PRINCÍPIOS DE DESENVOLVIMENTO DE SOFTWARE DIRIGIDO A MODELOS

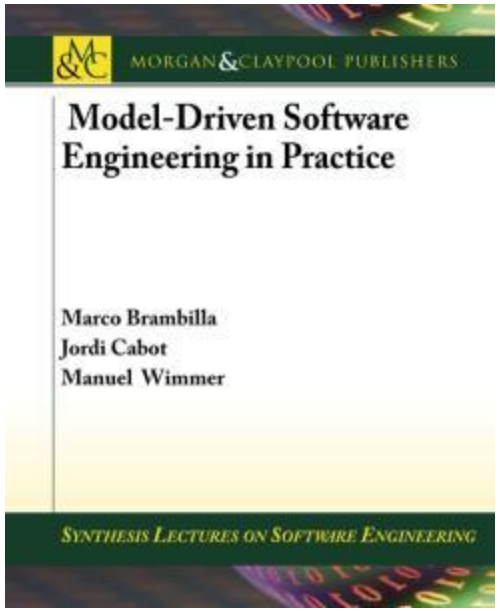
Profa. Rosana T. Vaccare Braga

MORGAN & CLAYPOOL PUBLISHERS

PARTE DO MATERIAL BASEADO NESTE LIVRO:

MODEL-DRIVEN SOFTWARE ENGINEERING IN PRACTICE

Teaching material for the book
Model-Driven Software Engineering in Practice
by Marco Brambilla, Jordi Cabot, Manuel Wimmer.
Morgan & Claypool, USA, 2012.



Definições importantes

Modelo

Um modelo pode ser definido como uma **descrição** ou **especificação** de um **sistema** e seu ambiente. Um modelo, geralmente é representado por uma combinação de **gráficos** e **explicações textuais** (OMG MDA Guide 2003).

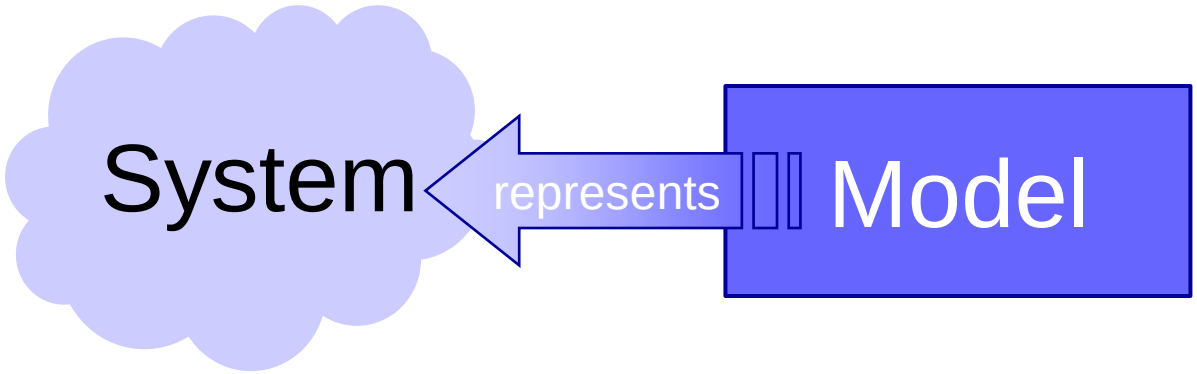
Definições importantes

Model Driven

Uma abordagem **é Model-Driven porque utiliza modelos para direcionar diretamente o entendimento, projeto, construção, *deployment*, operação, manutenção e modificação do sistema** (OMG MDA Guide 2003).

Modelos

O que é um modelo?



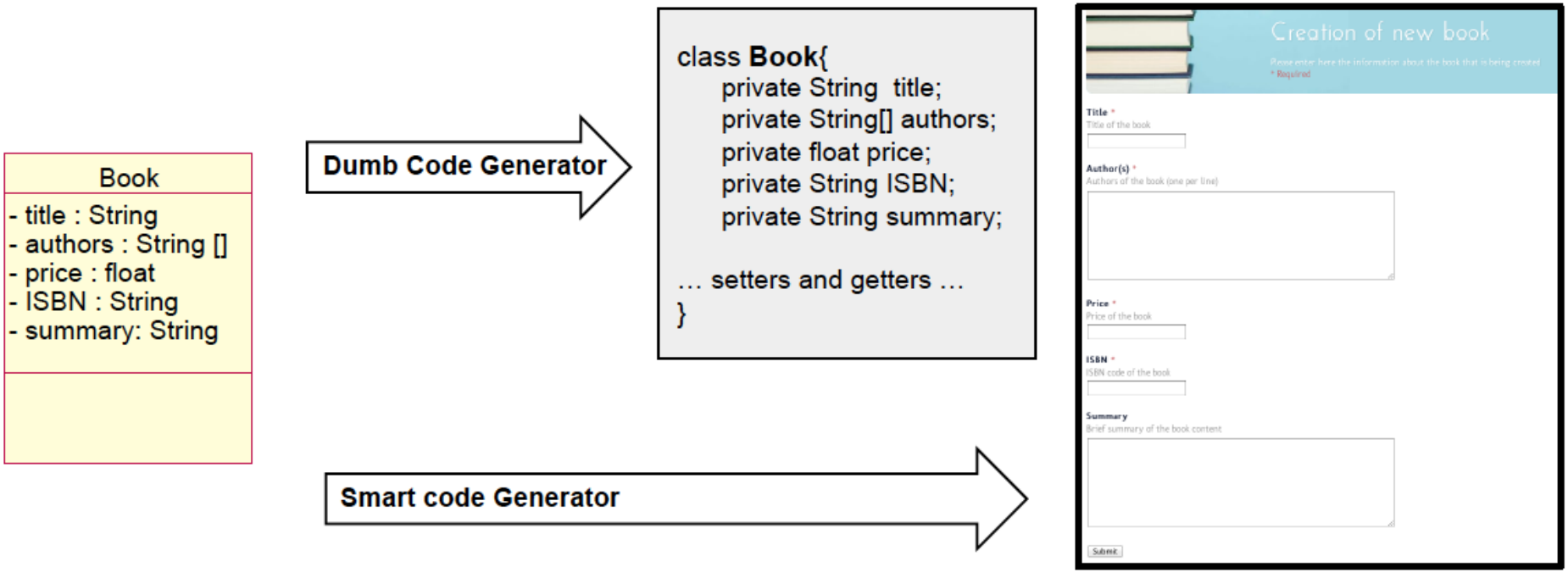
Mapping Feature	A model is based on an original (=system)
Reduction Feature	A model only reflects a (relevant) selection of the original‘s properties
Pragmatic Feature	A model needs to be usable in place of an original with respect to some purpose

MDSE – visão geral

- MDSE considera modelos como cidadãos de primeira classe na engenharia de sofware
- A forma com que modelos são definidos e gerenciados é baseada nas necessidades reais que eles vão endereçar
- MDSE define abordagens sólida para a definição de
  - modelos
  - transformações
  - Processo de desenvolvimento.

Motores de execução espertos vs burros

- Operações CRUD (Create/Read/Update/Delete) em geral contam como 80% da funcionalidade geral do software
- é um esforço grande gasto pra fazer coisas que seguem regras simples



Definições importantes (cont 3)

Transformações de modelos

Transformação de modelos é o processo de se converter um modelo em outro modelo do mesmo sistema (OMG MDA Guide 2003).

Conceitos

Princípios e objetivos

- **Abstração a partir de tecnologias de realização específicas**
  - Sabendo as regras de transformação, poderia ser gerado código em diferentes linguagens específicas
  - Aumenta a **portabilidade** do software para tecnologias novas ou em mudança: modele uma única vez e construa sempre e em qualquer lugar
  - **Interoperabilidade** entre diferentes tecnologias pode ser automatizadas (Technology Bridges)

- **Geração de código automática** a partir de modelos abstratos
  - e.g., geração de Java-APIs, XML Schemas, etc. a partir de UML
  - Requer modelos expressivos e precisos
  - Aumento de **produtividade** e **eficiência** (modelos estão sempre atualizados)

- **Desenvolvimento separado** da aplicação e infraestrutura
  - Separação do código da aplicação do código da infraestrutura (e.g., Application Framework) aumenta **reusabilidade**
  - Ciclos de desenvolvimento flexíveis

MDSE – ingredientes principais

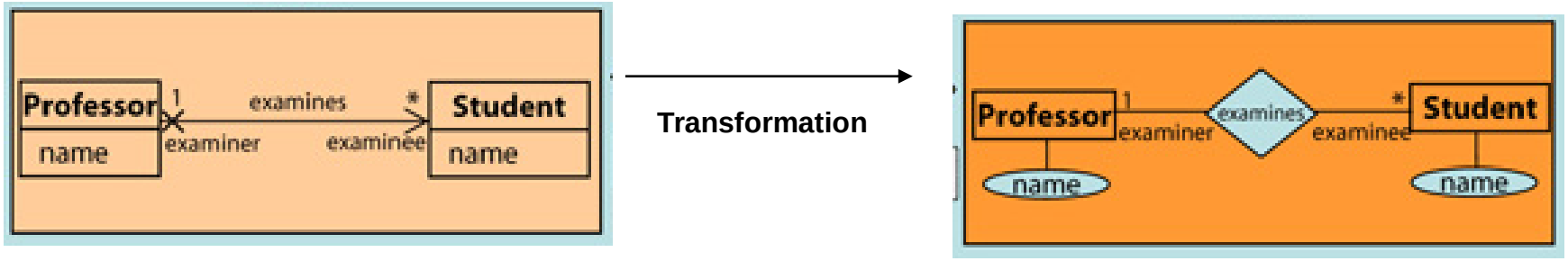
- **Conceitos:** componentes que levam à metodologia
- **Notações:** A forma com que os conceitos são representados
- **Processos e regras:** atividades que levam a produção do produto final
- **Ferramentas:** Aplicações que facilitam as atividades ou sua coordenação

MDSE - Equação

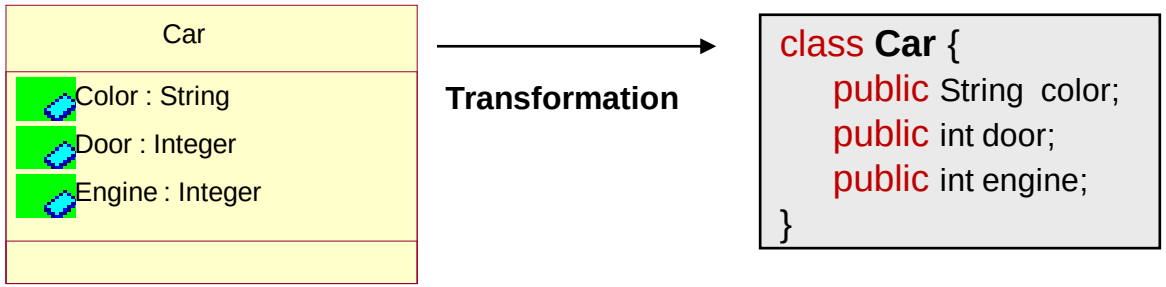
Modelos + Transformações = Software

Tipos de transformações

Model to Model →M2M



Model to Text →M2T



Tipos de transformações

Text to Text →T2T

```
class Car {
    public String  color;
    public int door;
    public int engine;
}
```

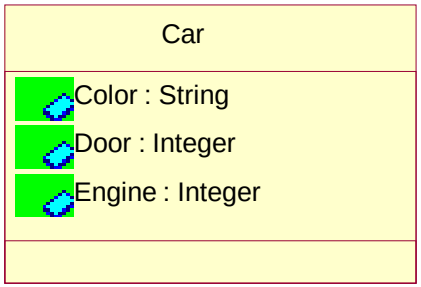
Transformation

```
class Car {
    public String  color;
    public int door;
    public int engine;
}
public String getColor () {
return this.color;
}
Public void setColor (String color) {
this.color = color;
}
…
```

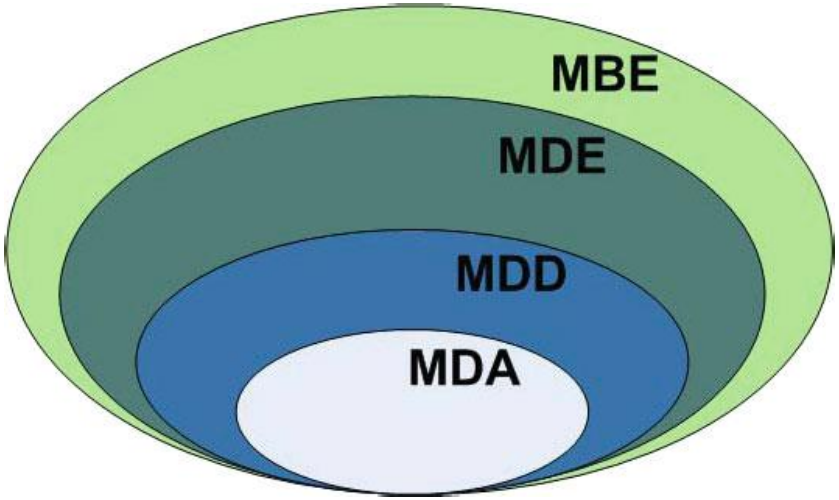
Text to Model → T2M

```
class Car {
    public String  color;
    public int door;
    public int engine;
}
```

Transformation



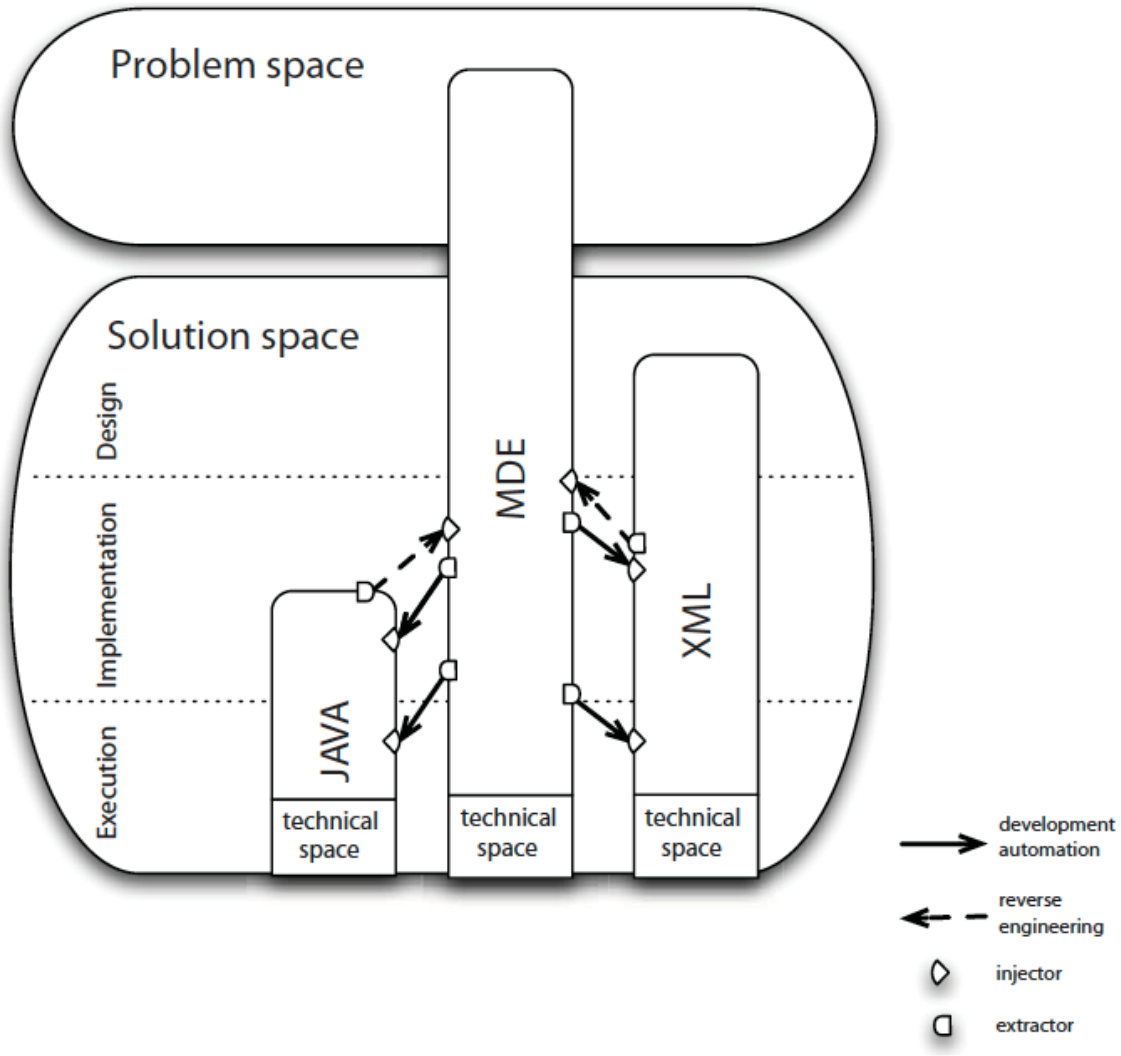
The MD* Jungle of Acronyms



- **Model-Driven Development (MDD)** is a development paradigm that uses models as the primary artifact of the development process.
- **Model-driven Architecture (MDA)** is the particular vision of MDD proposed by the Object Management Group (OMG)
- **Model-Driven Engineering (MDE)** is a superset of MDD because it goes beyond of the pure development
- **Model-Based Engineering** (or “model-based development”) (**MBE**) is a softer version of MDE, where models do not “drive” the process.

Target of MDSE

- The **Problem Domain** is defined as the field or area of expertise that needs to be examined to solve a problem.
- The **Domain Model** is the conceptual model of the problem domain
- **Technical Spaces** represent specific working contexts for the specification, implementation, and deployment of applications.



Modeling Languages

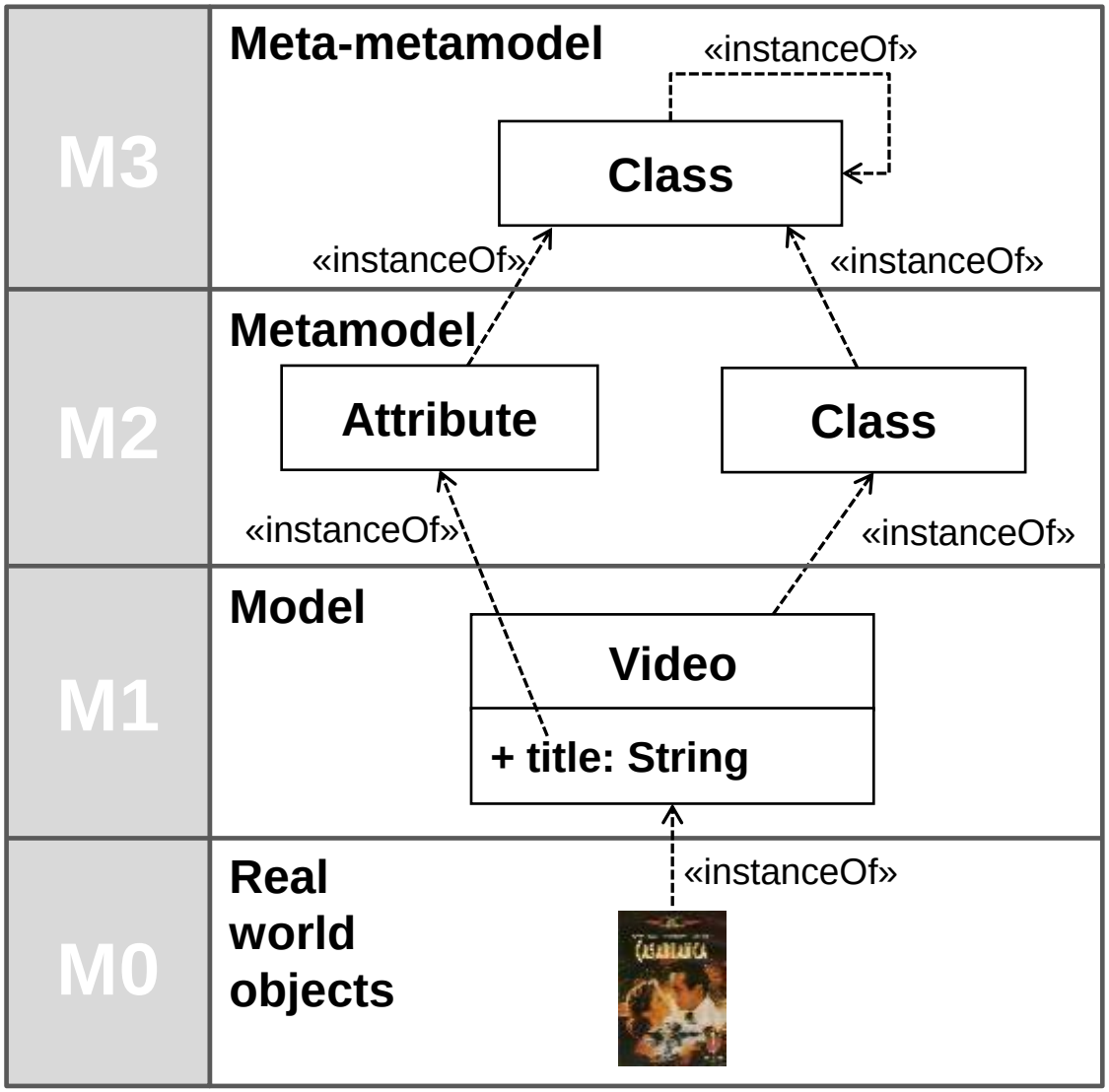
- **Domain-Specific Modeling Languages (DSMLs, DSLs):** languages that are designed specifically for a certain domain or context
- DSLs have been largely used in computer science.
- Examples: HTML, Logo, VHDL, Mathematica, SQL

- **General Purpose Modeling Languages** (GPMLs, GMLs, or GPLs): languages that can be applied to any sector or domain for (software) modeling purposes
- The typical examples are: UML, Petri-nets, or state machines

Metamodeling

- To represent the models themselves as “instances” of some more abstract models.
- **Metamodel** = yet another abstraction, highlighting properties of the model itself

- Metamodels can be used for:
  - defining new languages
  - defining new properties or features of existing information (metadata)

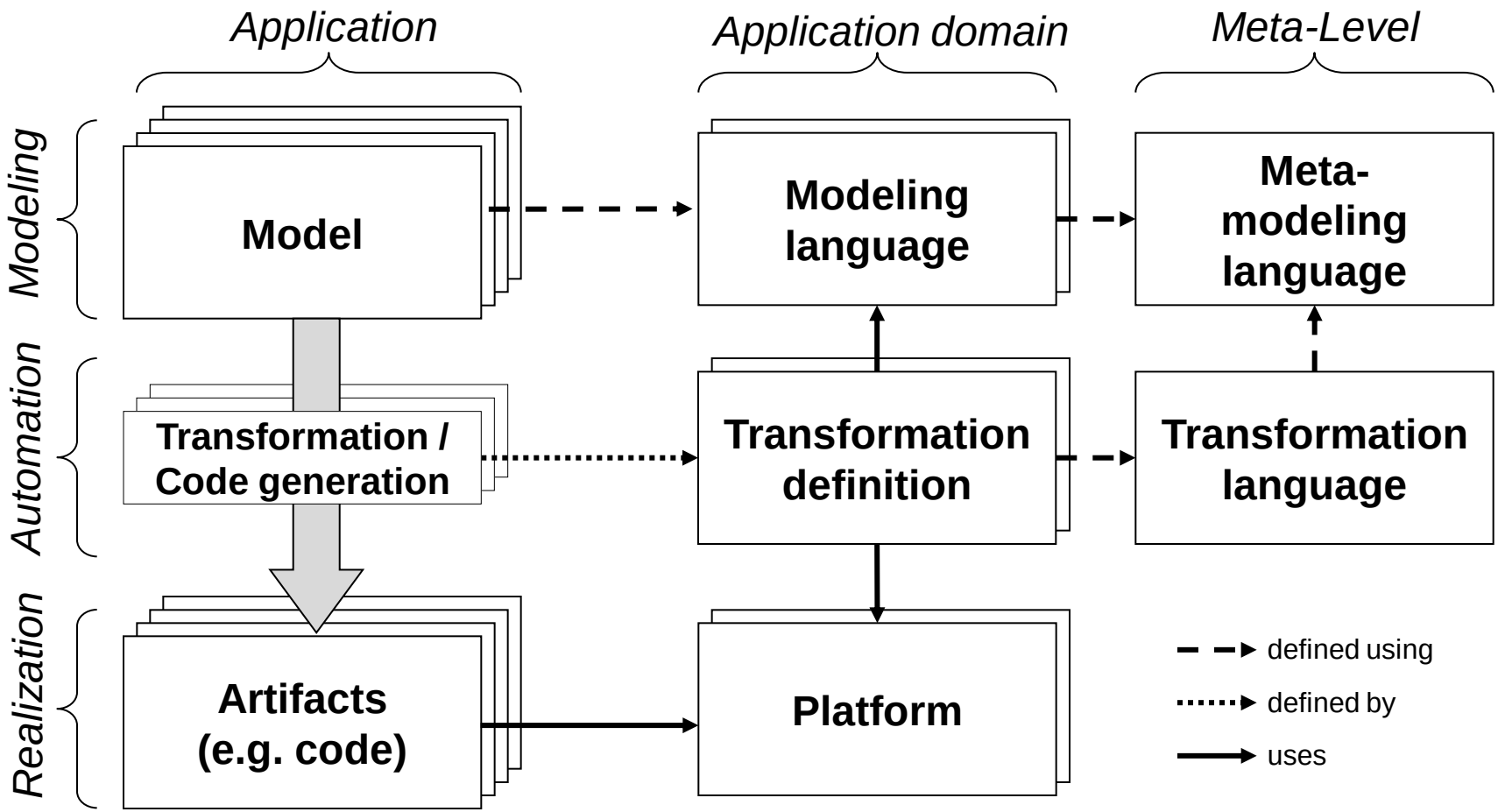


Model Transformations

- Transforming items
- MDSE provides appropriate languages for defining model transformation rules
- Rules can be written manually from scratch by a developer, or can be defined as a refined specification of an existing one.
- Alternatively, transformations themselves can be produced automatically out of some higher level mapping rules between models
  - defining a mapping between elements of a model to elements to another one (**model mapping or model weaving**)
  - automating the generation of the actual transformation rules through a system that receives as input the two model definitions and the mapping
- Transformations themselves can be seen as models!

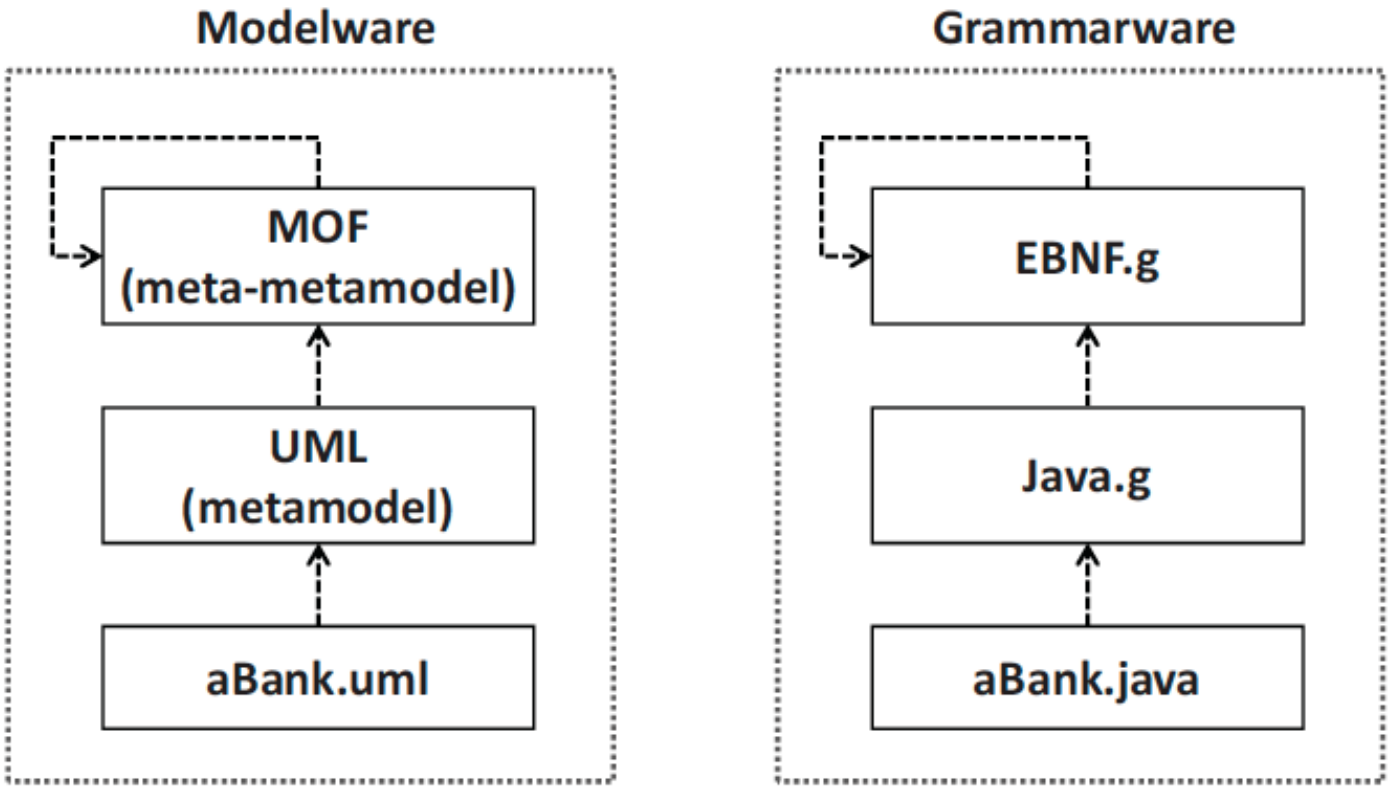
Concepts

Model Engineering basic architecture



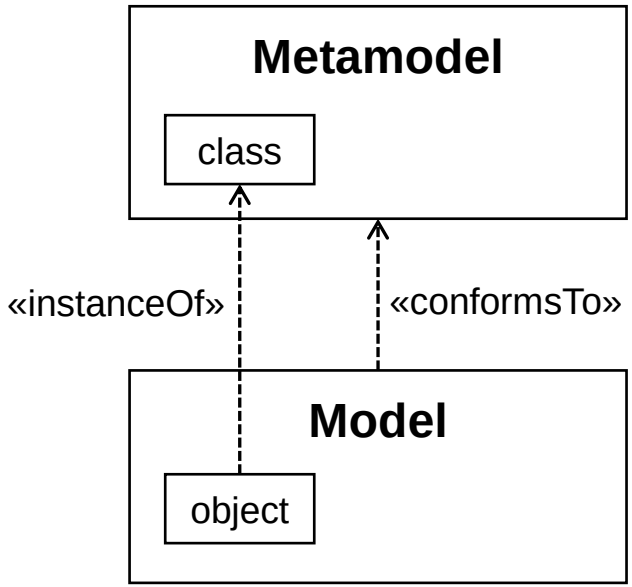
Modelware vs. Grammarware

- Two technical spaces



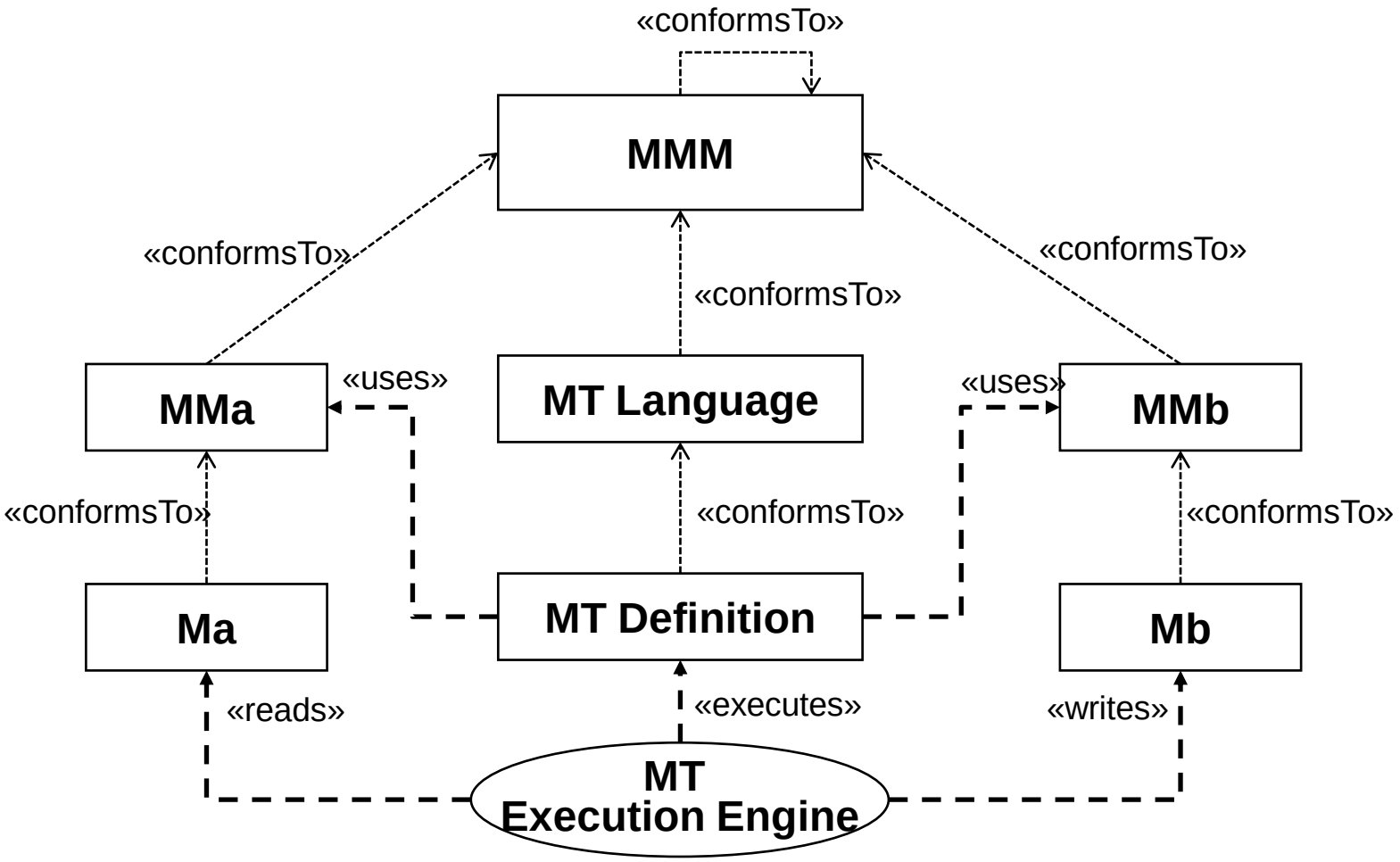
InstanceOf vs. ConformsTo

- Conformance is between models
- Instantiation is between model elements



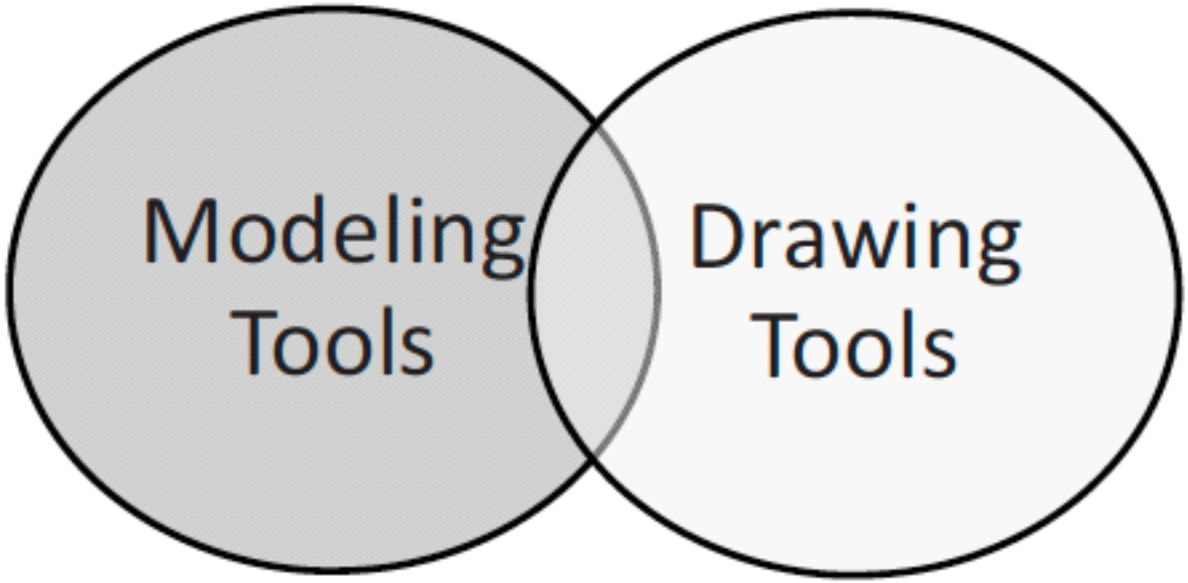
Model Transformations

MOF and transformation setting



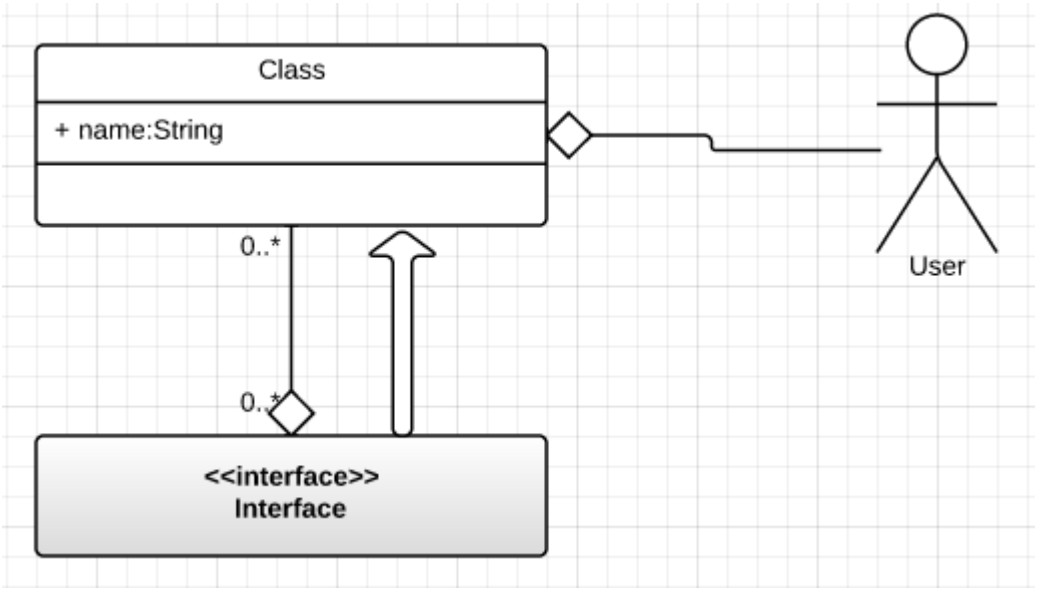
Tool support

- Drawing vs. modeling



Tool support

- Drawing vs. modeling



getAllRectangularShapes vs. getAllClasses

Model-based vs. Programming-based MDSE Tools

- Model-based: developed using MDSE principles → one should apply the principles ones advocates

- Programming-based: developed using traditional coding techniques

Eclipse and EMF

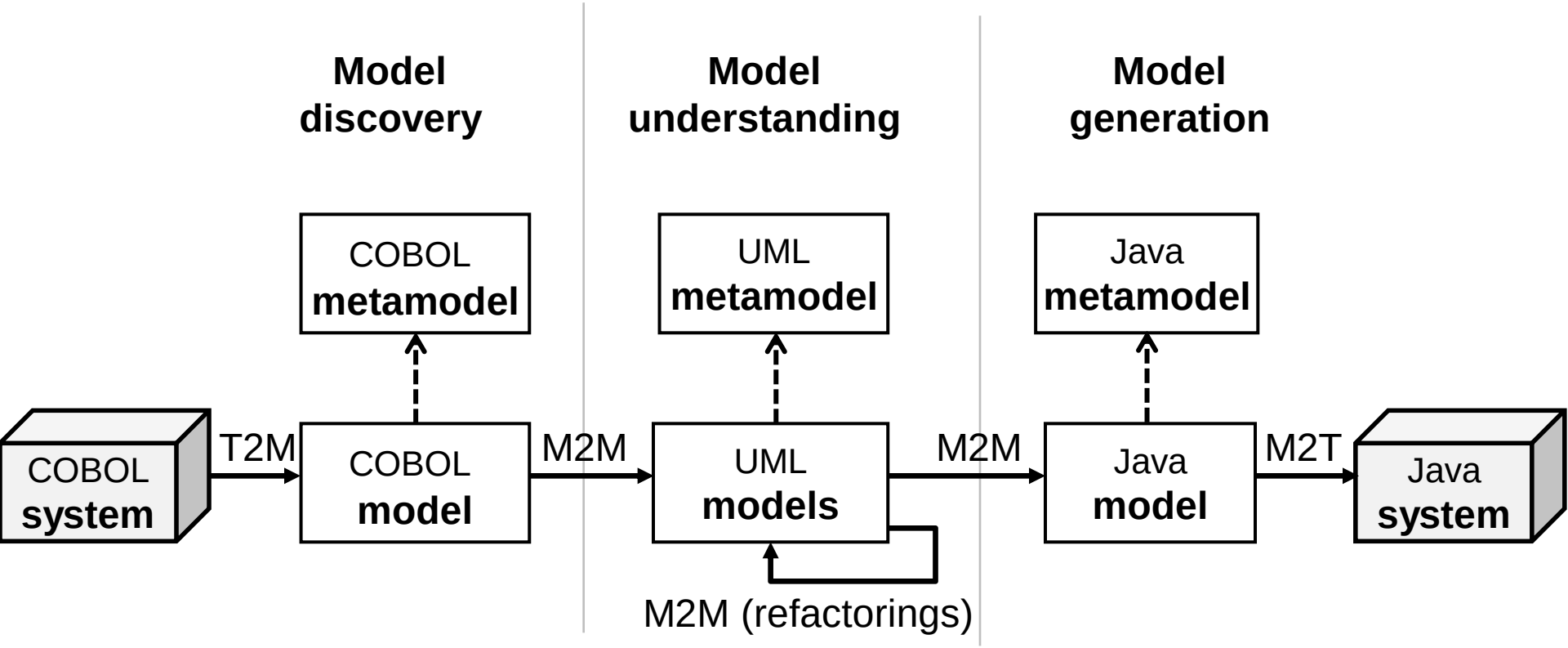
- Eclipse Modeling Framework
- Full support for metamodeling and language design
- Fully MD (vs. programming-based tools)

Code Generation

- Goal: generating running code from higher level models
  - Like compilers producing executable binary files from source code
  - Also known as model compilers

- Once the source code is generated state-of-the-art IDEs can be used to manipulate the code

Exemplo de uso de MDD: reengenharia de sistema Cobol para Java



PRINCÍPIOS DE MDA

Model-Driven Architecture (MDA)

- OMG - É uma associação internacional incorporada a uma corporação sem fins lucrativos que está associada a diversas organizações espalhadas pelo mundo.

Produtos (especificações) de sucesso:

- CORBA, UML, XMI, entre outros...

Definições importantes

Arquitetura

A arquitetura de um sistema é a especificação das partes, dos conectores do sistema e das regras de interação das partes que usam estes conectores (OMG MDA Guide 2003).

O que é MDA?

- Engenharia de software baseada em modelos
- Desenvolvimento para múltiplas plataformas
- Geração da aplicação

Problemas das abordagens para geração de código do passado

- Sistemas de baixa qualidade e baixa manutenibilidade
- Pouco código gerado em comparação com o tamanho total da aplicação

MDA – Conceito fundamentais

O MDA define uma abordagem que separa a especificação da funcionalidade do sistema, da especificação da implementação do sistema em uma determinada plataforma de desenvolvimento.

Benefício do MDA

Ao contrário das técnicas de geração de código do passado, o grande benefício do MDA não está na geração do código, **mas sim na possibilidade de primeiro se pensar no sistema em um alto nível de abstração**.

Especificação de exemplo

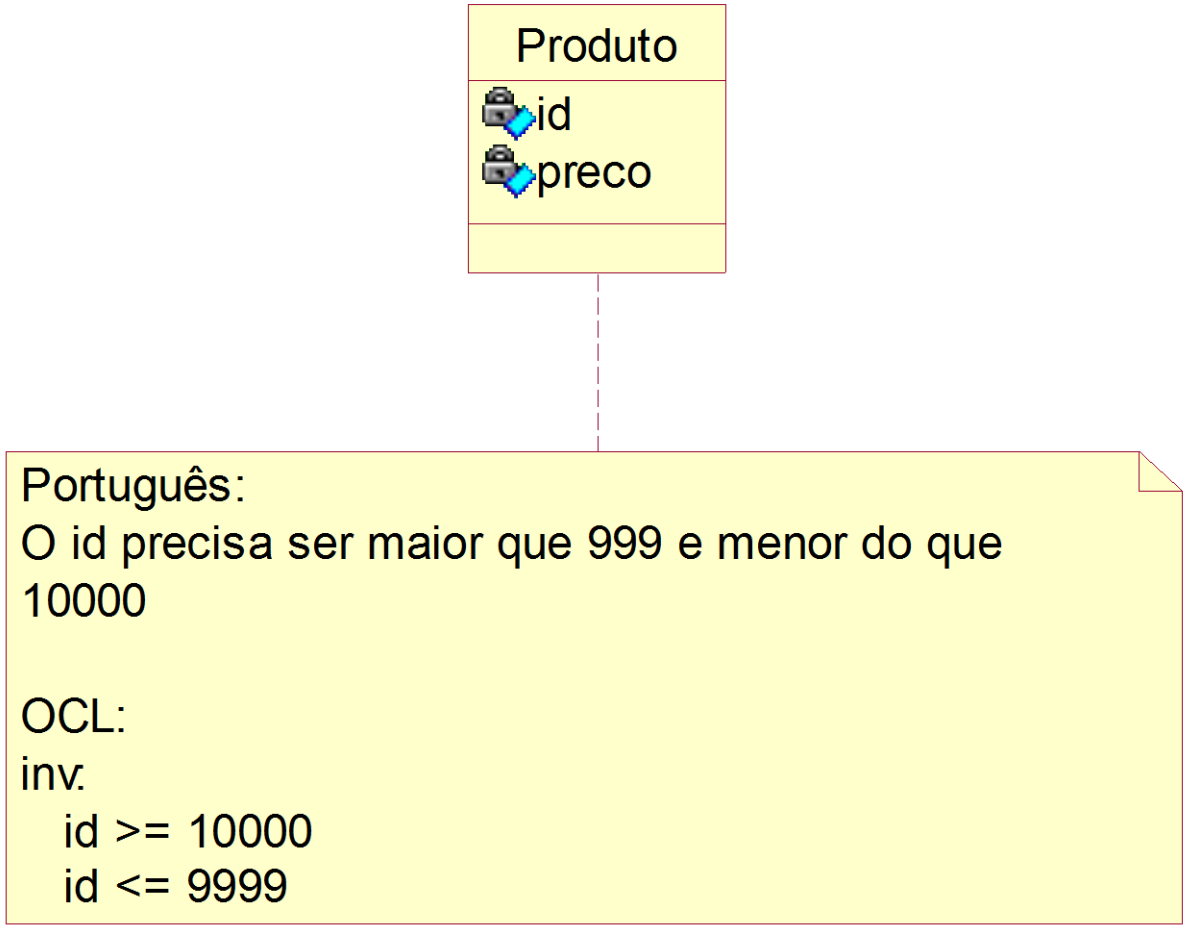
Considere uma descrição de um produto. O produto possui atributos como preço e um identificador. Um identificador de produto deve ser maior do que 1000 e menor do que 9999.

MDA – Primeiro passo

Construção do PIM

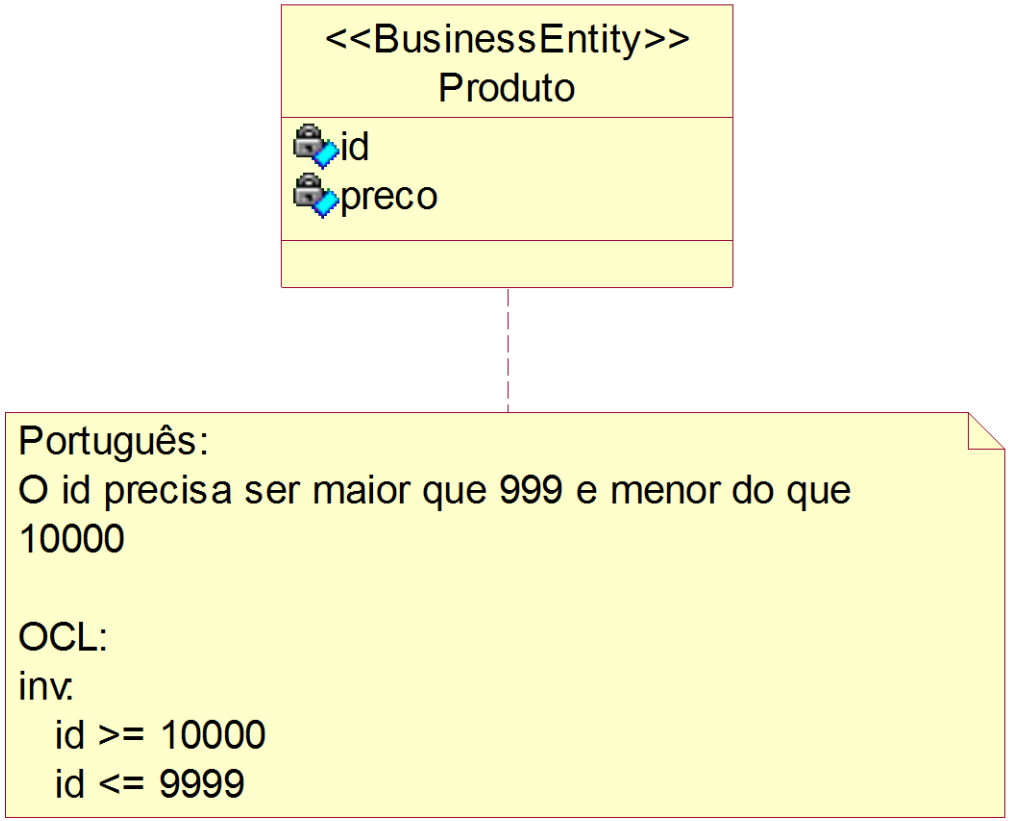
- PIM ou *Platform Independent Model* é um modelo independente da plataforma escolhida.

Exemplo PIM - UML



MDA – Segundo passo

Marcações

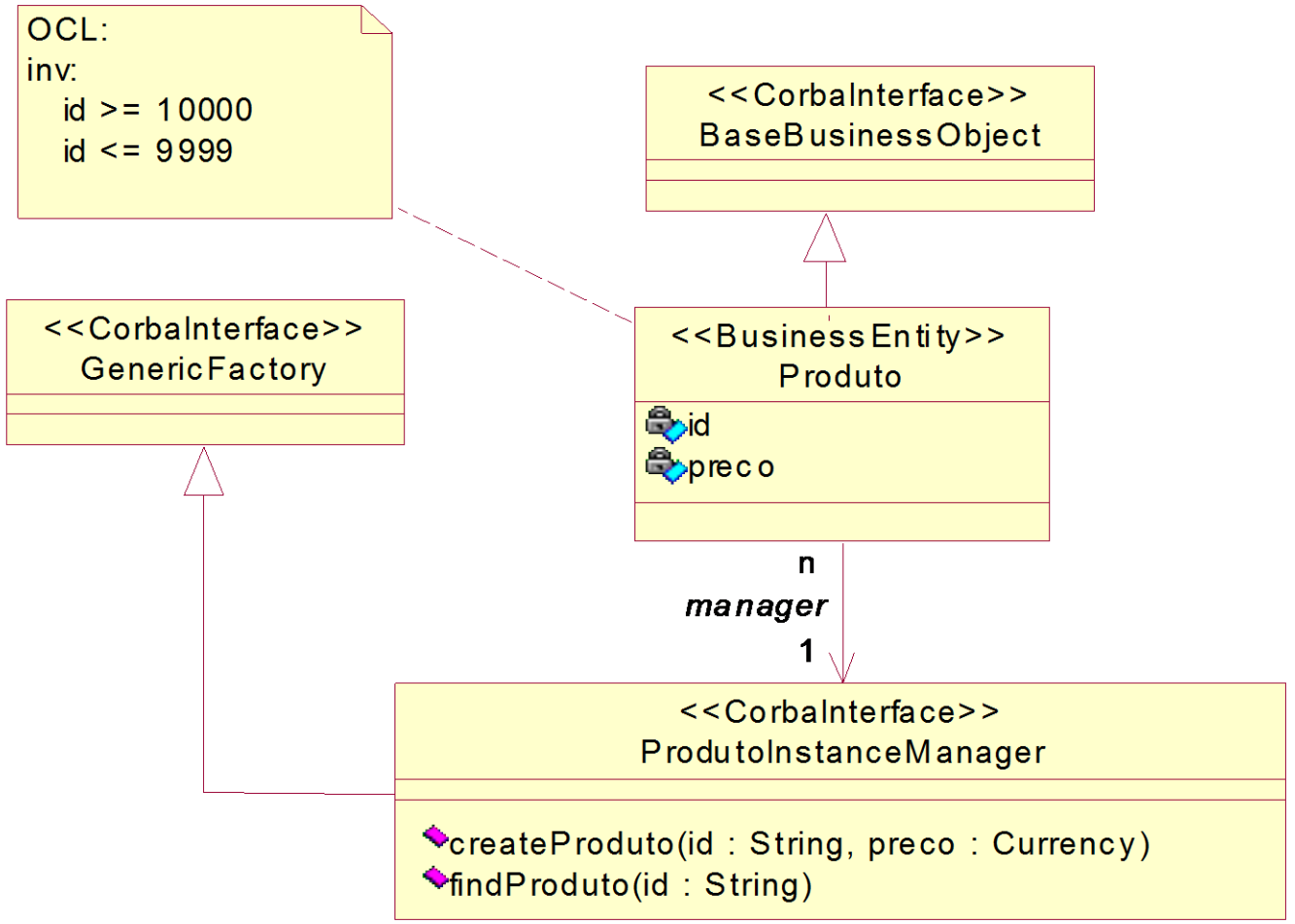


MDA – Terceiro passo

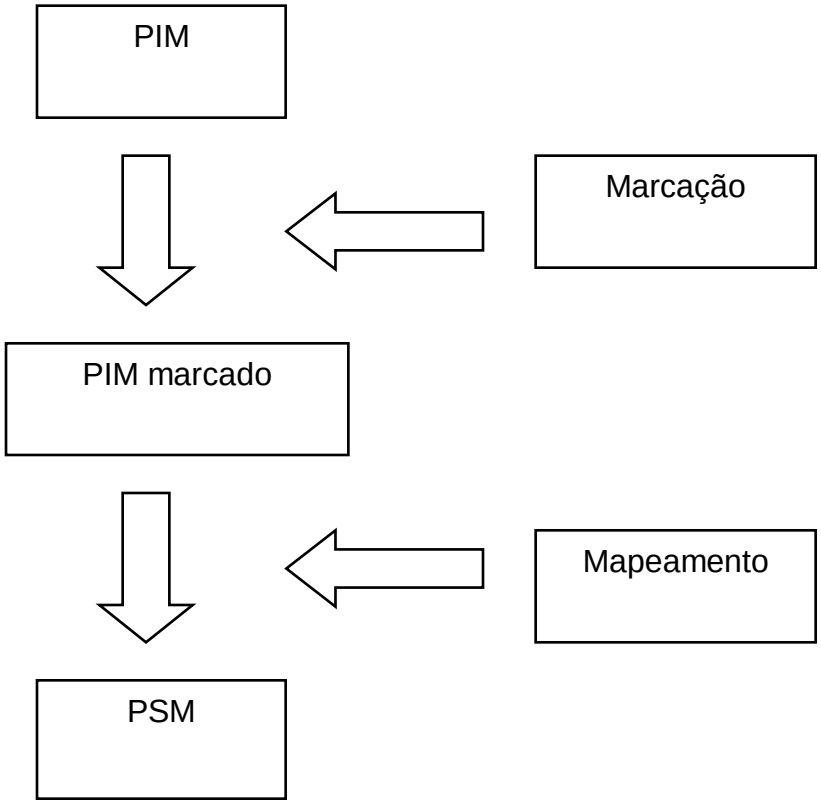
Transformação do PIM em PSM

- O modelo PSM representa o sistema em termos de sua implementação e é específico para uma plataforma de desenvolvimento

Exemplo PSM - UML



Generalizando o processo



Transformação final em código