



Automating Metamodel Mapping using Machine learning

Presented by: Jamal Abd-Ali

Co-author: Karim El Guemhioui

University of Quebec in Outaouais



Outline

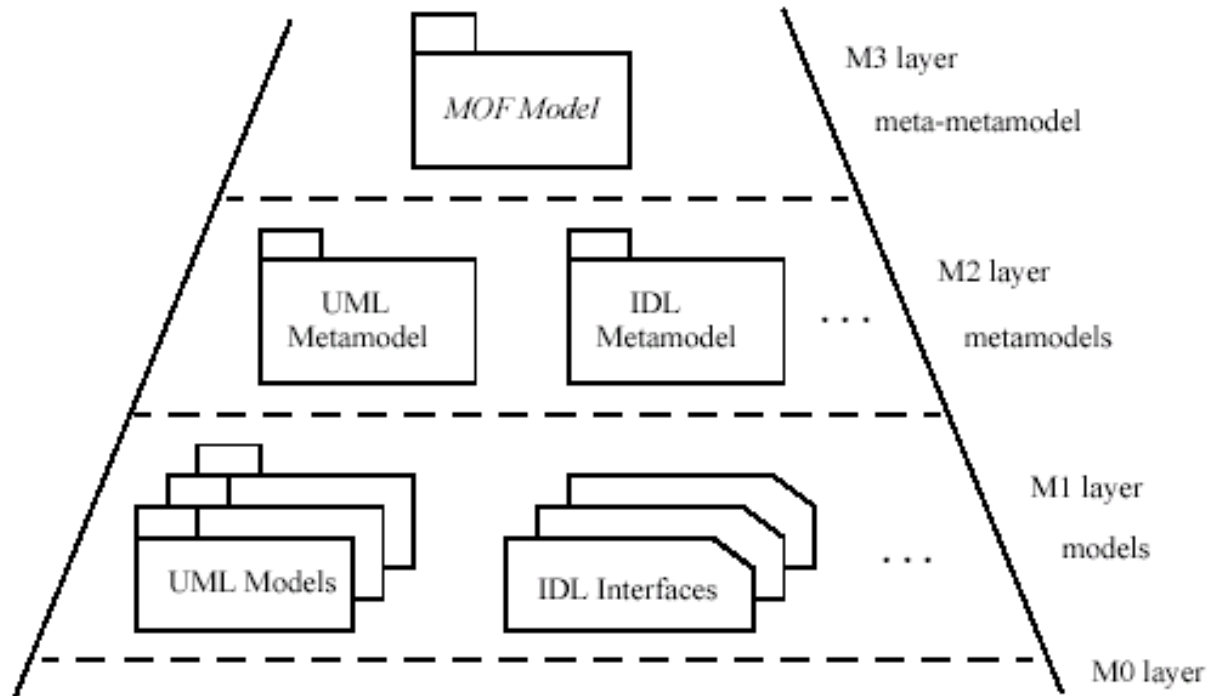
- **Context: MDA Transformation**
 - Transformations and metamodels
- **Hypotheses and knowledge base**
 - Analogy with translation of Natural languages
- **Possible Machine learning techniques**
- **Proposed algorithm**
- **An illustration example**
- **Conclusion**



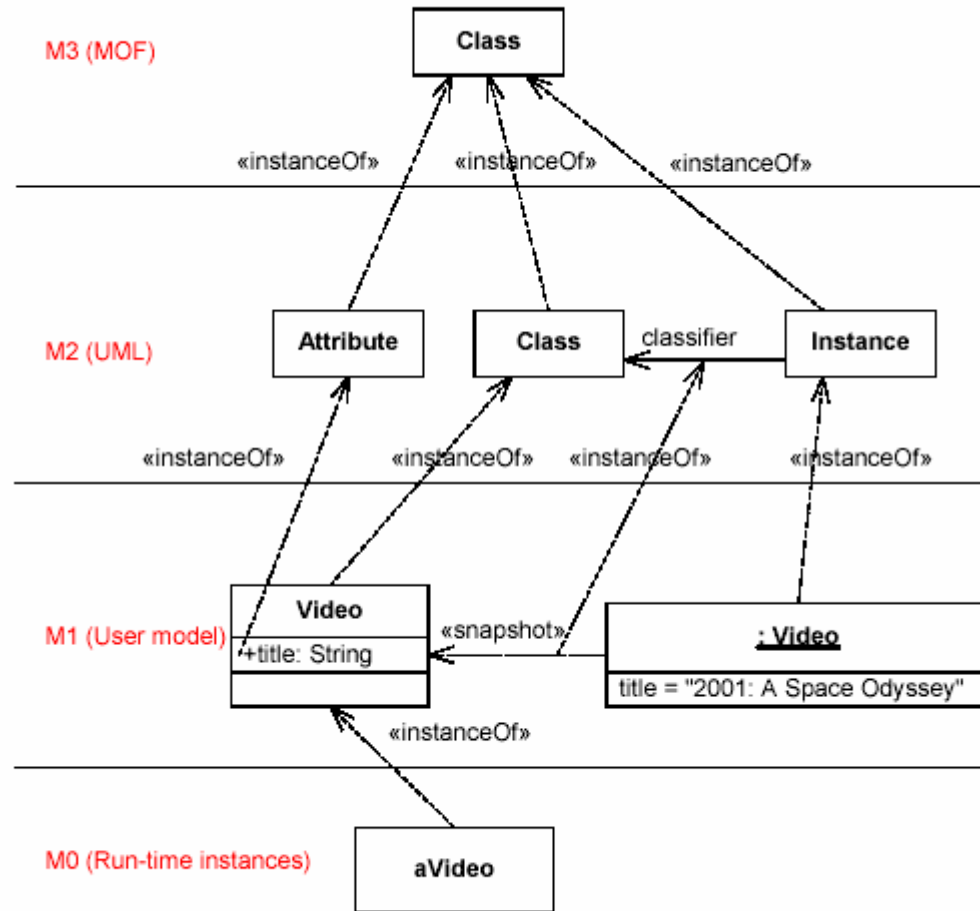
MDA Context

- Model transformation in an MDA framework:
 - PIM to PSM
 - PSM to PSM
- Input and output model represent the same system
 - The transformation rewrites the same system in other “words” (in other metamodel elements)
 - In an ideal world the transformation preserves the meaning of the input model
- Perfect transformation depends on:
 - Source and target Metamodels’ expressiveness
 - The mapping transformation rules

OMG modeling levels



An example of the four-layer metamodel hierarchy



Metamodels compared with natural languages

Natural languages	Metamodels
Huge number of words	Limited number of metamodel elements
Grammar: a set of rules that evolve continuously	Grammar: from metamodel structure and constraints
A text is a sequence of words	A model is a structured set of metamodel elements instances
Many different meanings for a single word with evolving new usage of the word	A defined meaning for any instance of a metamodel element



How to facilitate the elaboration of transformation rules?

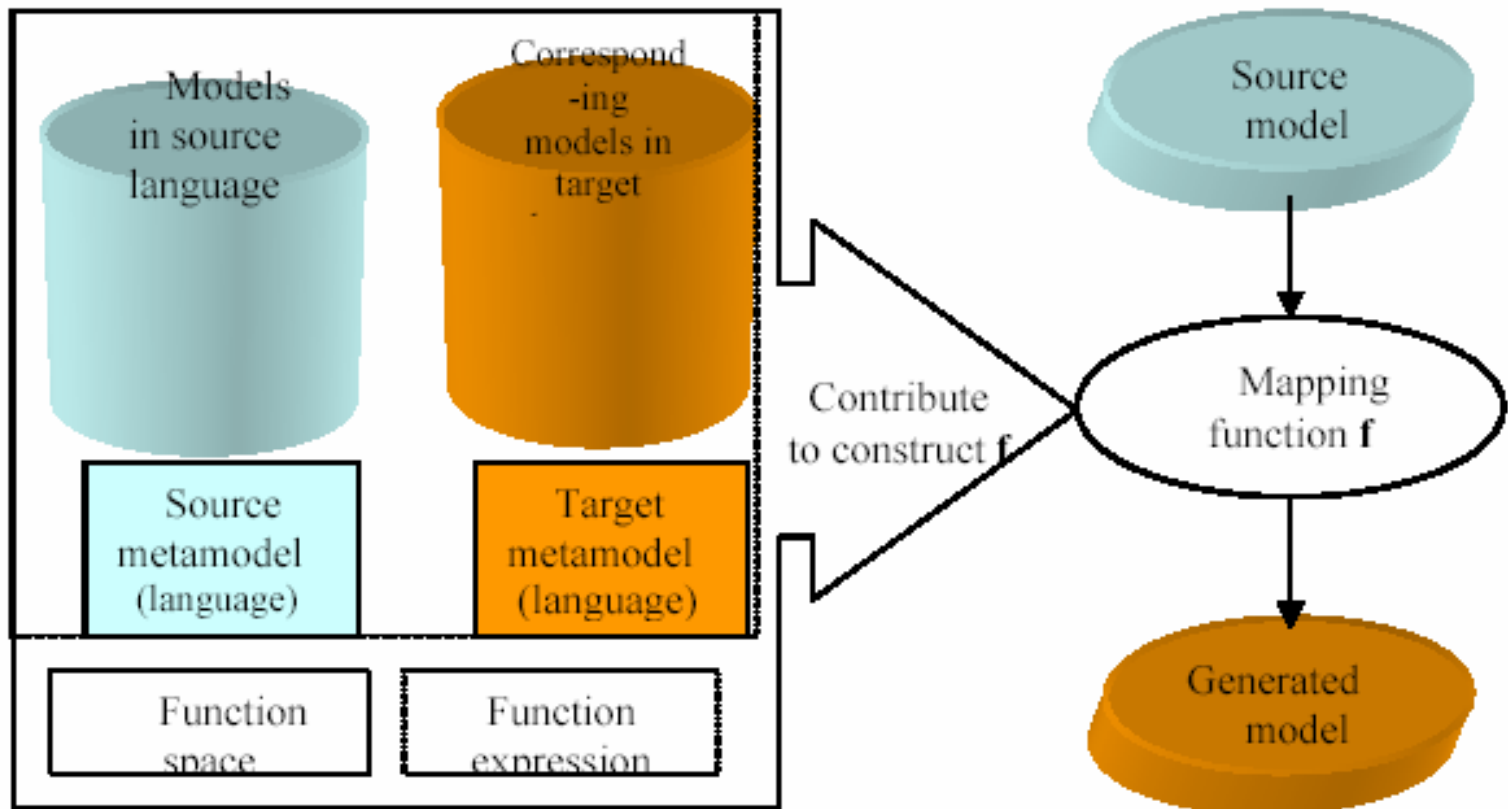
- Migrating from one platform to another implies defining model transformations
- Defining model transformation is a labour intensive and error prone activity
- Among the several approaches to elaborate the transformation rules, we are interested in exploring Machine Learning.



The Transformation Definition and Machine Learning

- We assume that we have a knowledge base consisting of a large number of corresponding models written in two metamodels.
- Machine learning technique :
 - Use the knowledge base to infer mapping rules
- Analogy with using Machine Learning in natural languages translation

Machine Learning context





Machine Learning techniques

- **Connectionist**

- A network of elements that change their structure and weights

- **Emergent learning**

- We start with a population of solutions of the domain problems

- **Symbol Based**

- We consider a set of symbols that represents the entities and relationships of a problem domain



Symbol Based Approach

- The transformation is a function f
 - A space H of Hypotheses on the sought after transformation function f :
 - H results from our knowledge about f characteristics.
 - f is a pattern mapping function.
 - The LH : the formalism expressing the function.
 - a vector representation of a pattern mapping.



The formalism expressing the function of the hypotheses space

- We consider finite metamodels MM1 and MM2.
- Metamodels elements are e_i ($i \leq k$) and e_j ($j \leq h$).
- V is a vector of $k+h$ Boolean elements :

$$V = [\alpha_1, \alpha_2, \dots, \alpha_k, \beta_1, \dots, \beta_h]$$

- An element e_i of MM1 participates in the pattern mapping represented by V iff α_i is true.
- An element e_j of MM2 participates in the pattern mapping represented by V iff β_j is true.



The Candidate elimination algorithm

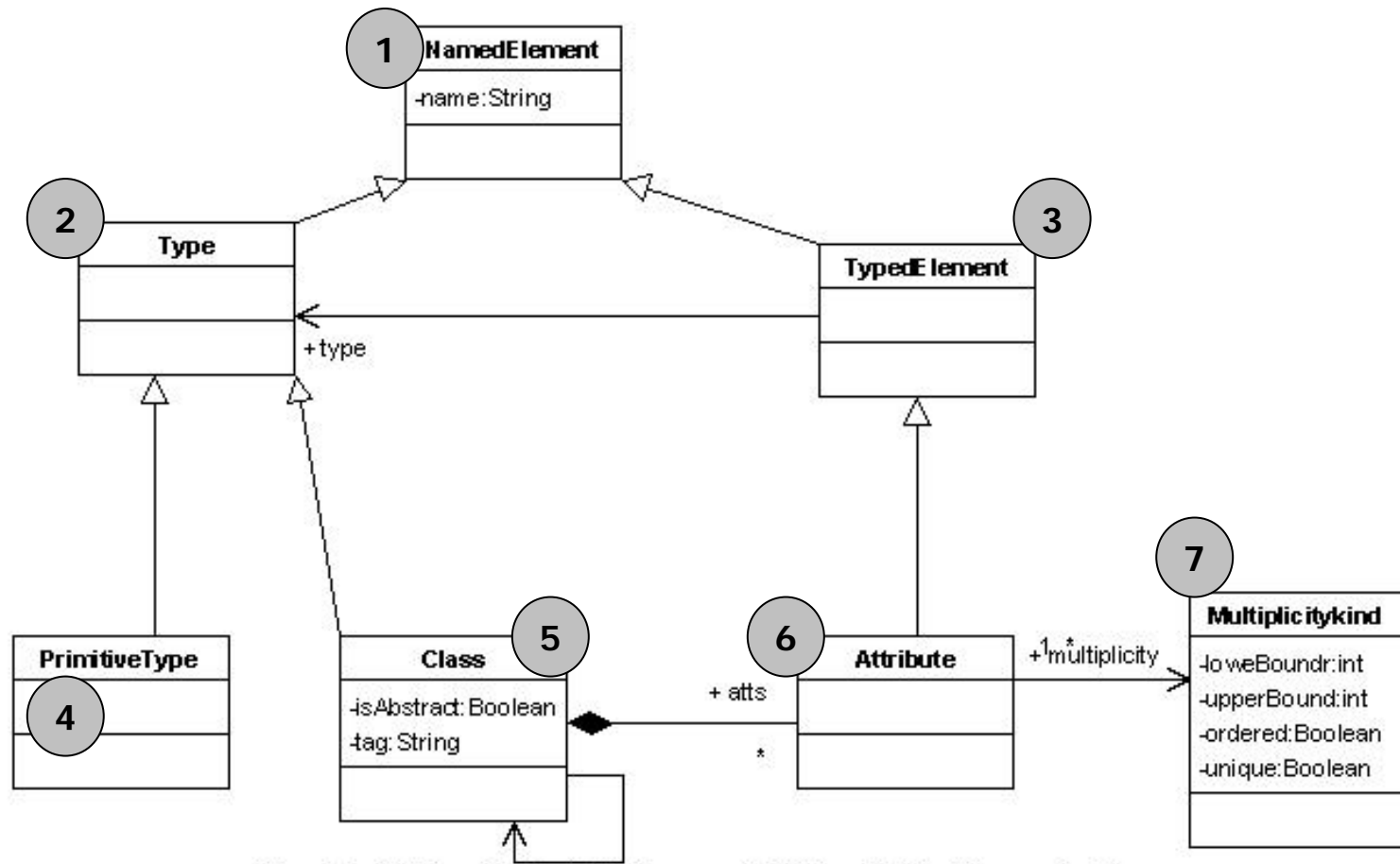
- We consider the set of all possible pattern mappings: a set of vector V .
- Credit vectors consistent with the training example.
- Vectors non consistent with the training example are candidates for elimination.
- For each vector V we can count positive and negative examples.



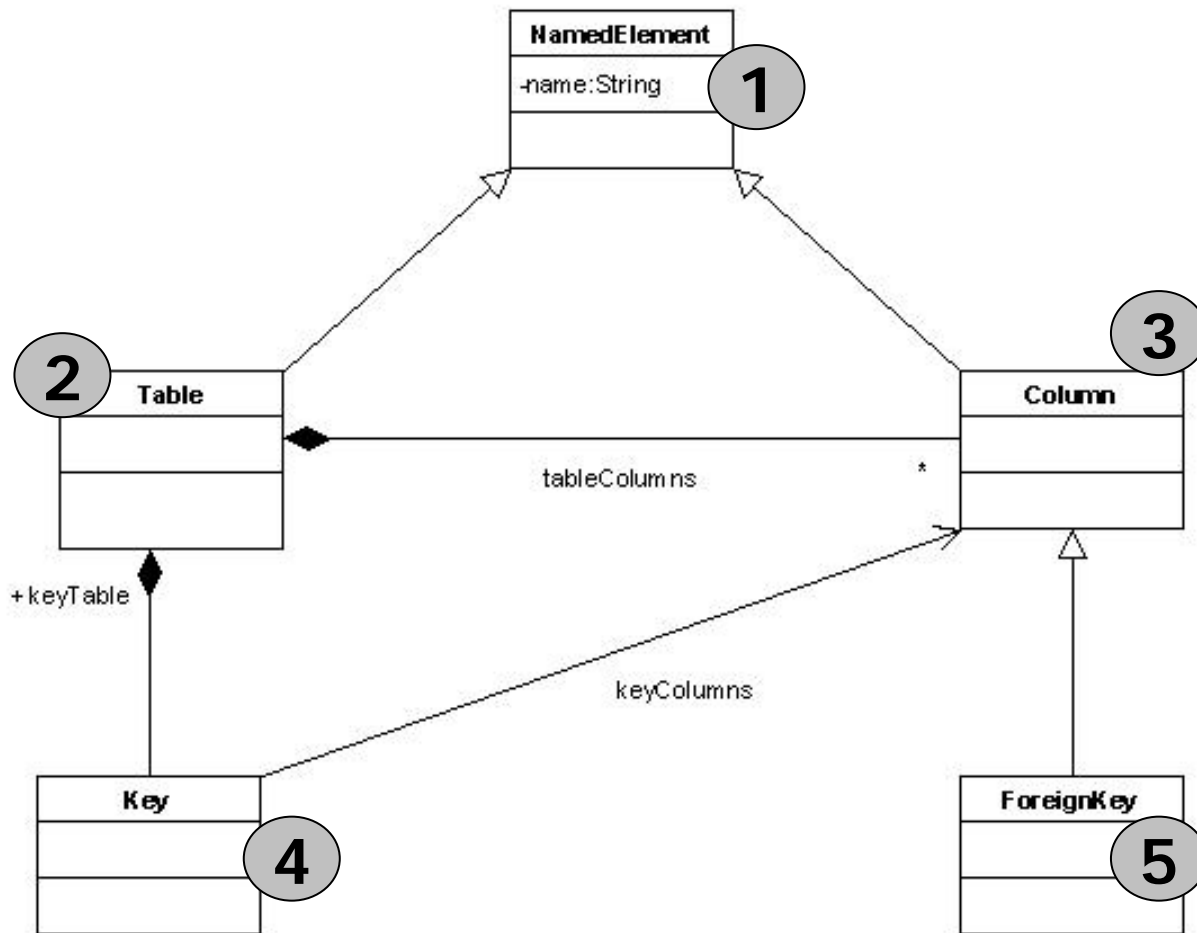
The Candidate elimination algorithm

- We can define strategies for detecting matching elements in a couple of models (ex. same name for named elements)
- Adding some restrictions to the sought after mapping helps avoid combinatorial explosion
- Associations can be considered as elements of the vector V

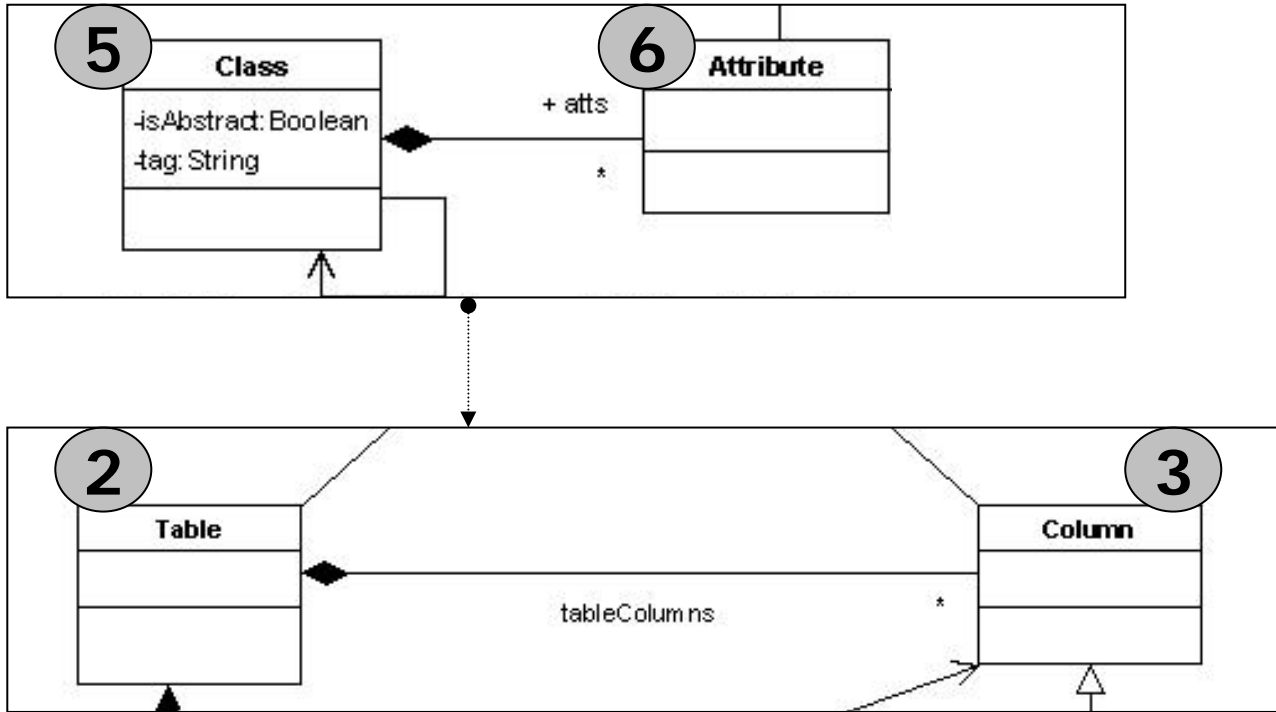
Illustration example with the OO and the relational metamodels



The relational metamodel



A vector V of a pattern mapping



The resulting Vector V =
[0,0,0,0,1,1,0, 0,0,1,1,0,0]
 Object-oriented , Relational
 1,2,3,4,5,6,7- 1,2,3,4,5



Conclusion

- We can use Machine Learning technique to:
 - Assess given transformation rules
 - Improve existing transformation rules
 - In absence of experts in the two considered metamodels, we can extract rules that are unpredictable by experts of each domain
- We expect:
 - The elaboration of an advanced hypotheses space and a formalism expressing the mapping rules.
 - Better results compared with natural languages translation.
 - Limited number of elements
 - Finite and non evolving grammar (structure)



Thanks for your attention

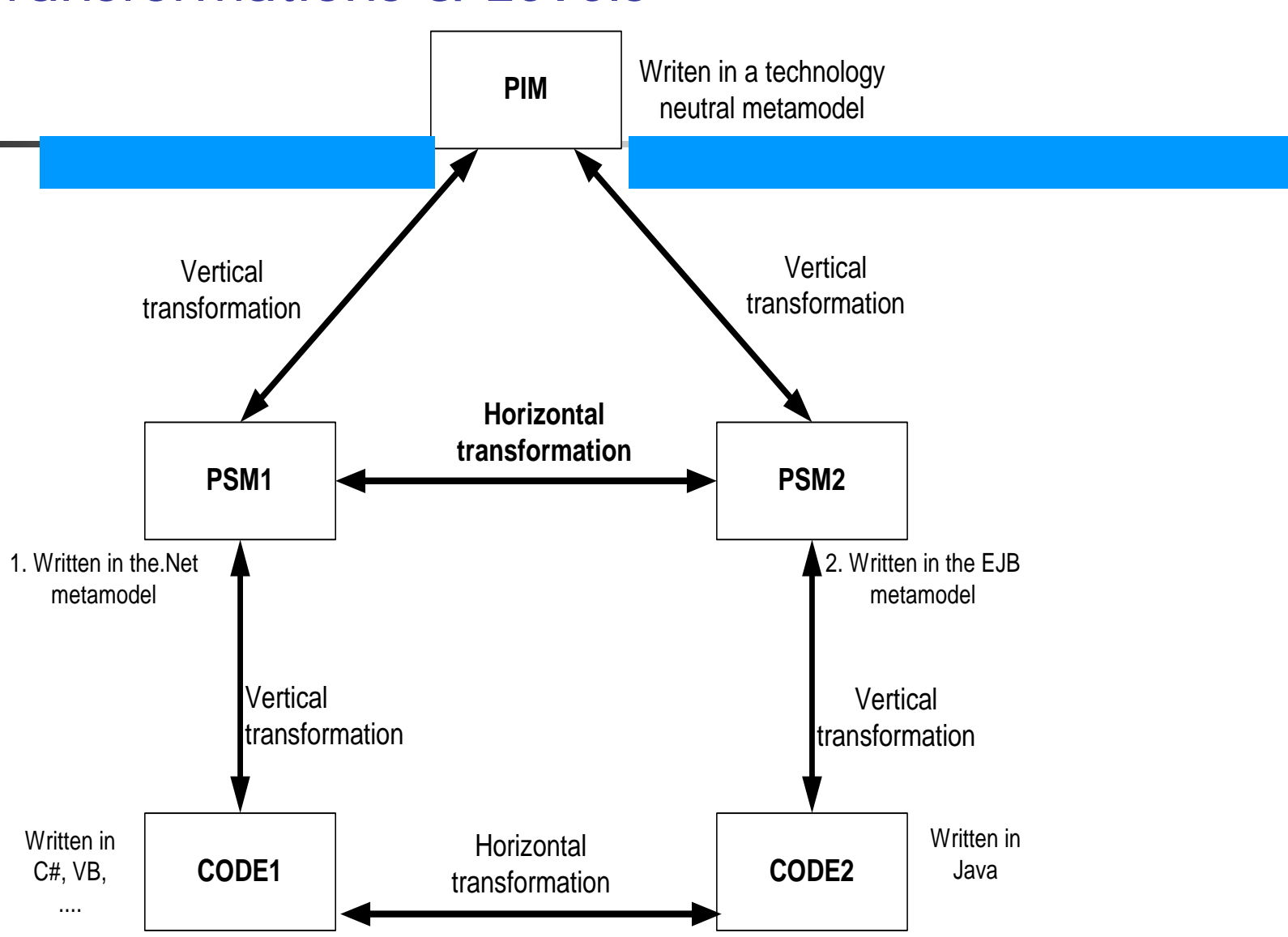
Questions

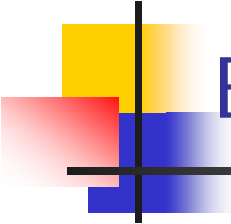


Backup Slides

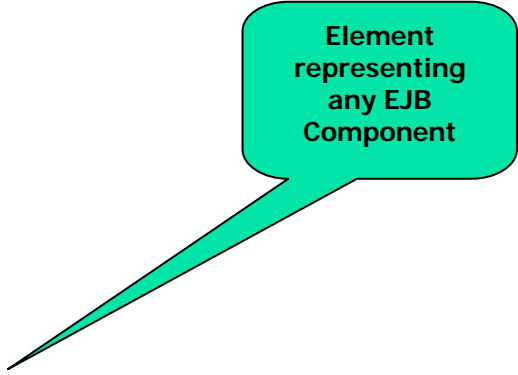
Backup Slides follow this slide

Transformations & Levels





Target & source metamodels: EJB metamodel / main diagram



Element
representing
any EJB
Component



References

- [EDOC] J. Abd-Ali and K. El Guemhioui. (2005) "An MDA-Oriented .NET Metamodel". Ninth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2005), Enschede, The Netherlands. 1541-7719/05:142-153.(travaux de maîtrise).
- [1] Anneke Kleppe, Jos Warmer and Wim Bast. 2002. *MDA Explained , The Model Driven Architecture : Practice And Promise*. Addison-Wesley, 170 p.
- [2] Object Management Group. Request for Proposal: MOF 2.0 Query / View / Transformations RFP". OMG 2002. <http://www.omg.org/docs/ad/02-04-10.pdf>.
- [3] OMG: MDA GUIDE Version 1.0.1 document No omg/2003-06-01 disponible @ <http://www.omg.org/docs/omg/03-06-01.pdf>
- [4] OMG: Metamodel and UML Profile for Java and EJB Specification. February 2004. Version 1.0, formal/04-02-02. Available at <http://www.omg.org/docs/formal/04-02-02.pdf>
- [5] Sheena R. Judson, Robert France. Model Transformations at the Metamodel Level. Published in Workshop in Software Model Engineering octobre,2003. Available at <http://www.metamodel.com/wisme-2003/19.pdf>
- [6] G. Rozenberg (ed.); "Handbook of graph grammars and computing by graph transformation: Volume I Foundations". World Scientific Publishing, 1997.
- [7] Software Modeling and Verification Lab.; "Genermorphous home page". <http://cui.unige.ch/smv/gmorph>



References

[9] XMOF. Queries, Views and Transformations on Models using MOF, OCL and Patterns. Proposal to document: MOF 2.0 Query / Views / Transformations RFP ad/2002-04-10

<http://www.omg.org/docs/ad/03-08-07.pdf>

[10] Response to the MOF 2.0 Query/Views/Transformations RFP (ad/2002-04-10)
Revised submission, version 1.0 August 18, 2003, OMG Document as/2003-08-05

<http://www.omg.org/docs/ad/03-08-05.pdf>

[11] Edward Willink, "The UMLX Language Definition", <http://www.eclipse.org/gmthome/doc/umlx/umlx.pdf>.

[12] DSTC, IBM, "MOF Query/Views/Transformations, Initial Submission", OMG Document ad/2003-02-03,
<http://www.dstc.edu.au/Research/Projects/Pegamento/publications/ad-03-02-03.pdf>.

[13] QVT Partners, "Initial submission for MOF 2.0 Query/Views/Transformations RFP", OMG Document
ad/2003-03-27, <http://www.qvtp.org/downloads/1.0/qvtpartners1.0.pdf>.

[14] Aditya Agrawal, Gabor Karsai, Feng Shi, "A UML-based Graph Transformation Approach for Implementing
Domain-Specific Model Transformations",

http://www.isis.vanderbilt.edu/publications/archive/Agrawal_A_0_0_2003_A_UML_base.pdf

[15] Jean Bézevin, Erwan Breton, Grégoire Dupé, Patricx Valduriez, "The ATL Transformation-based Model
Management Framework", submitted for publication.

[16] OMG: MetaObjectFacility(MOF) Specification.

<http://www.omg.org/docs/formal/02-04-03.pdf>

[17] OMG: UML Profile for EDOC.

http://www.omg.org/technology/University_of_Quebec_in_Outouais/#UML_for_EDOC

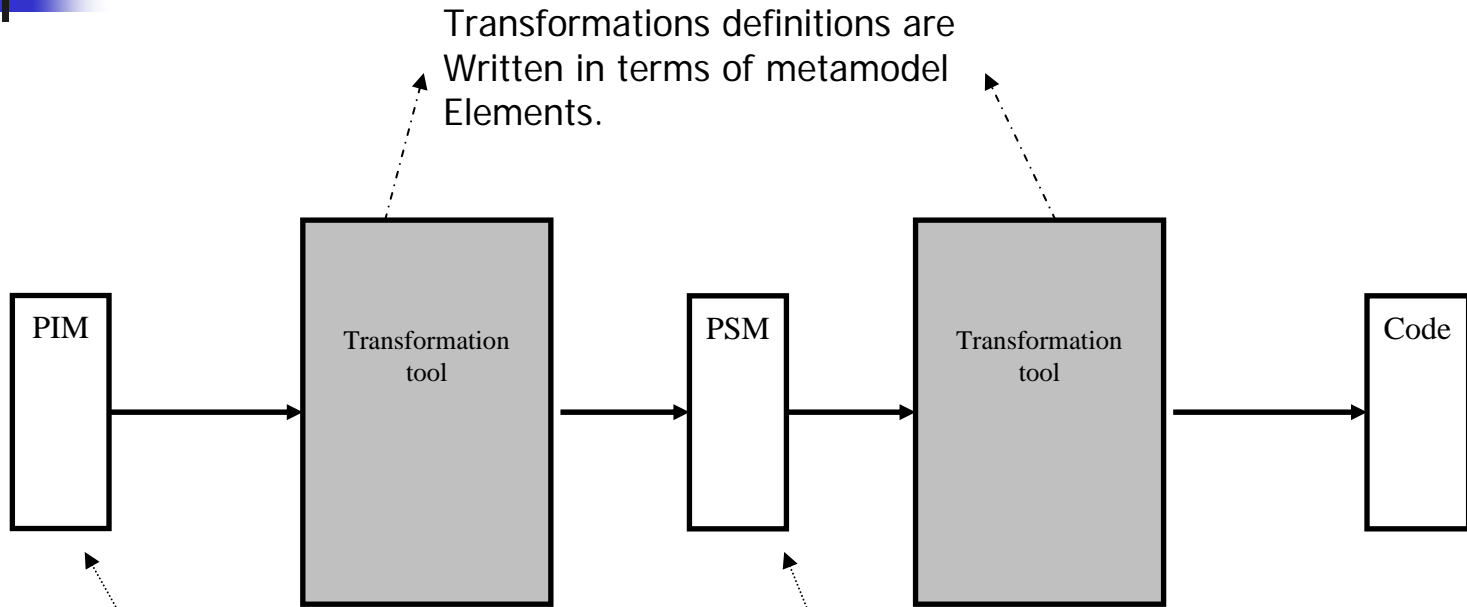


References

- [18] Sun Microsystems, Enterprise Java Beans. <http://java.sun.com/products/ejb/>
- [19] OMG: UML Profile for Enterprise Application Integration (EAI)
http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML for EAI
- [20] OMG: XMLMetadata Interchange (XMI)Specification
<http://www.omg.org/docs/formal/02-01-01.pdf>
- [21] OMG: Unified Modeling Language Specification Version 1.5. <http://www.omg.org/docs/formal/03-03-04.pdf> <ftp://ftp.omg.org/pub/docs/formal/00-03-01.pdf>
- [22] The ATL language definition Web page available @ <http://www.sciences.univ-nantes.fr/lina/atl//atlProject/languageDefinition/>
- [23] Christophe Calandreau, Alai Fauré et Nader Soukouti. 2002. *EJB 2.0 Mise en œuvre*. Paris : Dunod, 412 p.
- [24] The Microsoft Developer Network (MSDN) available at <http://msdn.microsoft.com/library/default.asp>
- [25] OMG : OCL Response to the UML 2.0 OCL RfP (ad/2000-09-03)
Revised Submission, Version 1.6. January 6, 2003. OMG Document ad/2003-01-07
<http://www.omg.org/docs/ad/03-01-07.pdf>
- [26] OMG / MOF 2.0, Query / Views / Transformation, ad/2002-04-10, Revised Submission, Version 1.0, 2003/08/18, OpenQVT, available at <http://www.omg.org/docs/ad/03-08-05.pdf>

MDA Lifecycle

[1]

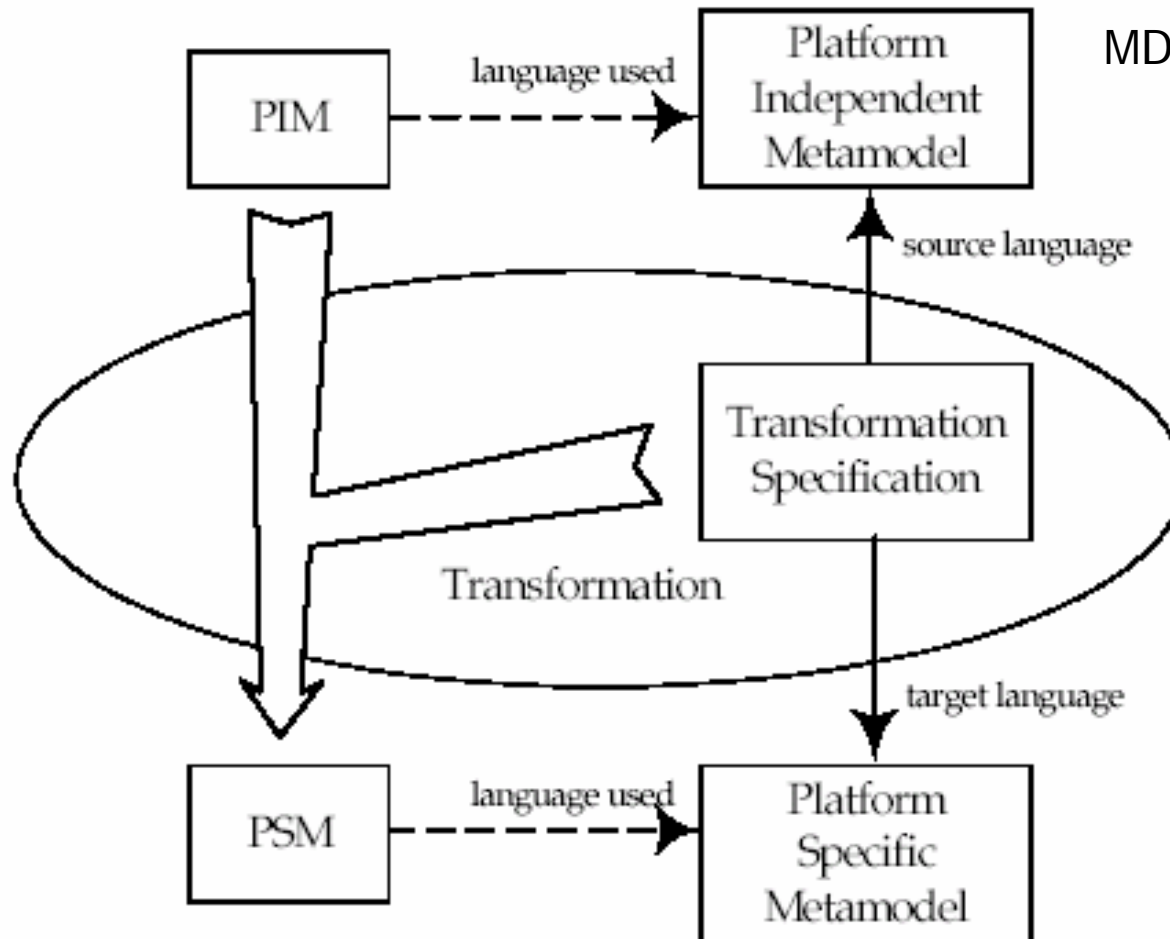


Transformations definitions are
Written in terms of metamodel
Elements.

PSM is a Platform Independent Model.
One metamodel is enough as a language
to write PIMs

PSM is a Platform Specific Model.
For each platform (J2EE, .Net, ...)
a metamodel is needed as a language
to write PSMs

Transformation & Metamodels



MDA Guide of OMG

The transformation depends on expression languages of source and target models



Metamodel characteristics

- Subsuming the technology
 - High level concepts: component interface (exhibited methods), host application.
 - Covering components implementation aspects like channels, ports, url, uri, activation modes, .Net Attributes, pooling policy, transaction management, synchronization, etc.
 - Capture the structure and interaction between the above elements
- Of intermediate complexity
- No redundancy
- Written in UML notations and complying with MOF ⇒ ease of integration in most well known tools like EMDF and ADT of Eclipse, Mia-Transformation, etc.



Use of The Metamodel

- Describes domain specific (PSMs) models
 - Ex. ServicedComponent (SC) instance expresses all the semantics and structure of SC
 - The metamodel gives the appropriate vocabulary and structure to .NET specific models
- Writes model transformation rules, ex.:
 - A session instance of EJB can be mapped to a ServicedComponent instance
 - A MethodTransaction EJB instance can be translated to a .NET TransactionAttribute
- Helps to elaborate a specific code generator