From Model Transformations to Model Driven Interoperability

Oslo | Apr, 21st 2008

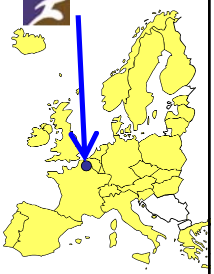
Jean-Pierre Bourey
Laboratoire de Génie Industriel de Lillel
Ecole Centrale de Lille (France)

Reyes Grangel
Research Group in Systems Integration and Re-Engineering (IRIS)
Universitat Jaume I (Spain)

"Please allow me to introduce myself, I'm…"*

- Jean-Pierre Bourey (jean-pierre.bourey@ec-lille.fr)
- Teaching domains
  - Software Engineering
  - Object-Oriented Modelling (UML)
  - Enterprise Modelling
  - Information Systems (Oracle)
  - Programming Language (Ada)
- Research domains
  - Petri Nets for Automated Systems
  - Modelling, Model Transformations,
  - Model Driven Interoperability

Ecole Centrale de Lille



*[Rolling Stones, Sympathy for the Devil, Beggars Banquet, 1968]

Contributor

- Reyes Grangel Seguer (grangel@uji.es)
  - Professora Col·laboradora
  - Research Group in Systems Integration and Re-Engineering (IRIS)
  - Department of Computer Languages and Systems at Universitat Jaume I (Castelló, Spain)
- Teaching domains
  - Software Engineering
  - Object-Oriented Modelling (UML)
  - Enterprise Modelling
  - Information Systems
  - ERP
- Research domains
  - Model Driven Interoperability
  - Knowledge Management

Objectives and Prerequisites

- Objectives
  - To give an overview on Models, Modelling and Model Transformations
  - To present an example of the use of a transformation language
  - To explain the OMG's Model Driven Architecture®
  - To present the current works on Model Driven Interoperability
  - To illustrate pragmatically all these points on concrete tested examples
- Prerequisites (but not strict)
  - To be able to read UML class diagrams
- OMG, Object Management Group, MDA, Model Driven Architecture, Model Driven Development, XMI, UML, Unified Modeling Language are either registered trademarks or trademarks of Object Management Group,Inc. in the United States and/or other countries.

Outline

- Part I: Models and Model Transformations
  - 1.1) Definitions: Model, Metamodel
  - 1.2) Transformations and Mappings
  - 1.3) Basic Examples
- Part II: Transformation Techniques
  - 2.1) From Programming Languages to Model Transformation Languages
  - 2.2) Applications
- Part III: Model Driven Architecture
- Part IV: Model Driven Interoperability

Part 1: Models and Model Transformations

1.1) Definitions: Model, Metamodel
1.2) Transformations and Mappings
1.3) Basic Examples

Oslo | Apr, 21st 2008

First Word about Models

"Essentially, all models are wrong, …

but some are useful "
Box, George E. P.; Norman R. Draper (1987)
Empirical Model-Building and Response Surfaces

Basic Concepts: formalism

- Language or formalism
- Set of constructs
  - Providing a **syntax** and a **semantics**
  - Put together according **precise grammatical rules**
  - **To represent an artefact** (object, message, knowledge, …)
- Syntax and Semantics
  - A language is needed to express/communicate the syntax and the semantics

Semantics

- Types of semantics
  - **Real World Semantics**: meaning refers to the real world which is modelled
  - **Formal Semantics**: based on mathematics, to validate, to transform, … …models
- Semantics required
  - To compare languages (used to represent the same thing)
  - To compare models (representing the same thing)
  - To validate models
- But problem of Semantics!
- "Semantics: this does not mean anything!" [Favre 2004]

Formalisms

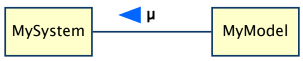
- Examples
  - Petri Nets (places, transitions,…)
  - Grafcet (steps, transitions,…)
  - UML Class Diagram (classes, associations, attributes, …)
  - Differential equations
  - etc

Model

- A **model** is a **consensus** about an **abstraction** of phenomena of the real world: a **representation** of an **aspect** of the world for a specific **purpose**
- Model = Representation
- Types:
  - informal (texts in natural language)
  - semi-formal (descriptive models, graphical notations)
  - formal (mathematical notations, First Order Logic)

Model

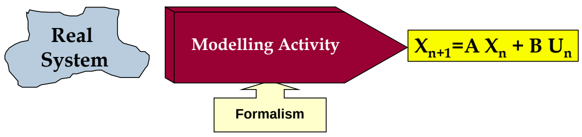
μ = represents



[Favre 2004]

Modelling

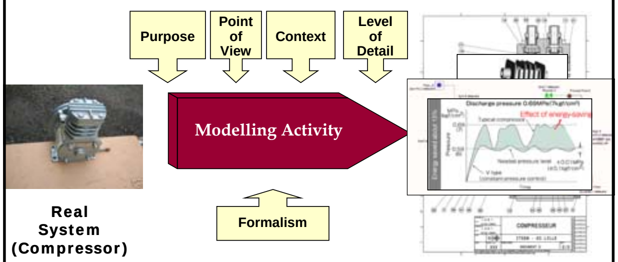
- Modelling
  - Activity of creating models
  - …using a formalism



Fundamental questions

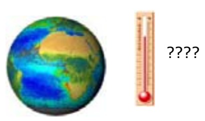
- Fundamental questions about a model :
  - Purpose : to do what ?
  - Scope of the model : context?
  - Point of View : aspects taken into account?
  - Level of detail : precision level?
- Always start with the purpose!

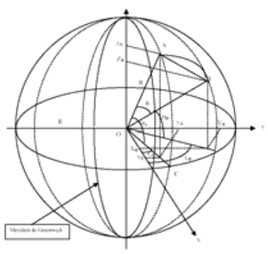
Model, Modelling



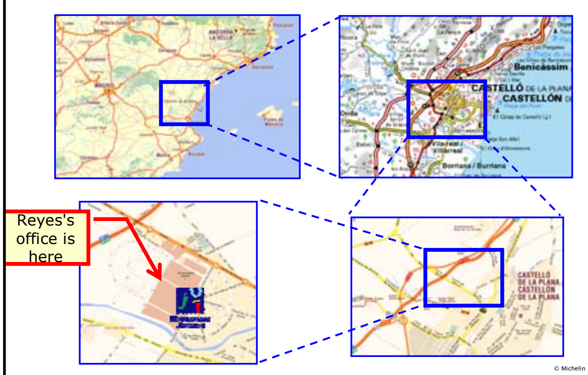
What is a good model ?

- "M is a good model of S if M makes it possible to answer predefined questions about S in a satisfactory way" (Ross)
  - Right level of detail
  - Right point of view

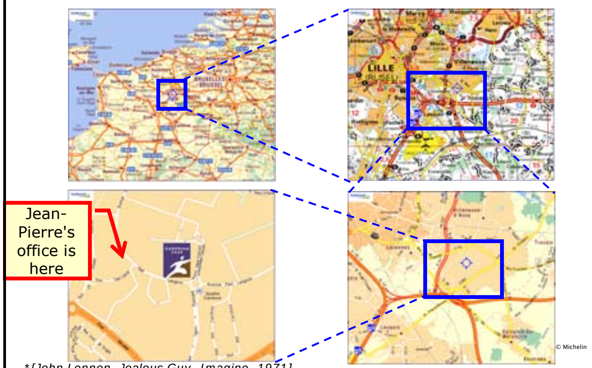




Model and Level of details

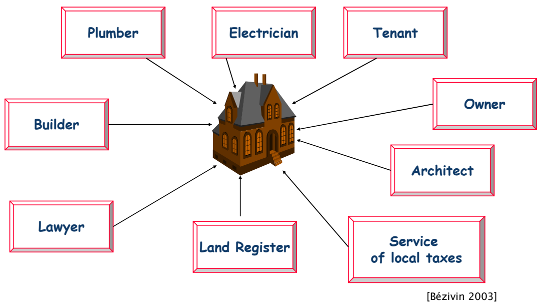


Model and Level of details (since "I'm just a jealous guy"*)

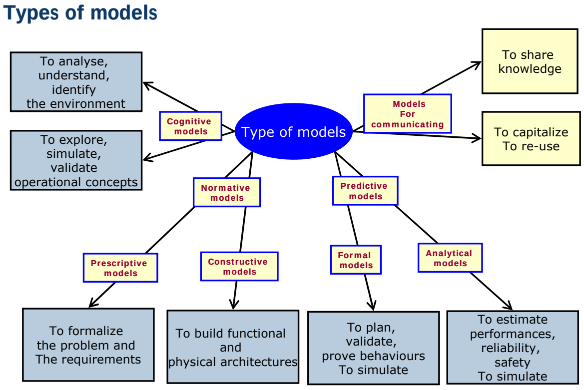


*[John Lennon, Jealous Guy, Imagine, 1971]

Points of view



Types of models



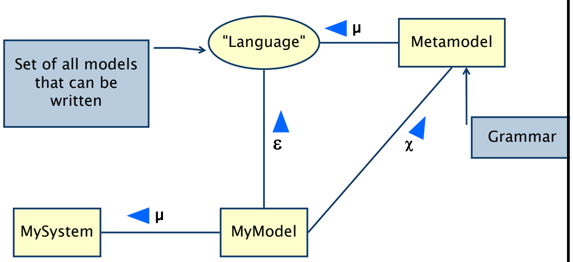
From [http://www.afis.fr]

From Model to Metamodel

- A **metamodel** is yet another abstraction, **highlighting properties of the model itself**.
- This model is said to **be conformed to** its metamodel
  - like a program corresponds to the grammar of the programming language used to write it
- A metamodel can be considered as an **explicit description** (constructs and rules) of **the way to build a domain-specific model**
- What is the exact nature of the relations between system, model, metamodel,... ?

From Model to Metamodel

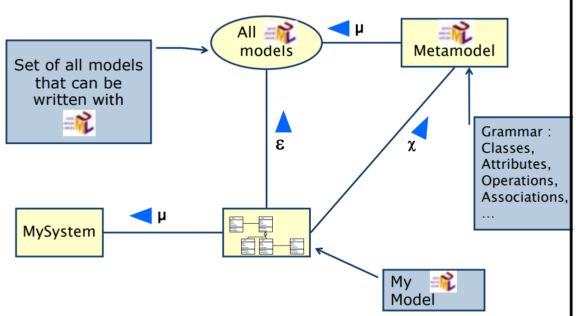
μ = represents
χ = conforms to
ε = belongs to

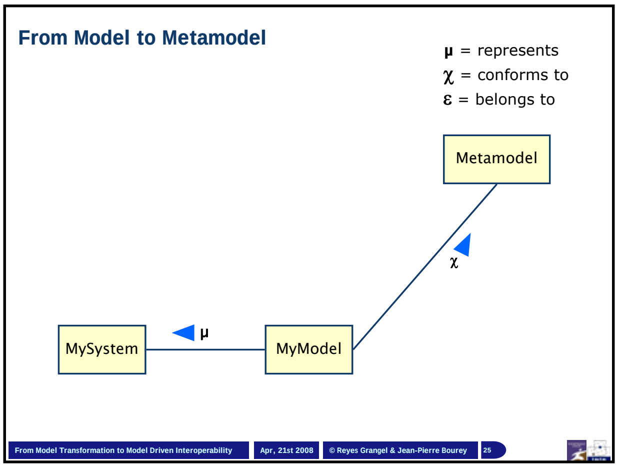


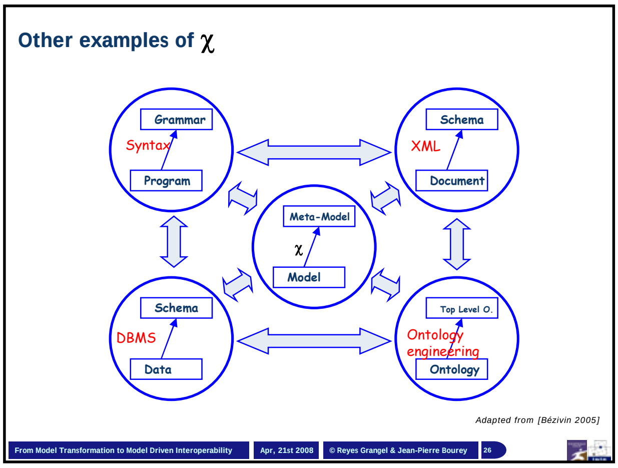
[Favre 2004]

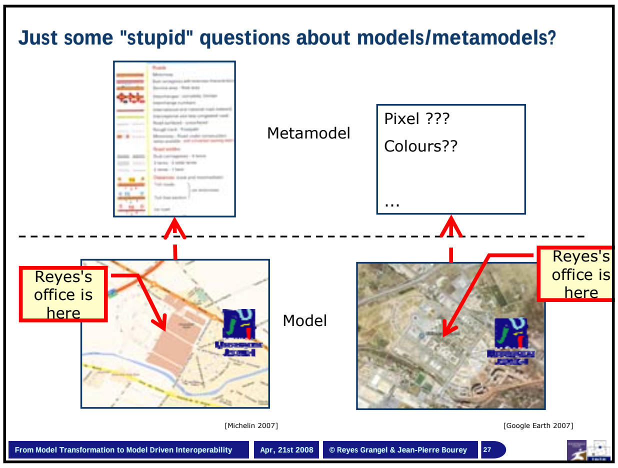
From Model to Metamodel: Example

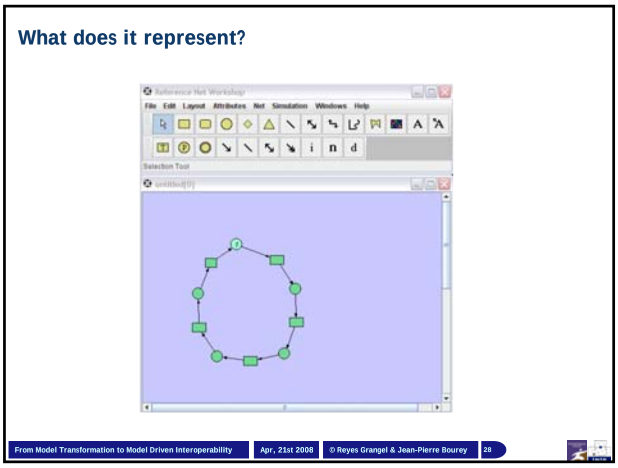
μ = represents
χ = conforms to
ε = belongs to

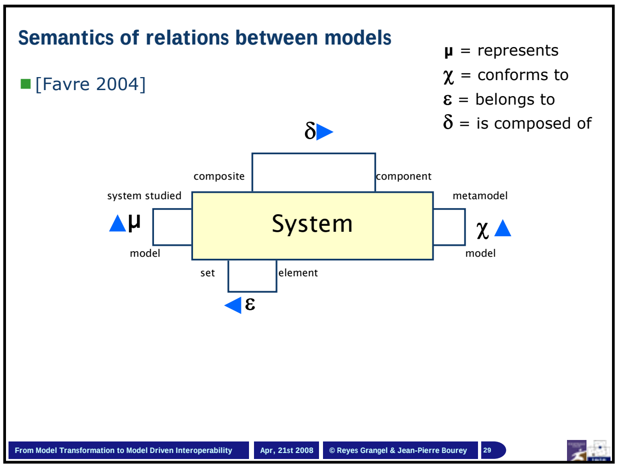


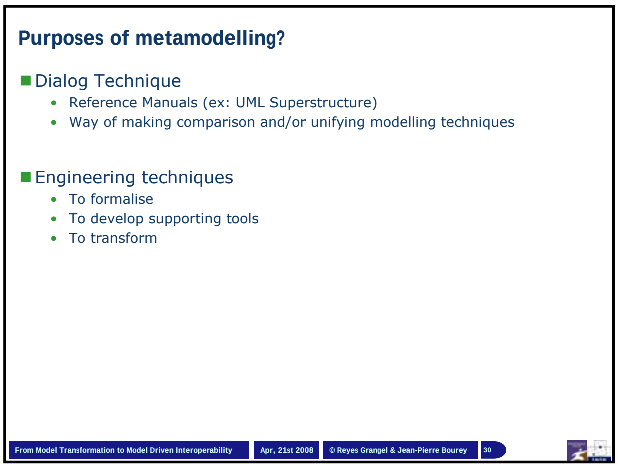

From Model to Metamodel
μ = represents
χ = conforms to
ε = belongs to
Metamodel
χ
μ
MySystem
MyModel
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
25

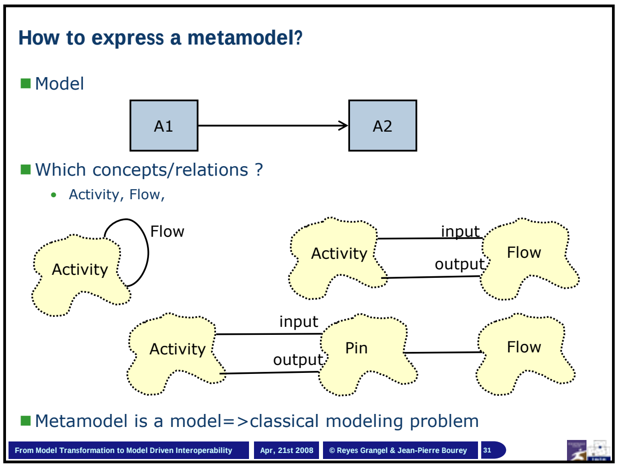

Other examples of χ
Grammar
Syntax
Program
Schema
XML
Document
Meta-Model
χ
Model
Schema
DBMS
Data
Top Level O.
Ontology engineering
Ontology
Adapted from [Bézivin 2005]
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
26

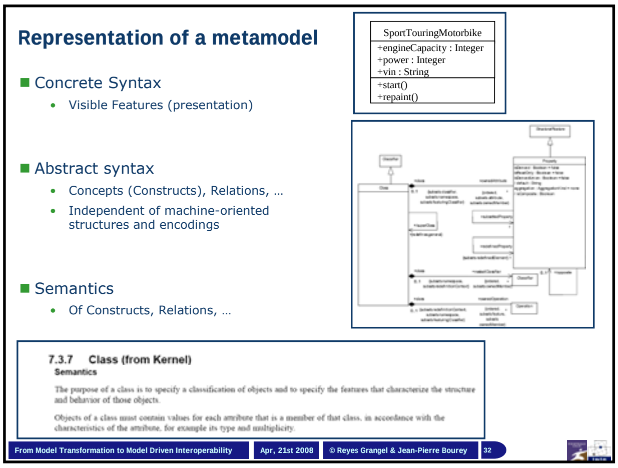

Just some "stupid" questions about models/metamodels?
Metamodel
Pixel ???
Colours??
...
Reyes's office is here
Model
Reyes's office is here
[Michelin 2007]
[Google Earth 2007]
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
27

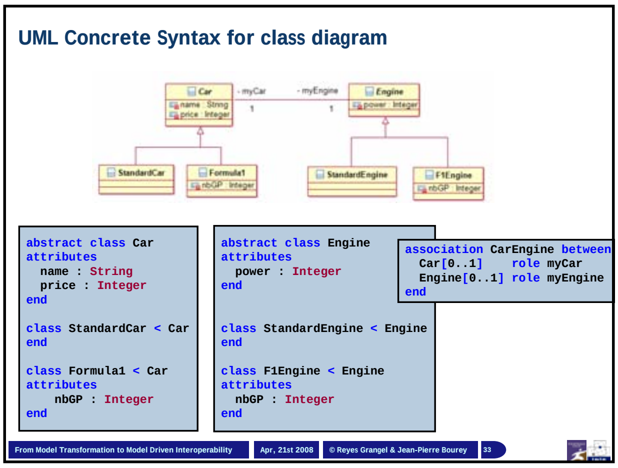

What does it represent?
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
28

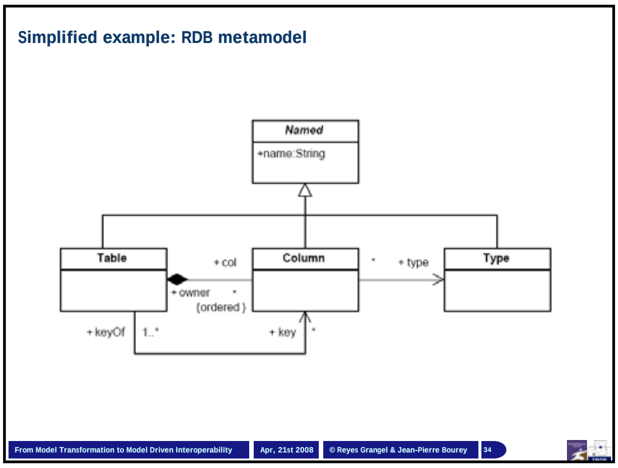

Semantics of relations between models
[Favre 2004]
μ = represents
χ = conforms to
ε = belongs to
δ = is composed of
δ
composite
component
system studied
metamodel
μ
System
χ
model
model
set
element
ε
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
29

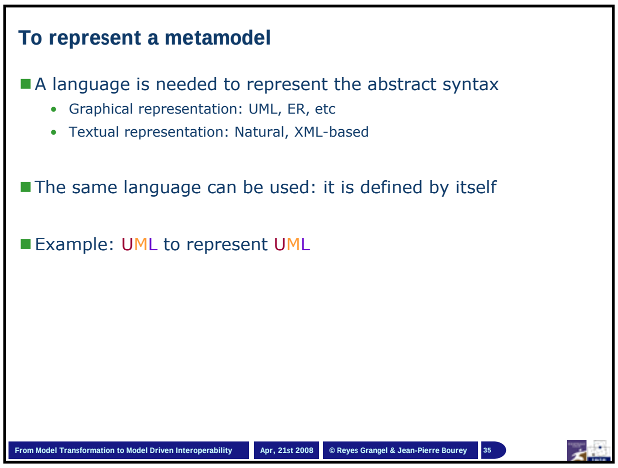

Purposes of metamodelling?
Dialog Technique
Reference Manuals (ex: UML Superstructure)
Way of making comparison and/or unifying modelling techniques
Engineering techniques
To formalise
To develop supporting tools
To transform
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
30

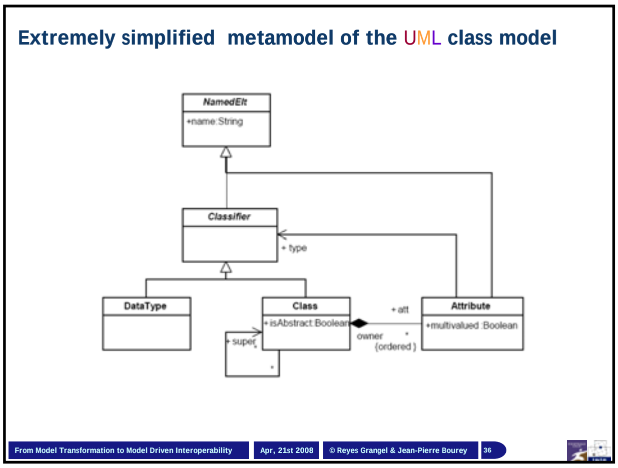

How to express a metamodel?
Model
A1
A2
Which concepts/relations ?
Activity, Flow,
Flow
Activity
Activity
input
output
Flow
Activity
input
output
Pin
Flow
Metamodel is a model=>classical modeling problem
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
31

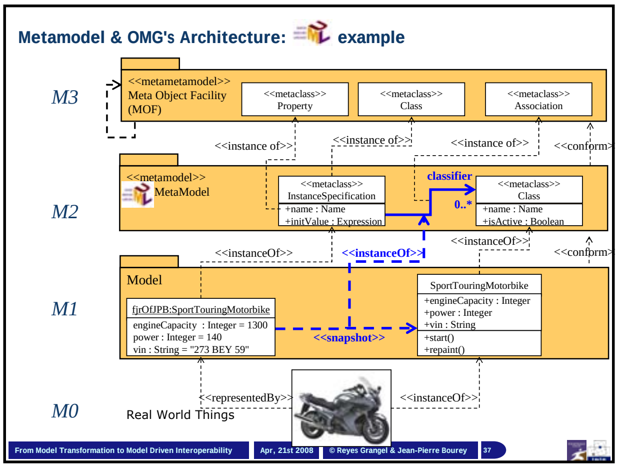

Representation of a metamodel
Concrete Syntax
Visible Features (presentation)
Abstract syntax
Concepts (Constructs), Relations, ...
Independent of machine-oriented structures and encodings
Semantics
Of Constructs, Relations, ...
SportTouringMotorbike
+engineCapacity : Integer
+power : Integer
+vin : String
+start()
+repaint()
7.3.7 Class (from Kernel)
Semantics
The purpose of a class is to specify a classification of objects and to specify the features that characterize the structure and behavior of those objects.
Objects of a class must contain values for each attribute that is a member of that class, in accordance with the characteristics of the attribute, for example its type and multiplicity.
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
32

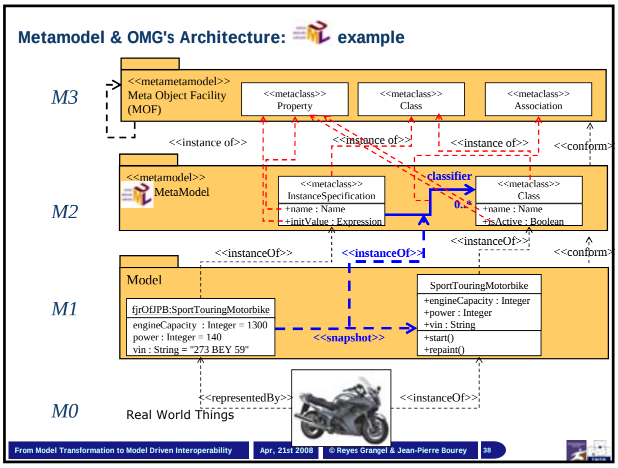

UML Concrete Syntax for class diagram
Car
name : String
price : Integer
- myCar
- myEngine
1
1
Engine
power : Integer
StandardCar
Formula1
nbGP : Integer
StandardEngine
F1Engine
nbGP : Integer
abstract class Car
attributes
name : String
price : Integer
end
class StandardCar < Car
end
class Formula1 < Car
attributes
nbGP : Integer
end
abstract class Engine
attributes
power : Integer
end
class StandardEngine < Engine
end
class F1Engine < Engine
attributes
nbGP : Integer
end
association CarEngine between
Car[0..1] role myCar
Engine[0..1] role myEngine
end
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
33

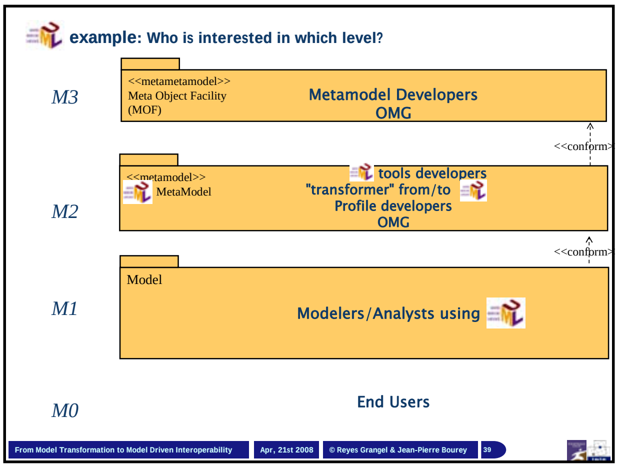

Simplified example: RDB metamodel
Named
+name:String
Table
+ col
Column
*
+ type
Type
+ owner
{ordered }
+ keyOf
1..*
+ key
*
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
34

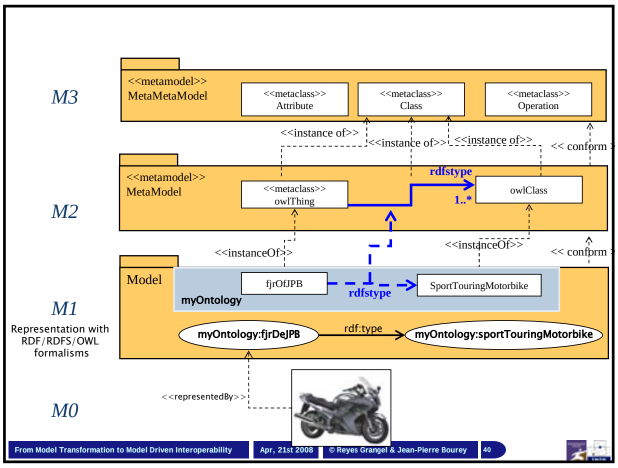

To represent a metamodel
A language is needed to represent the abstract syntax
Graphical representation: UML, ER, etc
Textual representation: Natural, XML-based
The same language can be used: it is defined by itself
Example: UML to represent UML
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
35

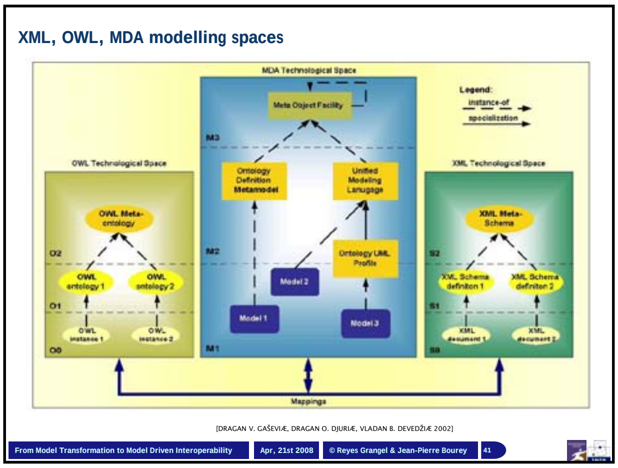

Extremely simplified metamodel of the UML class model
NamedElt
+name:String
Classifier
+ type
DataType
Class
+isAbstract:Boolean
+ super
+ att
Attribute
+multivalued:Boolean
owner
{ordered }
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
36











Model, MetaModels & Model Driven Engineering

- **Model-Driven Engineering** (or **MDE**)
  - systematic use of **models** as **primary engineering artefacts** throughout the engineering lifecycle
- In MDE
  - Models are considered as first class citizen
  - Everything is model
- MDE can be applied to
  - software,
  - system,
  - data engineering,
  - …
- OMG's **Model Driven Architecture (MDA)** is one MDE initiative
- MDE is based on **Model** and **Model Transformations**

From Model Transformation to Model Driven Interoperability | Apr, 21st 2008 | © Reyes Grangel & Jean-Pierre Bourey | 42

Last words about Models

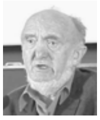
"We don't see the world with our eyes,...

We see it with our concepts"

From "Petite philosophie à l'usage des non philosophes" 1997

Albert Jacquart (1925-)

Genetician, Writer



Part 1: Model Transformations

1.1) Definitions: Model, Metamodel

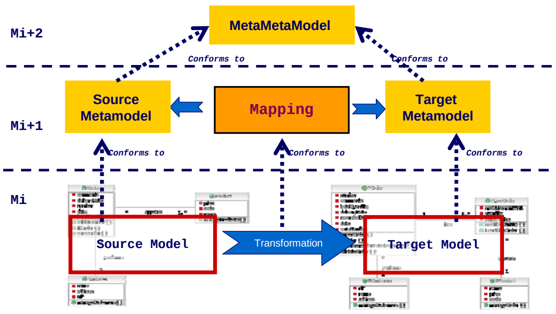
1.2) Transformation and Mappings

1.3) Basic Examples

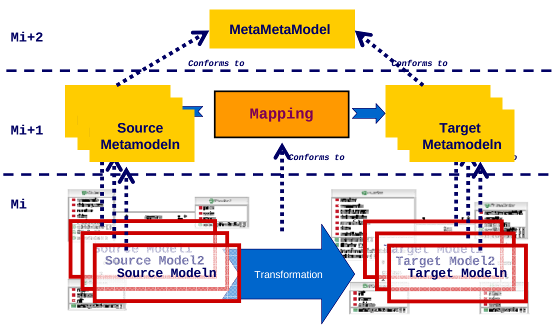
Model Transformation

- A Model Transformation takes a model and produce another model
- Model Transformations are …
- …..Models

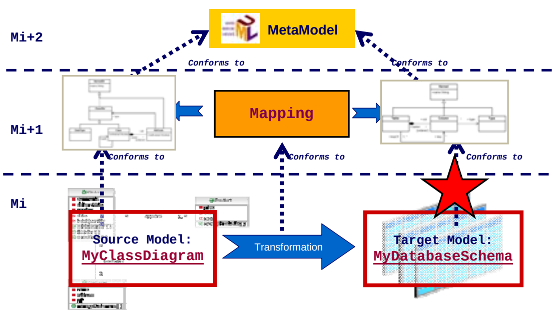
Transformation Architecture



Model Weaving



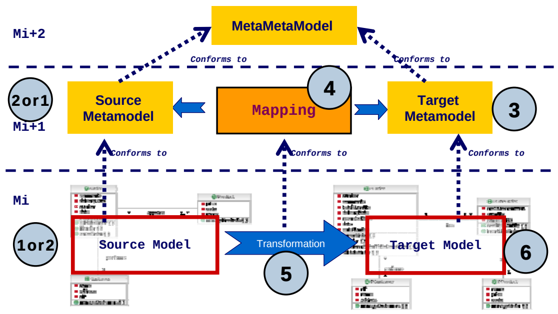
Transformation: UML2RDBMS



Mapping

- A specification for a Transformation [MDA 2003]
- Synonym: Model Morphism [MoMo 2006]
  - a **morphism** is an abstraction of a structure-preserving process between two mathematical structures
- To establish correspondences between
  - one or more constructs of the source language
  - and 0 or more constructs of the target language.

Transformation Architecture

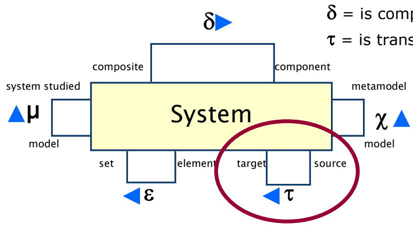


Transformation

- Takes input(s) and produces output(s)
  - One way process
- Composable
- Specialisable
- Classification of transformations
  - "Manual" / Tool Supported
  - Vertical (Example: code generation) / Horizontal (Example: UML2ER)
  - Reversible / Not Reversible
  - Endogenous / Exogenous

Semantics of relations between models

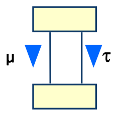
μ = represents
χ = conforms to
ε = belongs to
δ = is composed of
τ = is transformed into



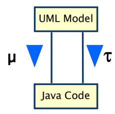
Vertical transformation

μ = represents
χ = conforms to
ε = belongs to
δ = is composed of
τ = is transformed into

- Forward engineering transformation



- Example

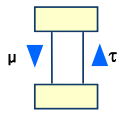


[Favre 2004]

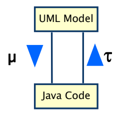
Vertical transformation

μ = represents
χ = conforms to
ε = belongs to
δ = is composed of
τ = is transformed into

- Reverse engineering transformation



- Example

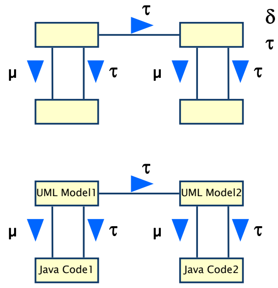


[Favre 2004]

Combination

- Model Driven Evolution

μ = represents
χ = conforms to
ε = belongs to
δ = is composed of
τ = is transformed into



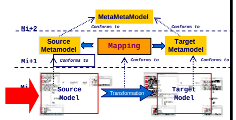
- Example

[Favre 2004]

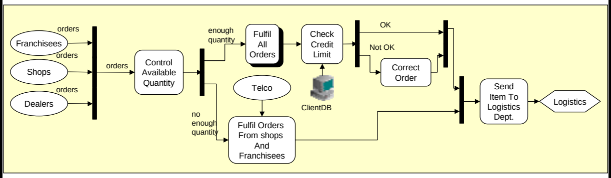
Part 1: Model Transformations

1.1) Definitions: Model, Metamodel
1.2) Transformations and Mappings
1.3) Basic Examples

Objective

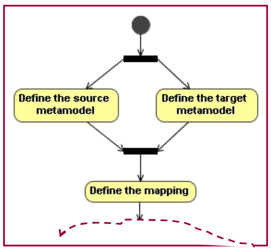


- To build a simple mapping
- From GRAI Extended Actigram to UML Activity Diagram



Method

- First steps



- Overview of the GRAI Extended Actigrams
- GRAI Extended Actigram Metamodel
- UML Activity Diagram Metamodel
- Mapping Proposal

GRAI Methodology

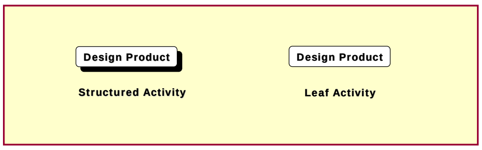
- It is an Enterprise Modelling Methodology
  - Under development since 1980 by the LAP Research Laboratory
- Developed using
  - A set of basic notions of the GRAI conceptual reference model
  - Graphical notations to describe models
- Takes several aspects into account
  - Decision-making
  - Functions/Processes
  - Information

GRAI Methodology: Extended Actigram

- Derived from IDEF0 (**I**CAM **DEF**inition) /SADT
- Built on
  - 3 primary concepts
    - Process
    - Activities
    - Flows
  - 3 secondary concepts
    - Connectors
    - Logical operators
    - Resources

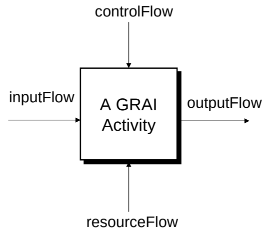
GRAI Process and Activity

- Process
  - Set of logically sequenced activities
- Activity
  - Transformation, production
  - Can be broken down: Structured Activity



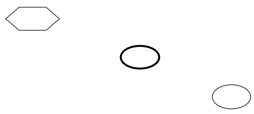
GRAI Flows and Resources

- Different flow types
  - Input
  - Output
  - Control
  - Resource
- Resource
  - Material/Human
- Flow
  - Information/Product

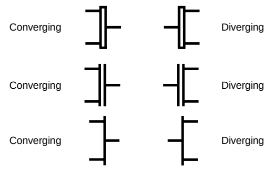


GRAI Connectors and Logical Operators

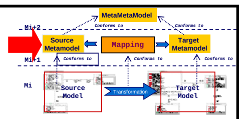
- Connectors
  - Process
  - Internal
  - External

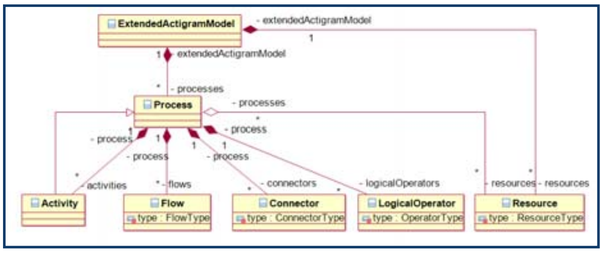


- Logical Operators
  - Synchronous And
  - Asynchronous And
  - OR

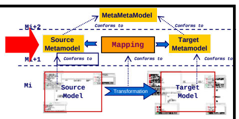


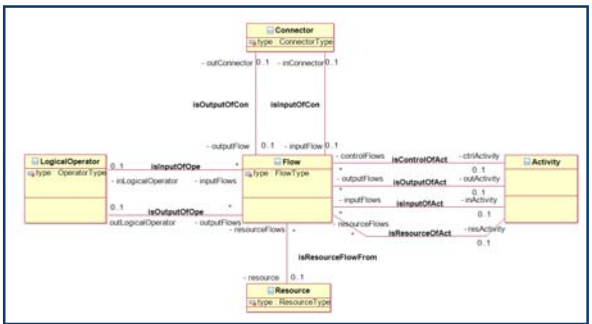
GRAI EA Metamodel: Structure





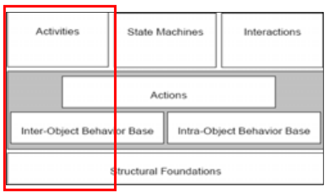
GRAI EA Metamodel: Flow Connections



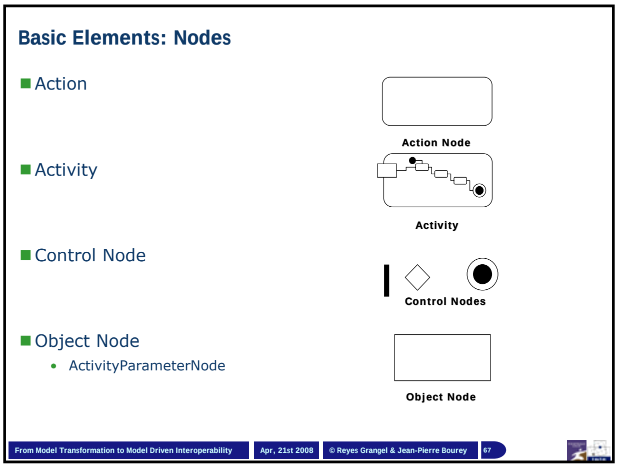


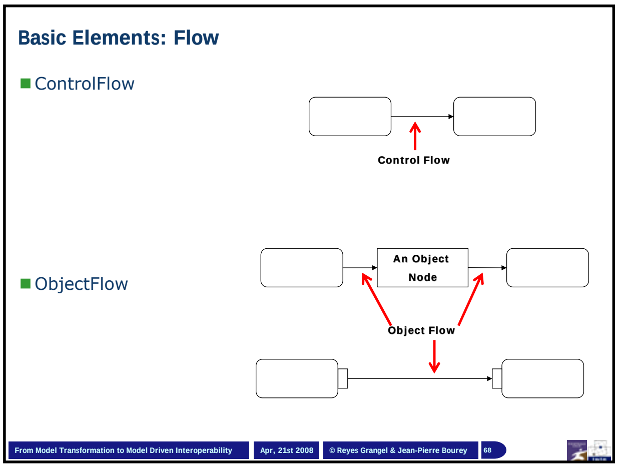
UML Activity Diagram

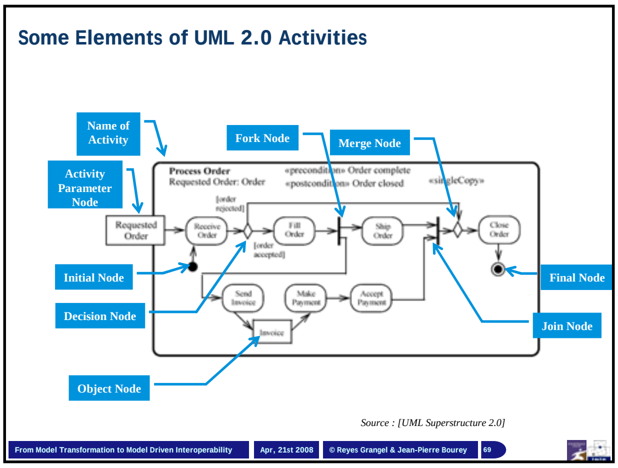
- Used to describe
  - Inter-object behaviour
  - Actions
  - Process

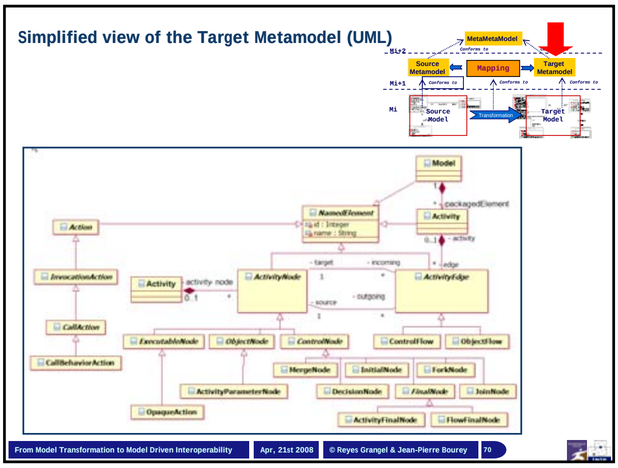


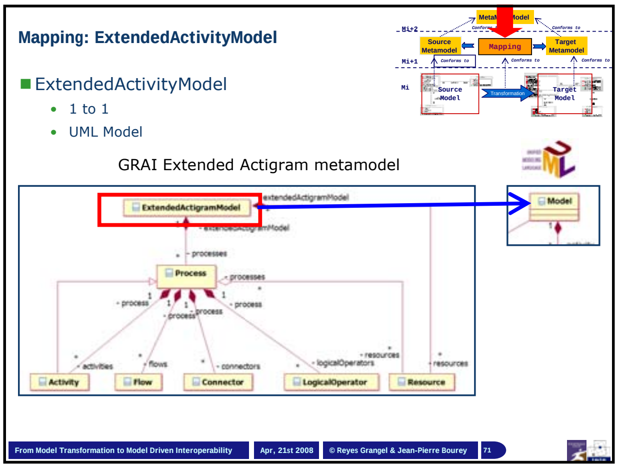
Source : [UML Superstructure 2.0]

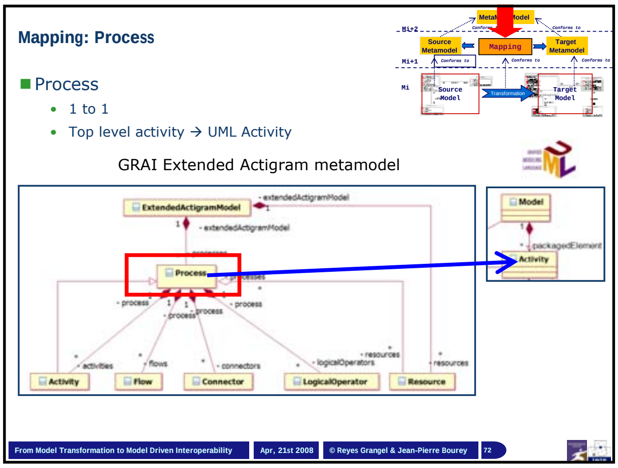

Basic Elements: Nodes
Action
Activity
Control Node
Object Node
ActivityParameterNode
Action Node
Activity
Control Nodes
Object Node
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
67


Basic Elements: Flow
ControlFlow
ObjectFlow
Control Flow
An Object Node
Object Flow
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
68


Some Elements of UML 2.0 Activities
Name of Activity
Activity Parameter Node
Fork Node
Merge Node
Initial Node
Decision Node
Object Node
Final Node
Join Node
Process Order
Requested Order: Order
Source : [UML Superstructure 2.0]
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
69

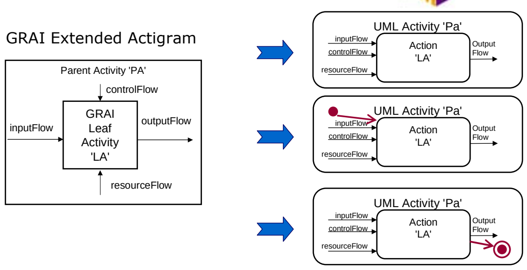

Simplified view of the Target Metamodel (UML)
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
70


Mapping: ExtendedActivityModel
ExtendedActivityModel
1 to 1
UML Model
GRAI Extended Actigram metamodel
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
71


Mapping: Process
Process
1 to 1
Top level activity → UML Activity
GRAI Extended Actigram metamodel
From Model Transformation to Model Driven Interoperability
Apr, 21st 2008
© Reyes Grangel & Jean-Pierre Bourey
72

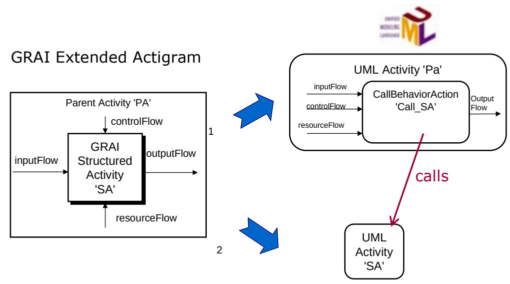
Mapping: Leaf Activities

- Conditional 1 to n
- Leaf Activity
  - UML Action (UML2.0) or OpaqueAction (UML2.1)
  - with optional initial/final elements



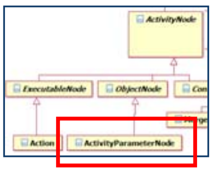
Mapping: Structured Activities

- Structured Activity: several UML candidates
  - UML Structured Activity
  - UML CallBehaviorAction + UML Activity (splitting)



Mapping: Connector, Resource

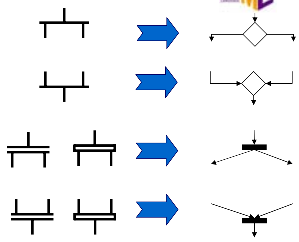
- Connectors and Resources can be considered as "External" from Activity viewpoint
- **Activity Parameter Nodes**
  - Object Node for Inputs and Outputs to Activities
- Connector
  - Many to one:
  - Process, Internal, External connectors => **Activity Parameter Node**
- Resource
  - One to one
  - Resource => **Activity Parameter Node**



Mapping: Logical Operators

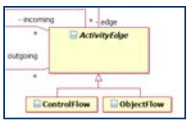
GRAI Extended Actigram

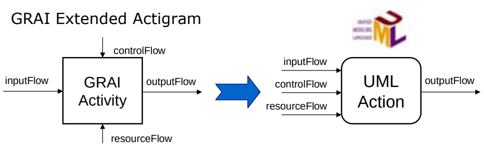
- Diverging OR
  - Decision Node
- Converging OR
  - Merge Node
- Diverging AND
  - Fork Node
- Converging AND
  - Join Node



Mapping: Flows

- Connected to a Resource or Connector
  - ObjectFlow
- Otherwise
  - ControlFlow
- Merge of Input, Resource and Control flows
  - UML input Flow





Mapping Summary

GRAI Extended Actigram	*Condition*	*UML Activity Diagram*
Extended Actigram Model		Model
Process		Activity
Activity	it is a structured Activity	Activity + CallBehaviourAction
	it is a **non** structured Activity	Action
Connector		ActivityParameterNode
LogicalOperator	Converging OR	MergeNode
	Converging AND	JoinNode
	Diverging OR	DecisionNode
	Diverging AND	ForkNode
Resource		ActivityParameterNode
Flow	not connected to a Resource or to a connector	ControlFlow
	connected to a Resource or to a connector	ObjectFlow

Conclusion on the mapping definition

- Detailed knowledge on both metamodels
  - Semantics
    - To find the best appropriated correspondence(s)
  - Syntactic
    - Ensure consistency
    - Ex: mapping of flows
- GRAI EA to UML AD
  - Illustration of different mapping cases
  - Semantic losses
  - Specifies irreversible transformation

Part 2: Transformation Techniques

2.1) From Programming Languages to Model Transformation Languages

2.2) Applications

Transformation Techniques

- Mapping=specifications of transformations
- How to implement these specifications?
- Four main implementation categories

Four categories of model transformation techniques

- General purpose programming languages
  - Java, C++, C#...
- Generic transformation tools
  - Graph transformations, XSLT
- CASE tools scripting languages
  - Objecteering, Rose ...
- Dedicated model transformation tools
  - OMG QVT style , ATL, MTL, MTF...

[Jézéquel 2005]

General purpose programming languages

- Java, VB, C++, C#,... Or your favorite language!
- Currently available in the tools via APIs
- Rules and scheduling implemented from scratch
- Advantages
  - No overhead to learn a new language
  - Tool support to write code (powerful IDE)
- Drawbacks
  - Programming effort
  - Maintenance

Generic Transformation Tools

- awk-like (awk, sed, ...)
  - Limited to 100 LOC
- XSLT to transform XML trees into other (XML) (trees)
  - More batch than interactive
  - Parameters are passed by values
  - More focus on syntax than semantics
  - XSLT transformations are not really easy to maintain (<1000 LOC)
- Graph Transformation Based
  - Declarative, based on theory of graph transformation/rewriting
  - Require careful consideration of termination of the transformation process and the rule application ordering
  - Examples
    - VIATRA2 [VIATRA2 2006],
    - GReAT(Graph Rewriting And Transfomation) [GreAT 2003]

CASE tool scripting languages

- Dedicated to a specific tool
- Example: **Objecteering** (Softeam)
  - J language
- Advanta
  - Comp
  - "Powe
- Drawbac
  - Comp
  - Often
  - Often

```
Package:listClasses ()
{
    OwnedElementClass
// Going through the Package classes
   {
      StdOut.write(NL, "CLASS:", Name);
      PartOperation.<select (Visibility == Public)
      // Going through the method's public classes
      {
         StdOut.write(NL, " Public method:", Name);
      }
   }
} // End of "listClasses
```


Tools dedicated to model transformation

- Declarative, Imperative, Hybrid,
  - **Declarative**: Declarative language describes relationships between variables in terms of functions or inference rules and the language executor (interpreter or compiler) applies some fixed algorithm to these relations to produce a result
  - **Imperative**: Any programming language that specifies explicit manipulation of the state of the computer system
  - **Hybrid**: combination of both
- Rule based
  - Invocation of the transformation rules
    - **Explicit**, via invocation operations (Java like)
    - **Implicit**, based on context and rules signature (Prolog like)

Tools dedicated to model transformation

- Example
  - OMG QVT RFP
  - ATL, MTF, MTL, ...
- Advantages
  - Openness
  - Separation Rule/Engine
- Drawbacks
  - New languages
  - "Prolog" Style
  - "Immaturity"

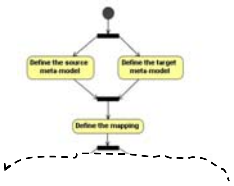
Part 2: Transformation Techniques

2.1) From Programming Languages to Model Transformation Languages

2.2) **Applications**

Application

- Back to the Transformation of GRAI Extended Actigram to UML Activity diagram
- Last steps of the proposed method

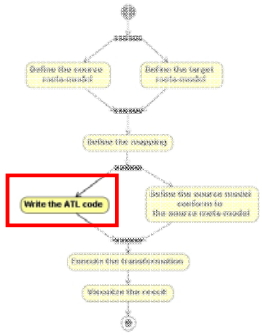


Reminder on the mapping

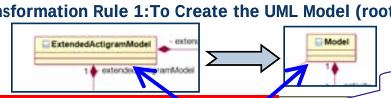
GRAI Extended Actigram	*Condition*	*UML Activity Diagram*
Extended Actigram Model		Model
Process		Activity
Activity	it is a structured Activity	Activity + CallBehaviourAction
	it is a **non** structured Activity	Action
Connector		ActivityParameterNode
LogicalOperator	Converging OR	MergeNode
	Converging AND	JoinNode
	Diverging OR	DecisionNode
	Diverging AND	ForkNode
Resource		ActivityParameterNode
Flow	not connected to a Resource or to a connector	ControlFlow
	connected to a Resource or to a connector	ObjectFlow

Write the ATL Code

- Presentation of one simple rule



Transformation Rule 1:To Create the UML Model (root of the model)




```
rule GraiExtendedActigram2UmlModel{

from
source :
  GraiExtendedActigramMetaModel!ExtendedActigramModel

to
target : UML2!Model (
  name    <- source.name
)-- end of 'to' section

do {
  --Store the Model node in a global variable
  thisModule.Global_UmlModel <- target ;
}-- end of rule 'GraiExtendedActigram2UmlModel'
```

Rule to create the UML model

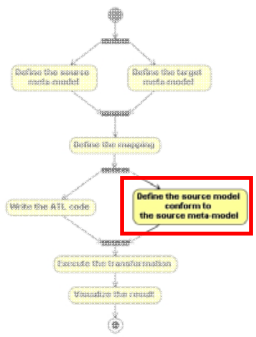
Input of the rule

Output of the rule with attribute setting

Additional Action

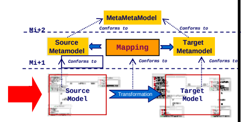
Define the Source Model

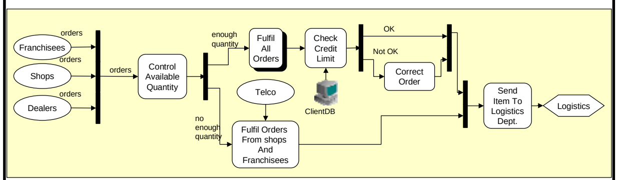
- Presentation of the source GRAI Extended Actigram



The Sales Process Model

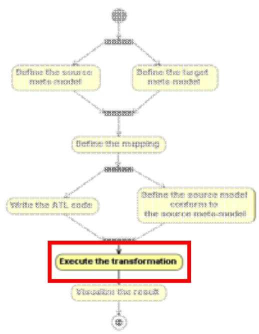
- Model "coding"





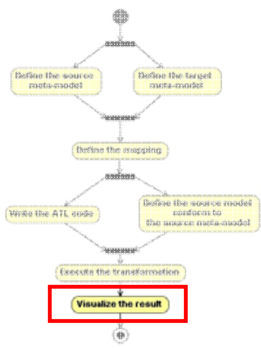
Execute the Transformation

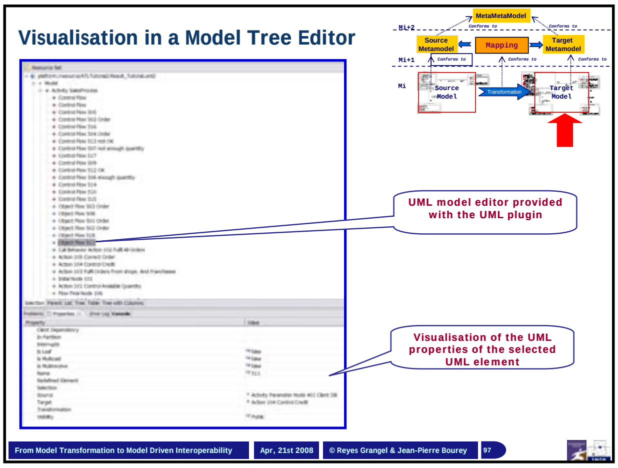
- Within a tool

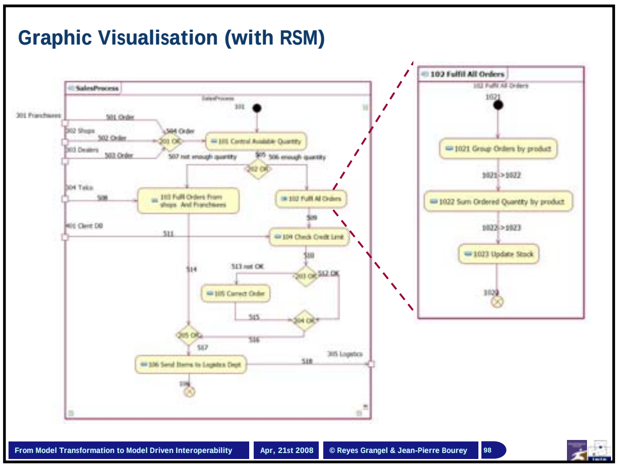


Visualise the Transformation

- "Rough" Visualisation
- Tree Editor Visualisation
- Graphic Visualisation







Part2: Conclusion

Oslo | Apr, 21st 2008

Conclusion and some issues

- Technical/practical presentation of Model Transformation
- Essential to understand the Model Driven approaches issues
- But important problems still remain:
- How to define a metamodel
  - Which level of details?
  - Which point of view?
  - What are the constructs ?
  - What is the semantics?
  - …
- How to define a mapping (Mapping discovery)?
  - How to choose the "good" matching between constructs?
  - How to formalize it?

From Model Transformation to Model Driven Interoperability | Apr, 21st 2008 | © Reyes Grangel & Jean-Pierre Bourey | 100

Part 3: Model Driven Architecture

3.1) Inside Model Driven Architecture
3.2) Abstraction Levels: CIM, PIM, PSM

Oslo | Apr, 21st 2008

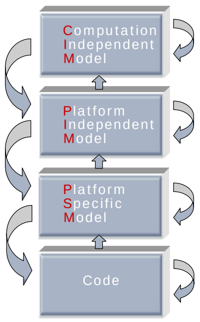
Model Driven Architecture

- Initiative from the software engineering community called "Model Driven Development" (MDD)
- Principle: First develop models, and then transform
- Model Driven Architecture has been developed in this context by the OMG

From Model Transformation to Model Driven Interoperability | Apr, 21st 2008 | © Reyes Grangel & Jean-Pierre Bourey | 102

MDA's Main Components

- MDA specifies 3 viewpoints
  - Computation independent
  - Platform independent
  - Platform specific
- Platform
  - Generic
    - Object, Batch, Dataflow
  - Technology Specific
    - CORBA, Java2Component
  - Vendor Specific
    - .NET, IBM Websphere,...
- =>Separation of concerns
- Model Transformations



Computation Independent Model (CIM)

- Specify business processes, stakeholders, departments, dependencies between processes, and so on
- Represents the system requirements
- Does not show details of the structure of systems
- Domain specific model specified by domain experts
- Is made-up two subdivisions:
  - Business Model
  - Business Requirements for Systems

Platform Independent Model (PIM)

- Shows a high level of abstraction
- Independent of any specific implementation technology= assumed to be executed on technologically independent virtual machine
- Describes the requirements of the software system

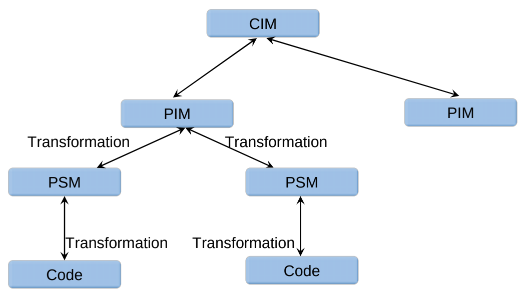
Platform Specific Model (PSM)

- Describes the system in one specific implementation technology
- A PIM is transformed into one or more PSMs
- A separate PSM is generated for each specific technology platform
- Example:
  - PSM for Relational Database (column, foreign key, etc.)
  - PSM for Enterprise Java Beans (entity bean, home interface, session bean, etc.)

Code

- Represents the final step in development
- Generated automatically

The MDA Transformation Steps

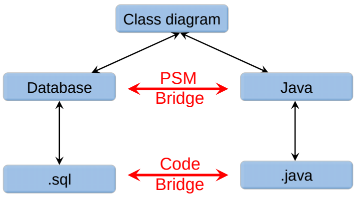


MDA Benefits

- Productivity Gain
  - Minimises the development time
  - More focused on PIM development
  - Better functionality in less time
- Portability Benefits
  - A PIM can be transformed into multiple PSMs
  - The PIM describes a portable specification

MDA Benefits

- Interoperability Benefits

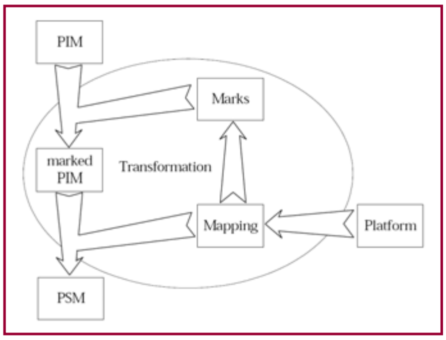


- Documentation Benefits
  - The documentation is kept up to date
  - The PIM represents the high level documentation
  - Consistency between high level of documentation and code is maintained

Some Model Transformation Approaches

- Marking and pattern
  - Mark=a concept of the PSM applied to a PIM element to indicate how to transform it
  - The marked elements of the source are transformed according to the pattern to produce the target.
- Automatic transformation
  - no need of additional information to produce the target
- Metamodel transformation

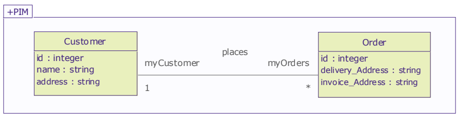
Marking



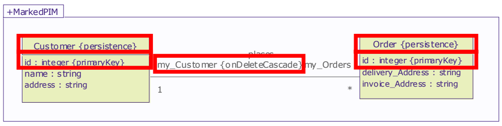
[MDA 2003]

Marking: Example (Objecteering 6.0)

- PIM Model

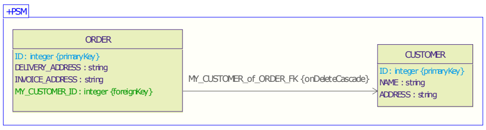


- Marked Model



Marking: Example

- PSM

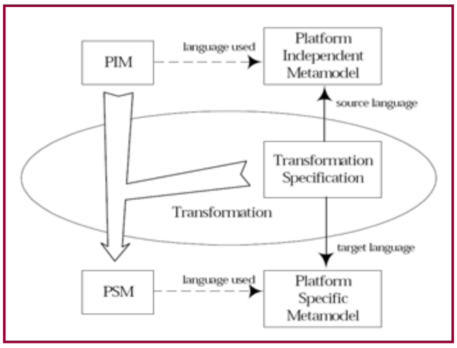


- Code

```
CREATE TABLE ORDER(
   ID                INTEGER  NOT NULL ,
   DELIVERY_ADDRESS VARCHAR2(100) ,
   INVOICE_ADDRESS  VARCHAR2(50) ,
   MY_CUSTOMER_ID   INTEGER  NOT NULL );

ALTER TABLE ORDER ADD (
   CONSTRAINT ORDER_PK PRIMARY KEY (ID)
);
ALTER TABLE ORDER ADD (
   CONSTRAINT MY_CUSTOMER_of_ORDER_FK FOREIGN KEY (MY_CUSTOMER_ID)
      REFERENCES CUSTOMER(ID)
   ON DELETE CASCADE);
```

Metamodel Transformation



[MDA 2003]

The MDA Paradigm Shift

- Main goal
  - To perpetuate the development efforts concerning rapidly changing technologies
- A real shift of focus from code to models where code will be generated automatically
- Consequences on:
  - Software development roles (PIM analyst, …)
  - Programming competencies->Modelling competencies
  - Tools (new IDE, model compiler)
- **A**rchitecture **D**riven **M**odernization (ADM) [Van Den Heuvel 2006]
  - Reverse Engineering
- **Are we at the beginning of a new era in software development??**

Last Words on MDA

"**Prediction** is very difficult, …

…especially about the **future**".

Niels Henrik David Bohr

(October, 7th, 1885 – November, 18th, 1962).

Danish physicist

Nobel Prize in 1922 in physics

"for his services in the investigation of the structure of atoms and of the radiation emanating from them"

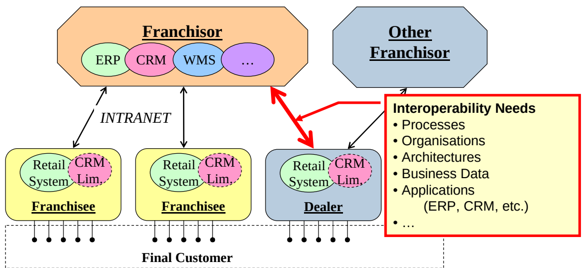
Part 4: Model Driven Interoperability

Oslo | Apr, 21st 2008

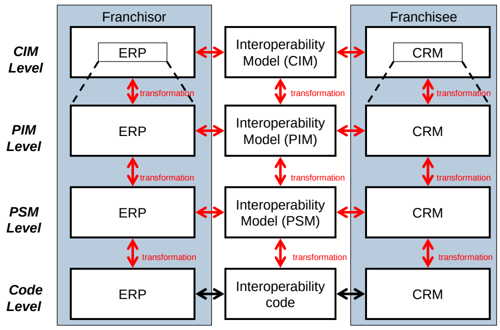
Interoperability concept

- **IEEE**
  - Ability of two or more systems or components to exchange information and to use the information that has been exchanged
- **IDEAS**
  - Ability of Enterprise Software Applications (ESA) to interact with each other
- **INTEROP NoE**
  - Is achieved if the interaction takes place on all layers of enterprises, that is the **business**, **knowledge**, and **ICT** levels, and **semantics** can also be used to accomplish a common understanding among collaborative enterprises
- **Task Group 2 (TG2)**
  - Focused on interoperability problems that enterprises face when it comes to generating ESA from enterprise Models

Case Study Description

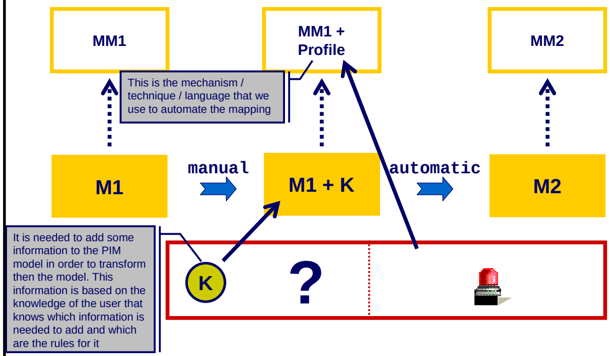


MDI Framework and Transformations



From Model Transformation to Model Driven Interoperability | Apr, 21st 2008 | © Reyes Grangel & Jean-Pierre Bourey | 121

General Approach to Transform Models



From Model Transformation to Model Driven Interoperability | Apr, 21st 2008 | © Reyes Grangel & Jean-Pierre Bourey | 122

Conclusion

Oslo | Apr, 21st 2008

Conclusion

- Model transformations are technologically feasible
- Model transformations can help to solve interoperability problems
- **Main problem** is how to add the required knowledge to perform transformations
- Adding semantic support is technologically feasible
- **Main future researches**: how to combine model transformation and semantic support from both methodological and technological viewpoints?
- MDI Method is a first step from methodological viewpoint

From Model Transformation to Model Driven Interoperability | Apr, 21st 2008 | © Reyes Grangel & Jean-Pierre Bourey | 124

Very Last Word

"Nothing is created or lost. Everything is transformed, including ideas"

Antoine-Laurent de Lavoisier (August 26, 1743 – May 8, 1794)

alias "the father of modern chemistry"
Chemist, Philosopher, Biologist, Economist
Stated the first version of the Law of conservation of mass
Named oxygen and hydrogen
Introduced Metric system
Invented the first periodic table including 33 elements

From Model Transformation to Model Driven Interoperability | Apr, 21st 2008 | © Reyes Grangel & Jean-Pierre Bourey | 125

References

- [Allilaire 2004]
  - Proceedings of the Second European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations (EWMDA-2), edited by D. H. Akehurst. Computing Laboratory, University of Kent, Canterbury, Kent CT2 7NF, UK, Canterbury, England, pages 171--178. http://www.sciences.univ-nantes.fr/lina/atl/www/papers/ADT_AllilaireIdrissi.pdf
- [Atlas 2005]
  - ATLAS group LINA & INRIA Nantes, KM3:Kernel MetaMetaModel Manual- version 0.3 - http://www.eclipse.org/gmt/atl/doc/KernelMetaMetaModel%5Bv00.06%5D.pdf
- [Bézivin 2005]
  - Jean Bézivin & Marcos Didonet del Fabro, Ingénierie des modèles: des principes aux outils,
- [Davies 2003]
  - Islay Davies, Peter Green, Simon Milton, and Michael Rosemann, "Using Meta Models for the Comparison of Ontologies", EMMSAD'03, http://www.emmsad.org/2003/Final%20Copy/23.pdf
- [Ernst 2003]
  - Johannes Ernst. What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model? http://www.metamodel.com/article.php?story=20030115211223271
- [Favre 2004]
  - Favre J.M., "Towards a Basic Theory to Model Driven Engineering", 3rd Workshop in Software Model Engineering, WiSME 2004, http://www-adele.imag.fr/~jmfavre
- [Gasevic 2006]
  - Dragan Gaevic, Dragan Djuric, Vladan Devedic,"Model Driven Architecture and Ontology Development", Publisher: Springer, ISBN-10: 3540321802 July 2006
- [GreAT 2003]
  - Gabor KARSAI, Aditya AGRAWAL, Feng SHI et Jonathan SPRINKLE. On the Use of Graph Transformation in the Formal Specication of Model Interpreters. J. UCS, 9(11):1296.1321, 2003.

From Model Transformation to Model Driven Interoperability | Apr, 21st 2008 | © Reyes Grangel & Jean-Pierre Bourey | 126

References

- [Gruber 1993]
  - Gruber T.R., "A translation approach to portable ontology specifications", Knowledge Acquisition, vol. 5, no 2, pp. 199-220
- [INRIA 2006]
  - http://modelware.inria.fr/article72.html
- [Jézéquel 2005]
  - http://modelware.inria.fr/rubrique21.html
- [Kermeta 2006]
  - Franck Fleurey, Zoé Drey, Didier Vojtisek, Cyril Faucher, Kermeta language Reference manual, http://www.kermeta.org/docs/KerMeta-Manual.pdf
- [Kleppe 2003]
  - Anneke Kleppe, Jos Warmer, Wim Bast ; MDA Explained : the model driven Architecture : practice and promise, Addison-Wesley , 2003 . Collection : The Addison-Wesley object technology series , ISBN 0-321-19442-X
- [Metacase 2004]
  - http://www.metacase.com/papers/ABC_to_metaCASE_in_French.pdf
- [MDA 2003]
  - MDA Guide Version 1.0.1. Object Management Group, document number: omg/2003-06-01 edition, June 2003
- [MoMo 2006]
  - TG3 MoMo: Toolbox definition and Workshop report DTG3.3, http://interop-noe.org/
- [ORMSC 2005]
  - A Proposal for an MDA Foundation Model, An ORMSC White Paper, V00-02, ormsc/05-04-01, http://www.omg.org/docs/ormsc/05-04-01.pdf
- [S. J. Sewall 2003]
  - Stanley J. Sewall, Executive Justification for Adopting Model Driven Architecture (MDA),, 11/2003, http://www.omg.org/mda/mda_files/11-03_Sewall_MDA_paper.pdf
- [Van Den Heuvel 2006]
  - Willem-Jan van den Heuvel, Matching and Adaptation: Core Techniques for MDA-(ADM)-driven Integration of new Business: Applications with Wrapped Legacy Systems, http://www.cis.uab.edu/EDOC-MELS/Papers/Willem-Jan.pdf
- [VIATRA2 2006]
  - András BALOGH et Dániel VARRÓ. Advanced Model Transformation Language Constructs in the VIATRA2 Framework. Dans Proceedings of ACM Symposium on Applied Computing (SAC 06), model transformation track, pages 1280.1287, Dijon, France, 2006.

GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others. This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software. We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

GNU Free Documentation License

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

GNU Free Documentation License

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque". Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition. The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

GNU Free Documentation License

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3. You may also lend copies, under the same conditions stated above, and you may publicly display copies.

GNU Free Documentation License

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects. If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

GNU Free Documentation License

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

GNU Free Documentation License

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

GNU Free Documentation License

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert orimply endorsement of any Modified Version.

GNU Free Documentation License

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

GNU Free Documentation License

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit.

When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form.

Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

GNU Free Documentation License

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.