG
 - Announced in 2000
 - 10 years of term to mature
 - It must persist during 20 years at least
- Middleware platforms move aside
- Models are the key
- MDA promotes the separation of the specification of the system functionality from its implementation in a specific technological platform
- <http://www.omg.org/mda>



Model Driven Architecture



- OMG Proposal (IBM, Borland, Hewlett-Packard, Boeing, etc.).
- *“MDA is an approach to use models in software development”.*
- Models = main resource for software development.



MDA: Beyond Technology

- **Currently, too many platforms and technologies exist**
 - Distributed objects, components, web services...
 - Actually, they are not interoperable
 - Which is the best technology that I should use (today)?
- **Technology evolution is too fast**
 - Technologies evolve...and are obsolete very soon.
 - Nowadays, I know very well the technologies but, which is the new hot technology that will go out tomorrow?
 - How long it will be in use?
 - How can I protect my investments against of those continuous changes?
- **I would like to separate my models from the concrete technology in which they are implemented (today)**
 - In a way that they can evolve in a independent way....
 - Without throwing away everything and starting from scratch every time one of them changes
 - and protecting my investment





(Expected) Advantages

- **Protect the investment against continuous technology changes**
 - Preserve the PIM of the **business model** when new middleware appears
- **Facilitates the development of more complex systems**
 - Separation of different concerns in different models
- **Allows the simulation and automatic implementation of the business models**
- **Allows the integration of existing systems (COTS, legacy systems)**
 - ADM: Architecture Driven Modernization
- **Allows the specification of the functional requirements in a way that is independent of the implementation platform**
 - MBA: Model-Based Adquisition



Basic Concepts

- **Computational Independent Model (CIM)**
 - “A model of the system and its environment, that describes the system requirements but hides the details of its structure and internals”
- **Platform Independent Model (PIM)**
 - “A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it.”
- **Platform Specific Model (PSM)**
 - “A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform.”
- **Platform**
 - “A set of subsystems/technologies that provide functionality through interfaces and specified usage patterns. It is specified so that any subsystem can use it without being aware of how the functionality provided by the platform is implemented.”



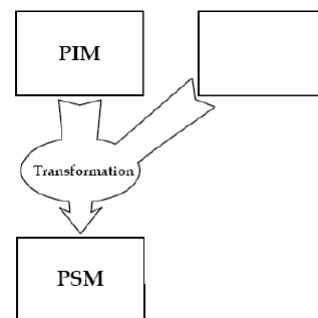
Examples of MDA Models

- **CIM**
 - Use Case Models capturing functional requirements
- **PIM**
 - Class Diagrams or Architecture Descriptions that include components and connectors
- **PSM**
 - A UML Profile for EJB
- **Code**
 - EJB components, configuration files and every file or information to completely deploy the software system



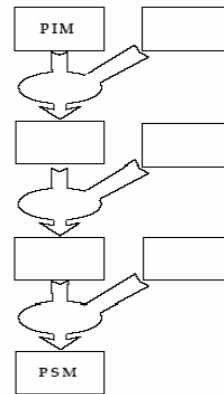
Model Transformation

- **A model transformation specifies the process of converting one model into another**
- **MDA pattern includes (at least):**
 - a PIM,
 - a Platform Model (PSM), and
 - a Transformation

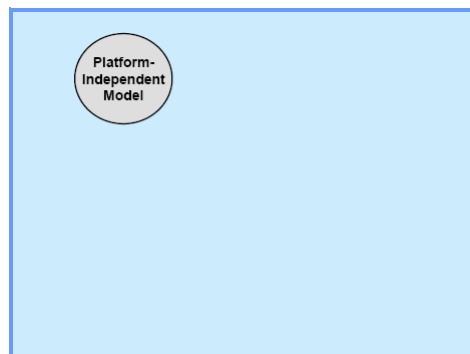


Model Transformation

- The MDA pattern can be applied in a consecutive way (several times) to produce successive changes:
- In a transformation the resulting “PSM” is the PIM of the following transformation
- In this way, each “platform” focus on a different aspect of the system



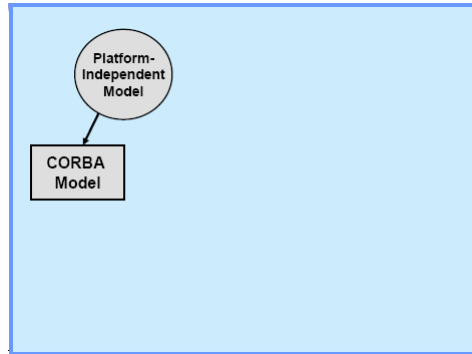
Applying MDA to build a SW Application



- We start with a Platform-Independent Model (PIM) that represents the business logic (functionality) in a way that is independent of implementation details



Applying MDA to build a SW Application

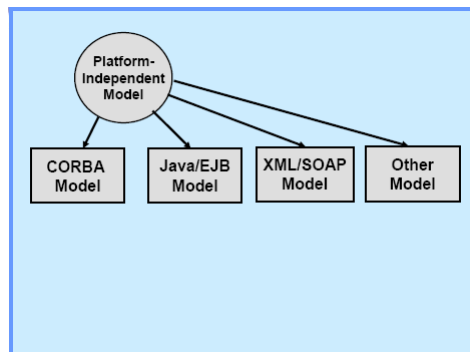


Generating the PSM

- Transformations (between source and target metamodels) can be defined with QVT
- Transformations can be partially or completely automated



Applying MDA to build a SW Application



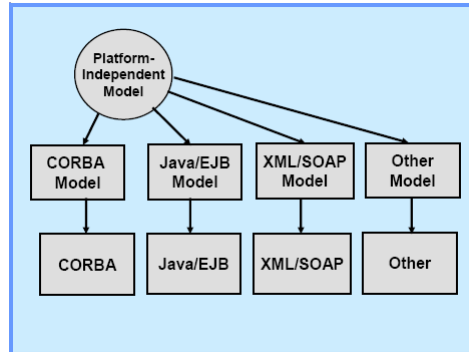
Generating to Multiple Tecnologies

- Transformations (between source and target metamodels) can be defined with QVT
- Transformations can be partially or completely automated



Applying MDA to build a SW Application

~~Write Once, Run Everywhere~~
Model Once, Generate Everywhere!

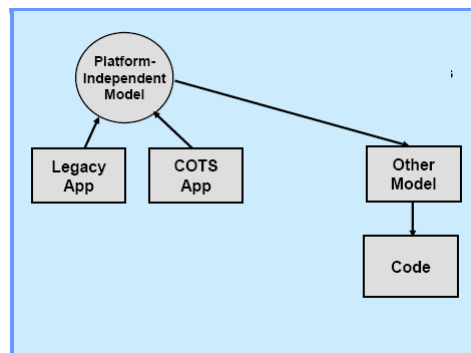


- It is easy to build code generators from PSMs because they are specified in a low level of abstraction

Generating Implementations



Applying MDA to build a SW Application



- Useful to:
 - (1) **Integrate** COTS, third party and legacy systems into our application
 - (2) **Support Architecture Driven Modernization**

NASA, DoD, Banks

Systems Integration





Advantages

- **Each model (CIM, PIM, PSM) is independent of the rest**
 - They are defined in a separate way
 - Each model defines its own “entities” at an adequate abstraction level, and it is expressed in a language that is appropriate for the kind of *stakeholders* that are going to interact with it
- **The development process turns into a model transformation process**
 - Each step selects a “platform” and transform one or more PIMs into one (or more) PSMs
 - ...until reaching the final implementation
 - Transformations can be automated
- **Modularity, Flexibility and Easy Evolution**
- **The models capture the business logic and they are valuable for the company**



Some Questions Arise

- **Can we adapt us to the new way of development?**
 - What can we do with the development team?
 - Do we want to change the processes and development tools?
 - New skills, knowledge and tools are needed!
- **What can we do with programmers?**
 - Should they be fired?
 - Should they be re-educated?
- **MDD needs to put money (investment)**
 - How much it costs? Can we justify the expenses?
- **Are we changing of platform and technology continuously?**
- **Is the generated code efficient?**
- **Is the generated code readable and easy to maintain?**
- **Can be applied to every application domain?**
- **Do we have CASE Tools supporting MDA processes?**



Tool Support

- OptimalJ
- ArcStyler
- AndroMDA
- Poseidon
- Together
- ONME
- ...much more (OMONDO)

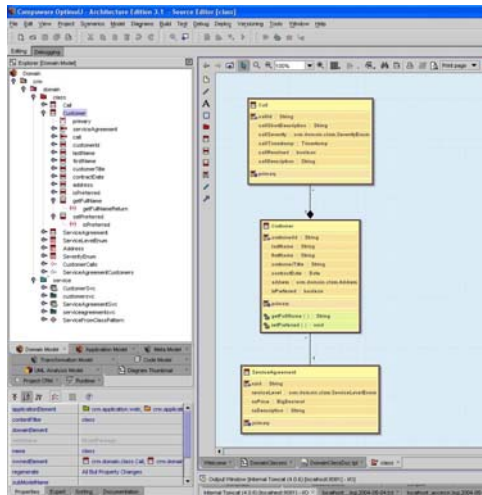


Tool Support

OptimalJ

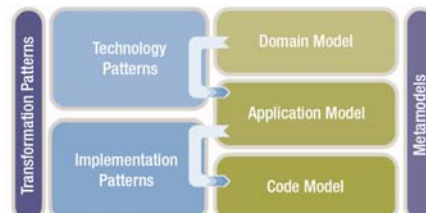


OptimalJ

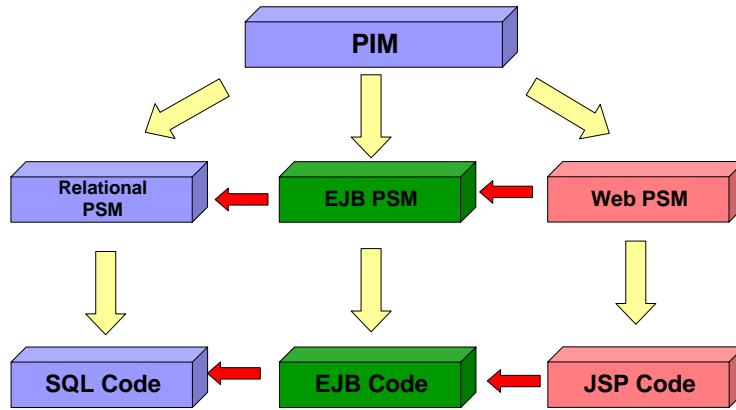


OptimalJ

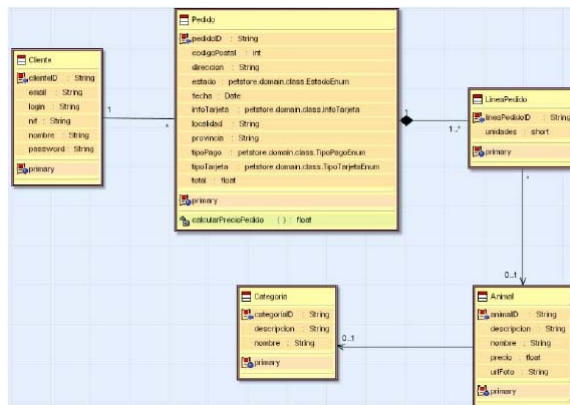
- Generation of distributed applications implemented in a J2EE architecture and the Struts *framework*
- Domain Model (PIM)
 - Class Model
- Application Model (PSM)
 - Presentation Model (Web)
 - Business Model (EJB)
 - Database Model
- Code Model



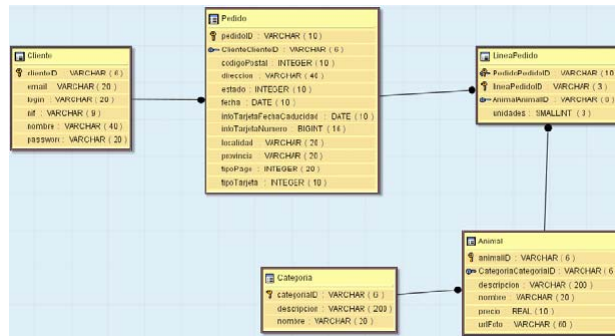
OptimalJ



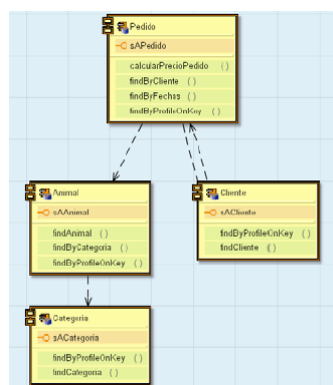
OptimalJ (PIM)



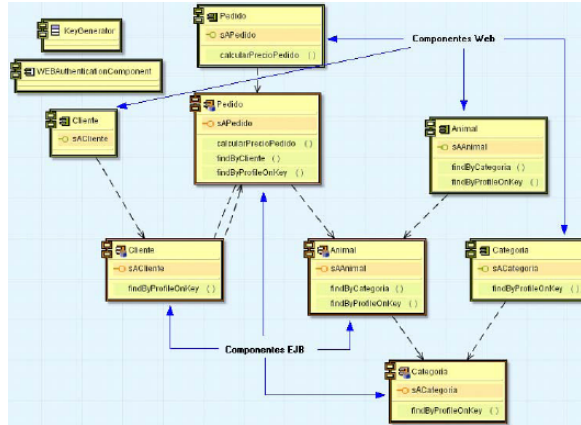
OptimalJ (Relational PSM)



OptimalJ (EJB PSM)



OptimalJ (Web PSM)



OptimalJ (Code)

```

public float calcularPrecioPedido() throws AlturaPreConditionException, AlturaPostConditionException {
    float returnValue = 0;

    // your code here

    // Changes in bean means you persisted data (e.g. appropriated data) need to be signaled to the application server.
    // In order to do so, call the following method (it is commented out by default for performance purposes).

    Iterator it = this.getLineaPedidoColl.iterator();
    while (it.hasNext()) {
        LineaPedidoData l = (LineaPedidoData)it.next();
        AnimalKey ak = l.animalKey;

        //Obtenemos la referencia a la instancia home de Animal
        AnimalLocalHome animalLocalHome = this.getAnimalLocalHome();
        AnimalLocal animalLocal = animalLocalHome.findByPrimaryKey(l.animalKey);
        AnimalLocalHome animalIDHome = animalLocal.getHome();
        // Incrementamos el precio total
        returnValue += animalIDHome.getPrecio() * l.getCantidad();
    }

    return returnValue;
}
    
```



Tool Support

ArcStyler

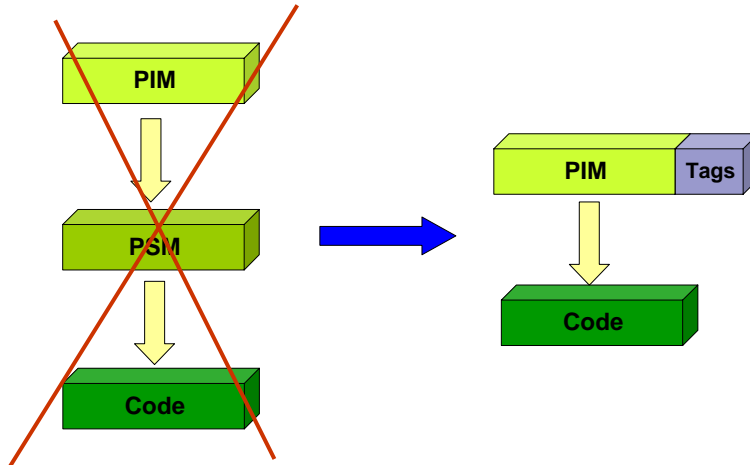


ArcStyler

- Model Transformations are expressed through MDA-Cartridges
- CARAT Architecture
- Available cartridges for EJB, .NET, web services, etc.
- Support to the creation of new cartridges
- Annotated PIMs, no PSMs



ArcStyler



Tool Support

AndroMDA





AndroMDA

- AndroMDA is an extensible framework for code generation that follows the principles of MDA.
- Models of several UML CASE Tools can be transformed into components in the selected platform (J2EE, Spring, .NET).
- AndroMDA is an *open source* tool.
- Transformations are based on *cartridges* that generate code from models with stereotyped elements.



AndroMDA

- AndroMDA provides a set of implemented cartridges that allow code generation in Axis, jBPM, Struts, JSF, Spring and Hibernate.
- AndroMDA also provides a toolkit for building cartridges or customizing existing ones.
- Support to MagicDraw, Poseidon, Enterprise Architect and others.
- Templates are based on template engines like Velocity and FreeMarker.



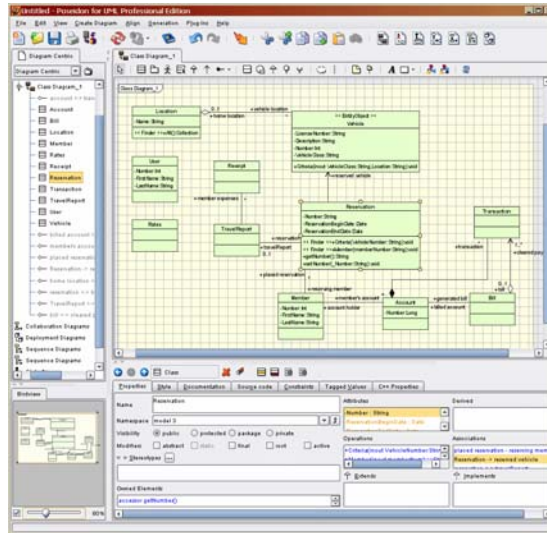
MagicDraw



Poseidon



Poseidon

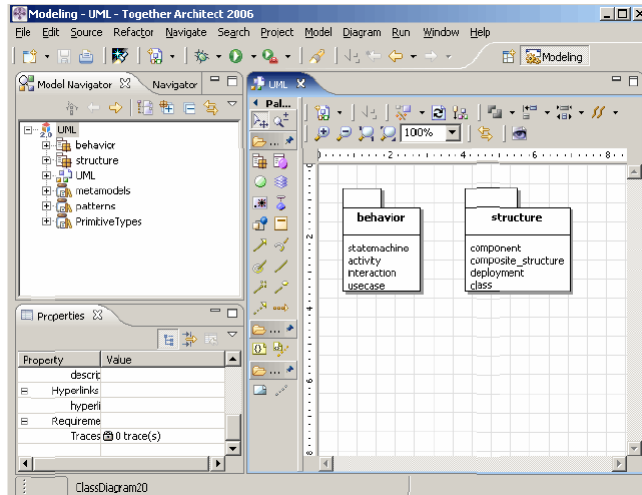


Tool Support

Together 2006



Together 2006



Tool Support OlivaNova Model Execution



CARE
TECHNOLOGIES

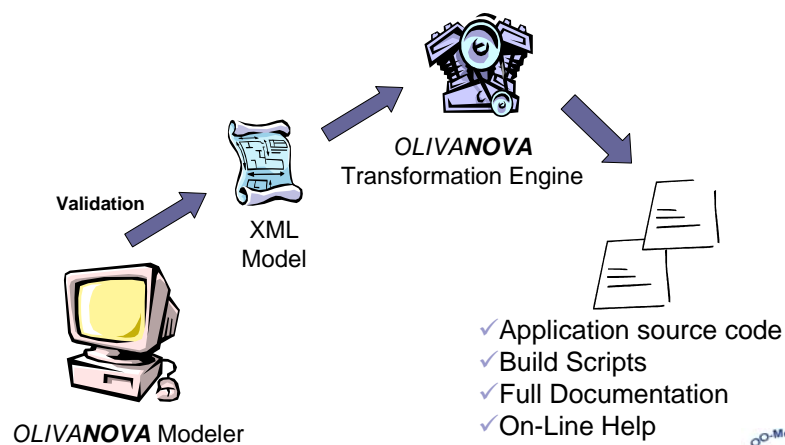


ONME

- CASE Tool providing 100% code generation.
- Implements MDA philosophy.
- PIM to Text Code Generation (transformation engines and repositories use proprietary technology).
- Based on the OO-Method. The Method was born and developed at the UPV.
- CARE technologies company is a spin-off whose headquarter is located at Denia (Valencia).

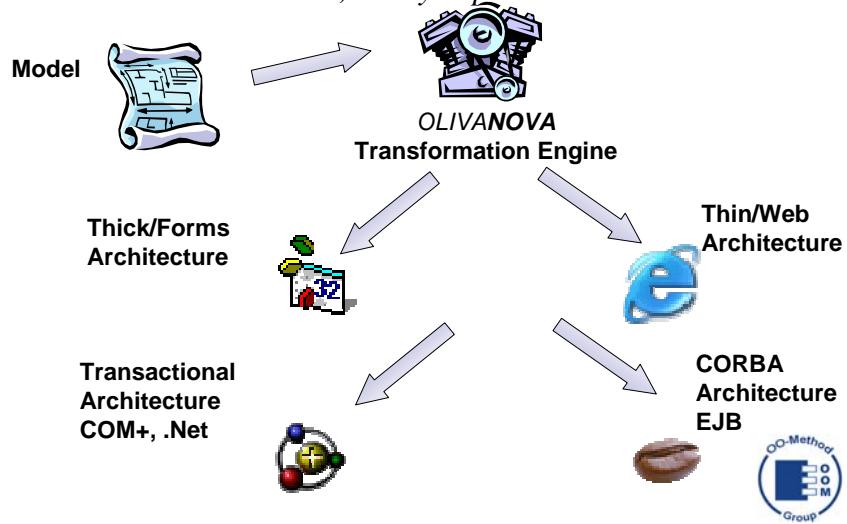


ONME (Transformation Engine)

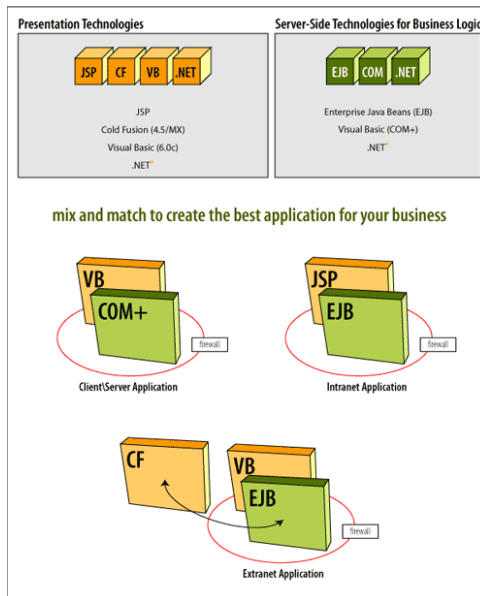


ONME (Several Technologies)

One model, many implementations



ONME





MDA vs MDD

- Differences between MDD and MDA:
 - **Model Driven Development (MDD):** Software Development Approach that is based on modeling the sw system and generating the code from models.
 - **Model Driven Architecture (MDA):** MDD + OMG languages (UML, CWM, MOF, QVT).
- MDA = Infrastructure for MDD



Software Factories

- Software Factories
 - Systematic Reuse and Product Lines
 - Model Driven Development and Domain Specific Languages
 - Incremental Code Generation and Frameworks Development
 - Component Assembly
 - Process Frameworks
- Microsoft DSL Tools (VS 2005) and MetaEdit +



Software Factories

"It is a software product line that configures extensible tools, processes and content [...] to automate the development and maintenance of variations of a archetypical product, through the adaptation, assembling and configuration of components that are based on frameworks."



"a Software Factory is a development environment configured to support the rapid development of a specific type of application." Jack Greenfield



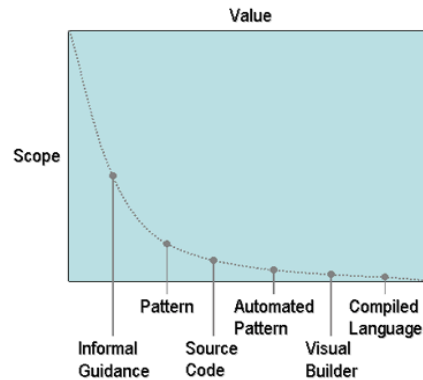
Software Factories

- **Objective:** Increase the Level of Automation in the SW Production Process
- Are we really doing it?
- Why does it fail? Two reasons:
 - The Current Economic Model
 - Chronic Software Development Problems



Software Factories

The Current Economic Model



"the value of an abstraction increases with its level of specificity to the problem domain." M.Jackson The more knowledge an abstraction provides, the narrower the domain (scope) to which it applies, but the more value it provides in solving problems in that domain.



Software Factories

Chronic Software Development Problems

- Monolithic Construction
- Gratuitous Generality
- One Off Development (without Reuse)
- Process Inmaturity

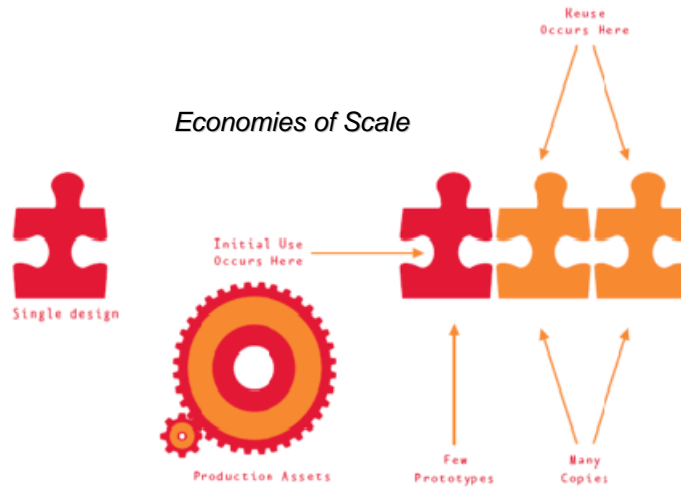
How do we move forward?

- Systematic Reuse,
- Model Driven Development,
- Development by Assembling Components, and
- Process Frameworks.



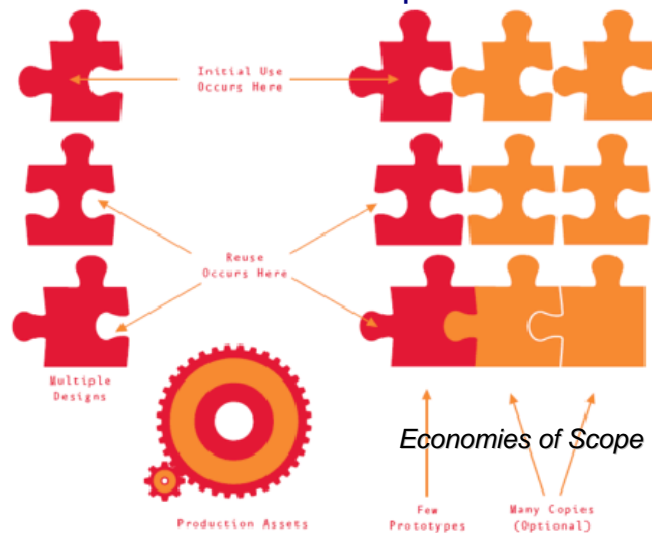
Systematic Reuse

Economies of Scale vs. Scope



Systematic Reuse

Economies of Scale vs. Scope



SW Product Lines

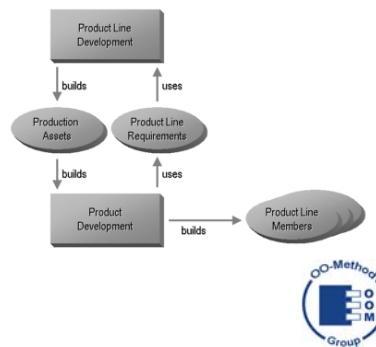


What is a Software Product Line?

A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.



A **SPL** produces a **Family of Software Products** in a specific domain



Model Driven Development

- The aim of the MDD is to rise the level of abstraction (*with tools*) to provide higher levels of automation (*for narrower problem domains*).
- MDD **use models** to capture information and to automate their implementation by compiling models to produce executables or using them to guide the development.
- MDD can make easier the automation of activities like debugging and software configuration.

© Jack Greenfield





Model Driven Development

Domain Specific Languages (DSL/DSM)

- The Software Factories are interested in precise models that can be processed by tools and can be used in the same way as the source code.
- To rise the level of abstraction, a modeling language should be oriented to narrower (specific) domains than those that a general purpose language can model.



Model Driven Development

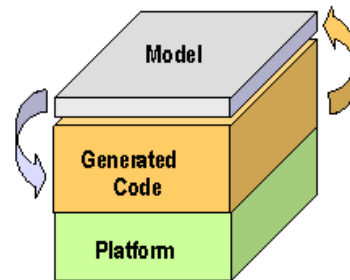
Domain Specific Languages (DSL/DSM)

- The **purpose** for which the language is designed must be explicitly stated, so that an observer familiar with the domain can **evaluate the language** and determine whether or not it **fulfills its purpose**.
- The language must capture **concepts** used by **people** who work with the **domain**.
- The language must use **names** for those concepts that are **familiar to the people** who use them.
- The **notation** for the language, whether graphical or textual, must **be easy to use**.
- The language must have a **well defined set of rules**, called a grammar, governing the way the concepts can be combined to form expressions.
- The **meaning** of every well formed expression must be **well defined**, so that users can build models that other **users understand**, so that tools can **generate valid implementations** from models.



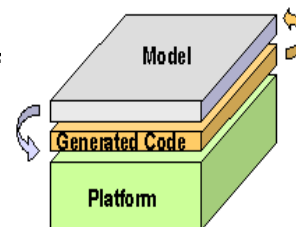
Incremental Code Generation

- The key to effective code generation is to generate **small amounts of code** that span only **small abstraction gaps**.
- This allows the tool to take advantage of platform features, and therefore to produce focused, efficient, platform specific implementations.



Incremental Code Generation

- One way to make code generation more incremental is to **move the models closer to the platform**.
- The model can now become a **view of the code**.
- The tool exposes relationships and dependencies that are hard to see in the code, and saves time and effort by generating the code for program structures.



BUT...



Incremental Code Generation

- This **reduces the power of the model**, by limiting it to abstractions already available on the platform, or only slightly more powerful, such as programming idioms.
- How then, *do we work at higher levels of abstraction?* We use more abstract models, and move the platform closer to the models with either **frameworks** or **transformations**.



Incremental Code Generation

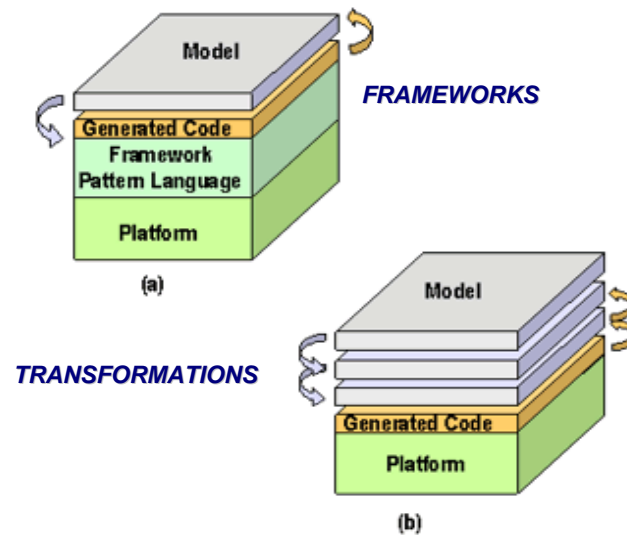
Frameworks + Transformations

- We can use a framework to implement higher level abstractions that appear in the models, and use the models to generate snippets of code at framework extension points.
- A pattern language can be used instead of a framework.
- Instead of a framework or pattern language, we can generate to a lower level DSL. We can also use more than two DSLs to span a wide gap, leading to progressive transformation, where models written using the highest level DSLs are transformed into executables through a series of refinements.
- This is how compilers work, transforming expressions written in a relatively high level language like C# first into an intermediate representation like byte code or IL, and then into a binary format for a target platform using just-in-time (JIT) compilation.



Incremental Code Generation

Frameworks + Transformations



Component Assembling

- Platform Independent Protocols
- Self Description
- Assembly by Orchestration
- Architecture Driven Development





Process Frameworks

- **Scalable** and **Agile Processes**.
- **Small infrastructure**.
- Processes must be adapted/specialized to build **suitable processes for a product family**.
- This kind of personalization only makes sense if it can be reused more than once (*Not reinvent the wheel*).



Tool Support

- DSL TOOLS
- METAEDIT +
- And also ECLIPSE Modeling Project...



Microsoft DSL Tools

Suite for defining DSLs, building a graphical designer and defining code generators in Visual Studio 2005.

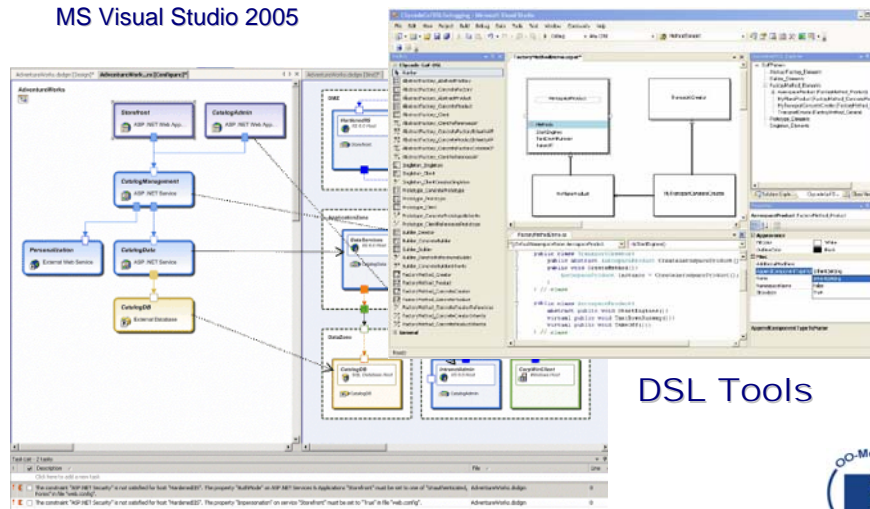
DSL Tools provide a *project wizard* to create a DSL:

- Allows defining and editing a **domain specific language** through a graphical designer (serialized in a proprietary XML format).
- Allows defining **designer definitions** using a proprietary XML format which is the source to generate the code (without any manual programming) that implements graphical modelers of the DSL.
- Includes **Code Generators** that take a DSL definition and a designer producing the code that implements the graphical editors.
- Includes a framework to **define code generators based on template languages** that take an instance of a domain model and generates code based on the template.



Microsoft DSL Tools

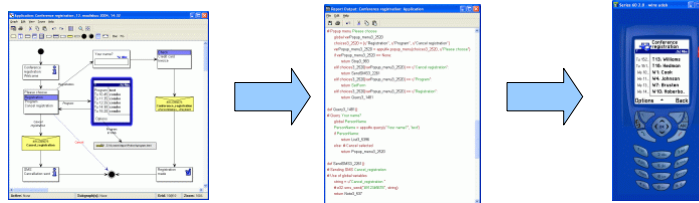
MS Visual Studio 2005



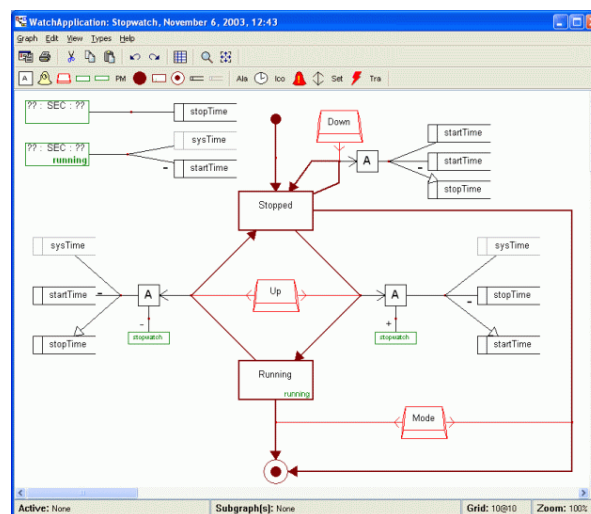
MetaEdit+ (<http://www.metacase.com>)



- MetaEditor (Allows building Graphical Editors for DSLs)
- Proprietary Metamodeling Notation and Repository
 - Provides an API for model management
- Code Generation using Templates (proprietary language)
- Supporting DSM (in multiple domains)
 - http://www.metacase.com/cases/dsm_examples.html



MetaEdit+ (<http://www.metacase.com>)



MDA vs. Software Factories

- Technical Journals and personal blogs
- Commercial Interest vs. Scientific arguments
- Advantages and Weak Points
- Which approach should I choose?
- Are these approaches compatible? Can they be combined?



MDA vs. Software Factories

| <i>MDA</i> | <i>Software Factories</i> |
|--|---|
| (Ok) Provides Techniques | (X) Not concrete |
| (X) Only (PIM->PSM) | (Ok) More Guides and Methodological Support (DSL, Frameworks, etc.) |
| (Ok) Older | (X) New (2004) |
| (Ok) More Tools (Arcstyler, OptimalJ, EMF, etc.) | (X) Inmature Tools and not well known (DSL, MetaEdit+) |
| (X) QVT not widely supported | (Ok) Integrates several mature techniques |
| (_) Promoted by OMG | (_) Promoted by Microsoft |





MDA vs. Software Factories

- Integration. **Why not?**

- Development of a product line
- Implementation Framework
- Building a DSL
 - Conceptual Primitives that are Platform Independent (PIM)
 - MOF or UML profile
- Code Generation through Transformations “**DSL to Framework**”



MDA vs. Software Factories


- They are compatible approaches
 - But not recommended by
 - OMG: Say No to the “Babel” of Languages
 - Microsoft: OMG Standards are not precise enough
- The selection depends on circumstances
- The final success can depend on marketing/commercial reasons.





UML vs. DSL

- Domain Specific Languages (DSL/DSM)
- Language Oriented Programming
- DSLs vs. UML/MDA



Domain Specific Languages (DSL/DSM)

- **Domain-Specific Modeling** raises the level of abstraction beyond programming by specifying the solution directly using domain concepts. The final products are generated from these high-level specifications.



- *Most new ideas in software developments are really new variations on old ideas.the growing idea of a class of tools that I call **Language Workbenches** - examples of which include **Intentional Software**, JetBrains's **Meta Programming System**, and Microsoft's **Software Factories**.*





Language Oriented Programming

■ Language Oriented Programming (Fowler, Dimitriev):

- Environments to build domain specific languages and to model (program) with them.
 - “*Language Oriented Programming: The next programming paradigm*” S. Dimitriev, JetBrains
 - <http://www.onboard.jetbrains.com/is1/articles/04/10/lop/>



Language Oriented Programming

- I use **Language Oriented Programming** to mean the general style of development which operates about the idea of building software around a set of **domain specific languages**. I use **Language Workbench** as a generic term for this new breed of tools. So a language workbench is one way to do language oriented programming. You may also be unfamiliar with the term **Domain Specific Language** (usually abbreviated to **DSL**). It is a limited form of computer language designed for a specific class of problems.

**martin
Fowler
.com**





DSLs vs. UML/MDA

Language Workbenches and Model Driven Architecture



- *“...In my view the MDA means different things to different people - and this effects how we view the **relationship between MDA and language workbenches**. Certainly there groups of MDA practitioners who are **using MDA ideas to build a language workbench**. However my feeling is that the help MDA provides is partial, at best. A broader school of **Model Driven Development (MDD)** echoes many of these ideas without the links to the MDA standards - this is something that is very much in line with the ideas of a language workbench. “*
 - **The UML PIM Camp:** “You could use the UML meta-model to define a DSL schema, but here the UML is both too much and too little”.
 - **The MOF Camp:** “MOF may be useful as an interchange mechanism between language workbenches for DSL schemas”.
 - **Closing Thoughts:** “I’ve become known for having a pretty skeptical view of the MDA. Most of this negativity is towards the UML PIM camp - I think the UML is too complex and is too semantically incoherent to act as a serious base for future work.”

