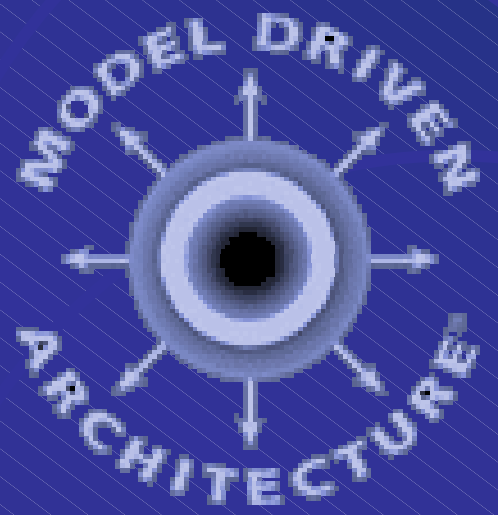


Model Driven Architecture

Centro de Informática/UFPE

Fernando Trinta

Roteiro

- **Contexto**
- **Introdução**
- **Conceitos MDA**
  - Platform Independent Model
  - Platform Specific Model
  - Transformations
- **Consequências**
- **Promessas**
- **Conclusões**

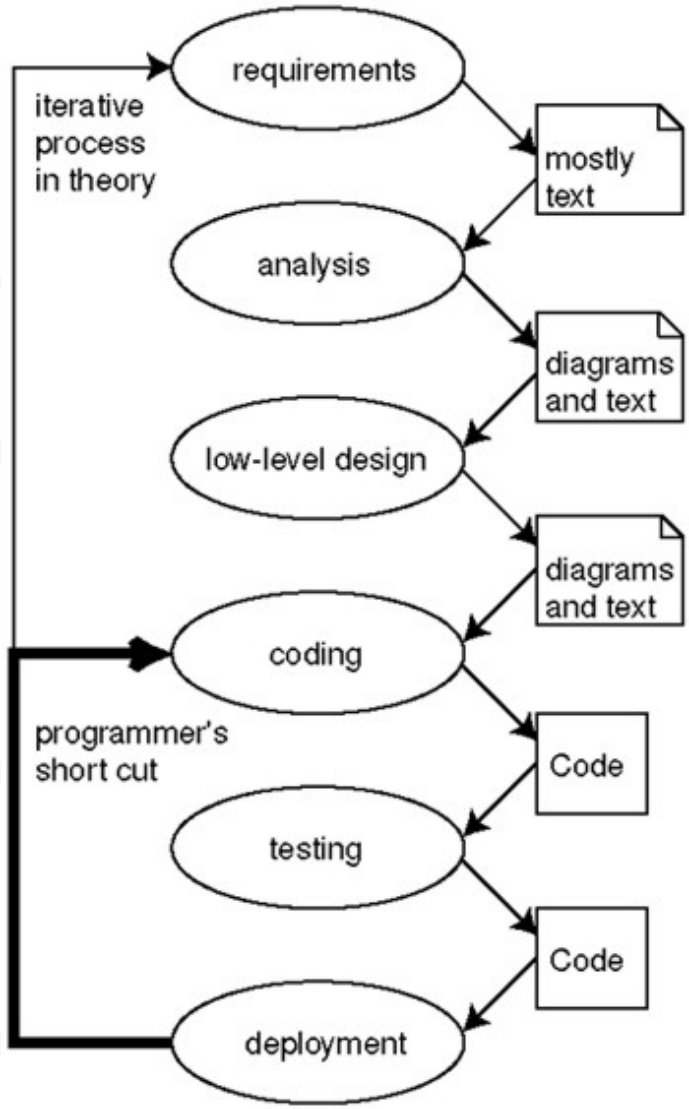
Engenharia de Software Hoje

- **Problema da produtividade**
  - Sistemas cada vez mais complexos
  - Grande número de tecnologias envolvidas
  - Equipes médias/grandes
  - Ênfase na codificação
- **Problema de Portabilidade**
  - Portar sistemas entre diferentes tecnologias é uma tarefa árdua
    - Padrões de Projeto, Componentes, Middleware ajudam, mas não são suficientes para resolver o problema!!!

Engenharia de Software Hoje

- **Problema de Interoperabilidade**
  - Sistemas legados, fusões entre empresas implicam na necessidade de interoperabilidade entre diferentes sistemas
- **Problema da Documentação**
  - Modelos utilizados apenas na fase inicial de concepção dos sistemas
    - Unified Modeling Language – Padrão *de facto* para modelagens de sistemas
  - Durante implementação, ênfase no código produzido
    - Possíveis mudanças de requisitos são realizadas diretamente sobre código

Engenharia de Software Tradicional



Model Driven Architecture

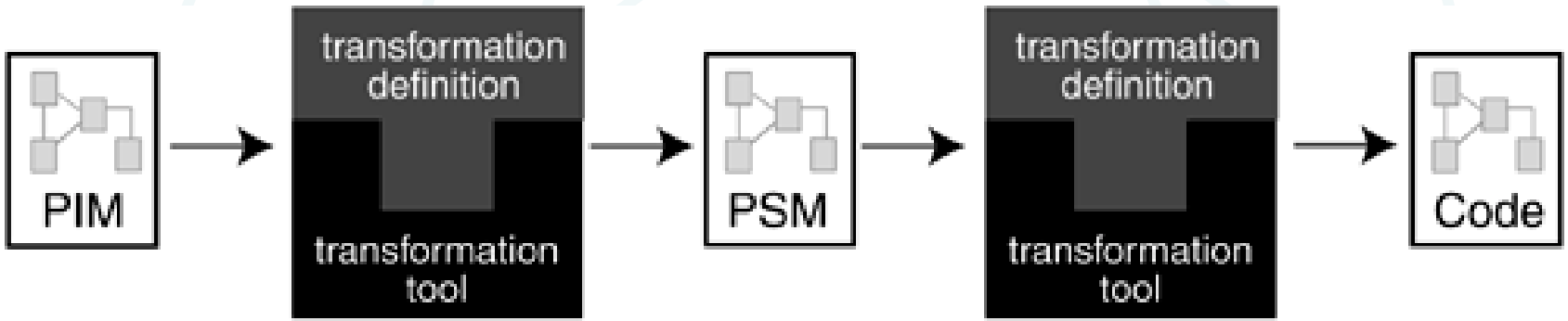
- **Proposta do Object Managment Group:**
  - Objetivo: Portabilidade e Interoperabilidade de aplicações através do uso de modelos que utilizem linguagens formais
  - "Design once, build it on any platform"

- **Segundo MDA Guide:**

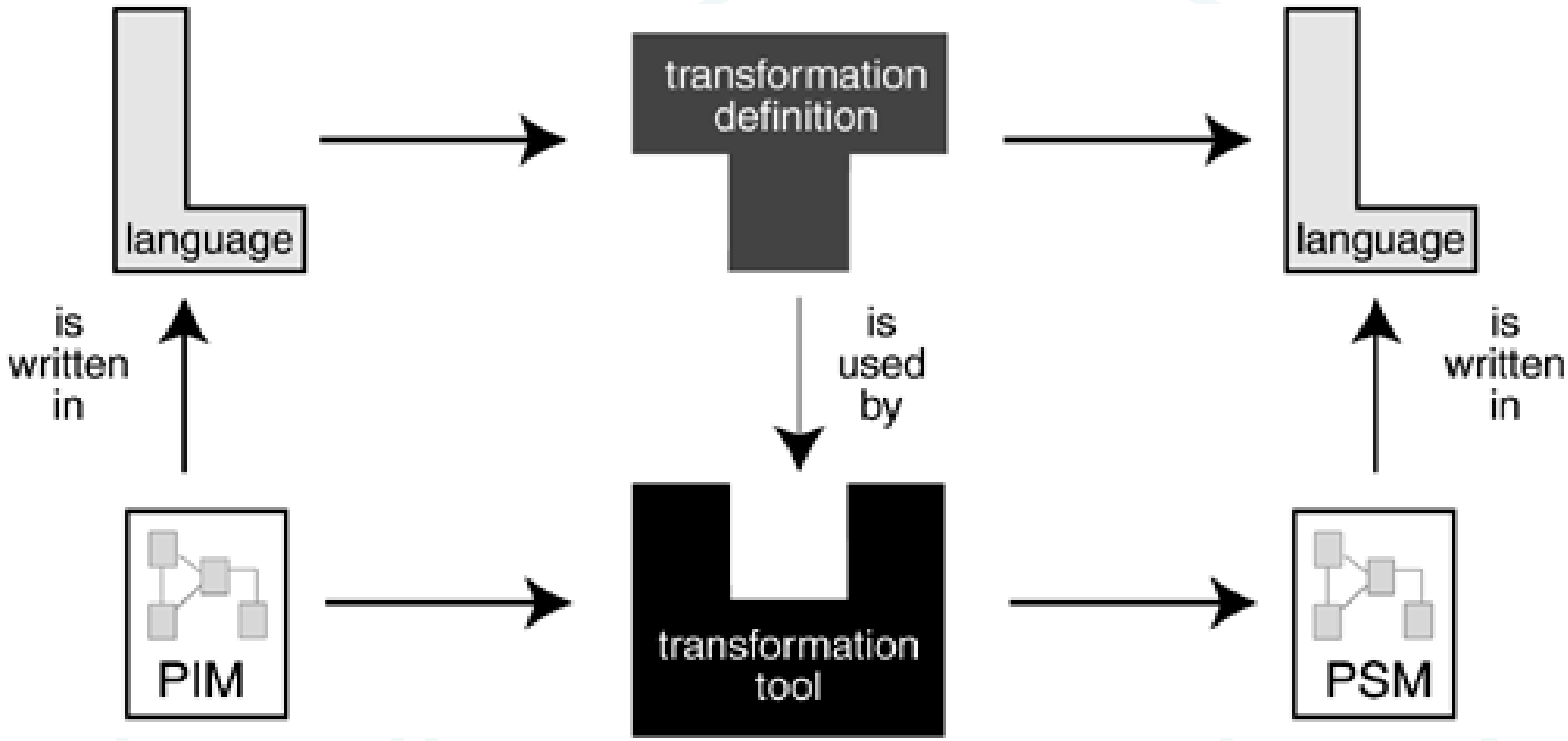
  "Abordagem para especificação de sistemas de TI que busca separar a especificação da funcionalidade do sistema, da especificação da implementação desta funcionalidade sobre alguma plataforma específica"

MDA Pattern

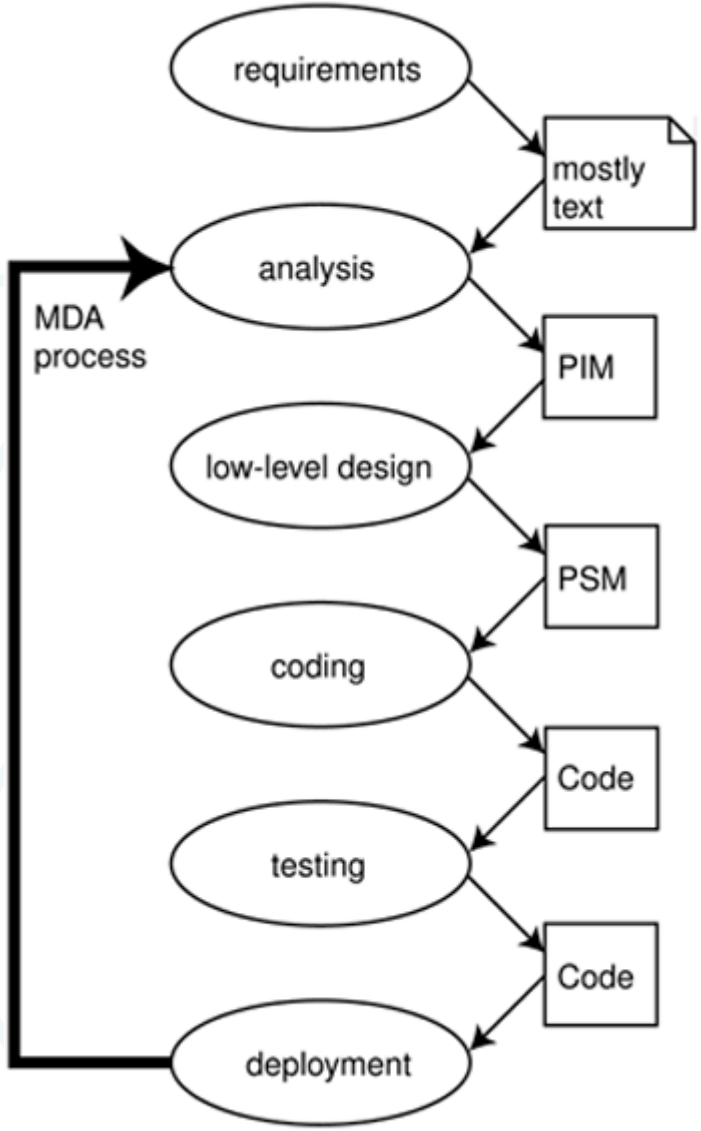
- **Um modelo independente de plataforma (PIM) é criado, e partindo de suas definições são gerados modelos para plataformas específicas(PSM), através da definição de transformação entre estes modelos.**
- **O PSM por sua vez é transformado em código.**



Outra visão da MDA Pattern

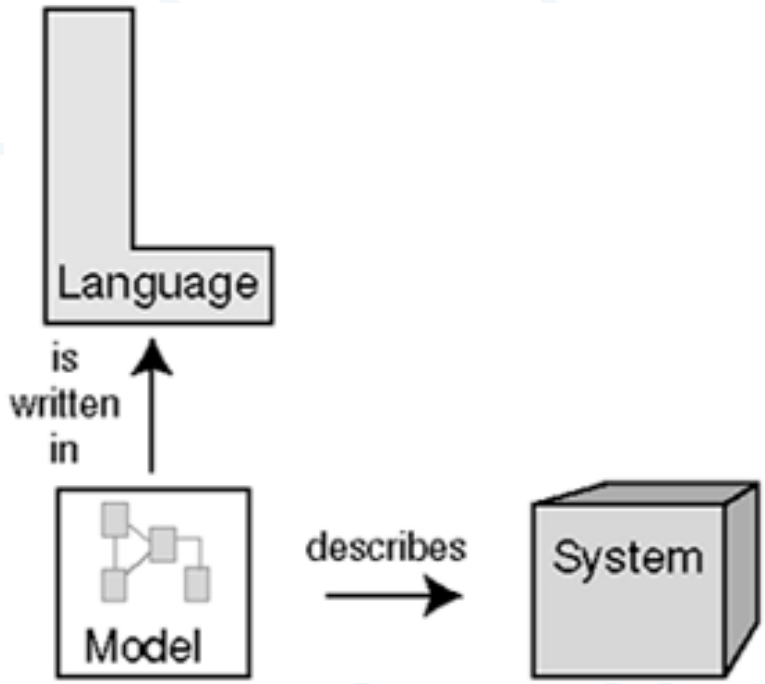


Engenharia de Software - MDA



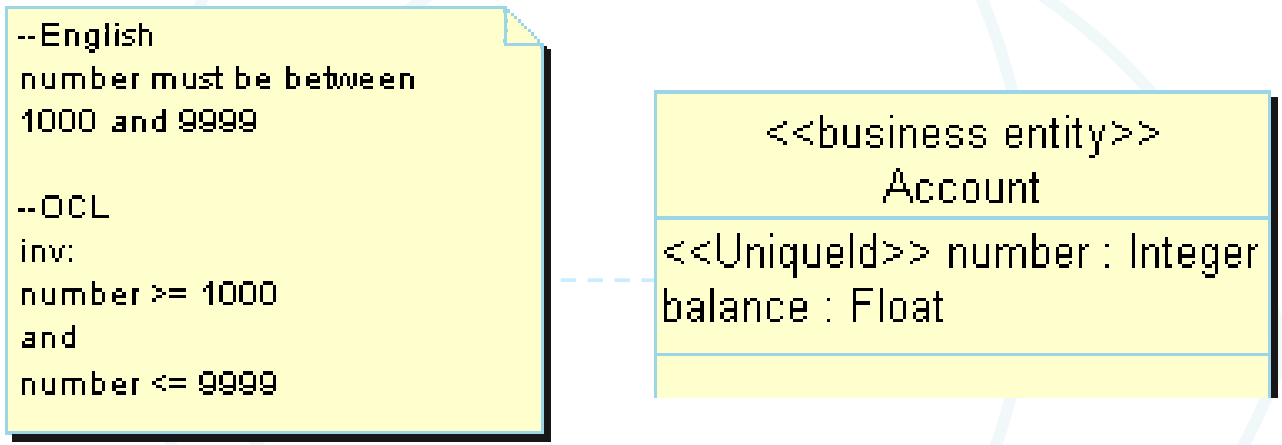
Model Driven Architecture

- **Conceitos chaves:**
  - Um modelo é a especificação formal da função, estrutura e/ou comportamento sistema
    - Exemplos:
      - Codigo Fonte
      - Uma especificação UML



Model Driven Architecture

- ***PIM - Platform Independent Model*:**
  - Modelos que representam a funcionalidade do sistema, sem representações inerentes a qual plataforma onde o sistema será desenvolvido. Possivelmente UML como linguagem

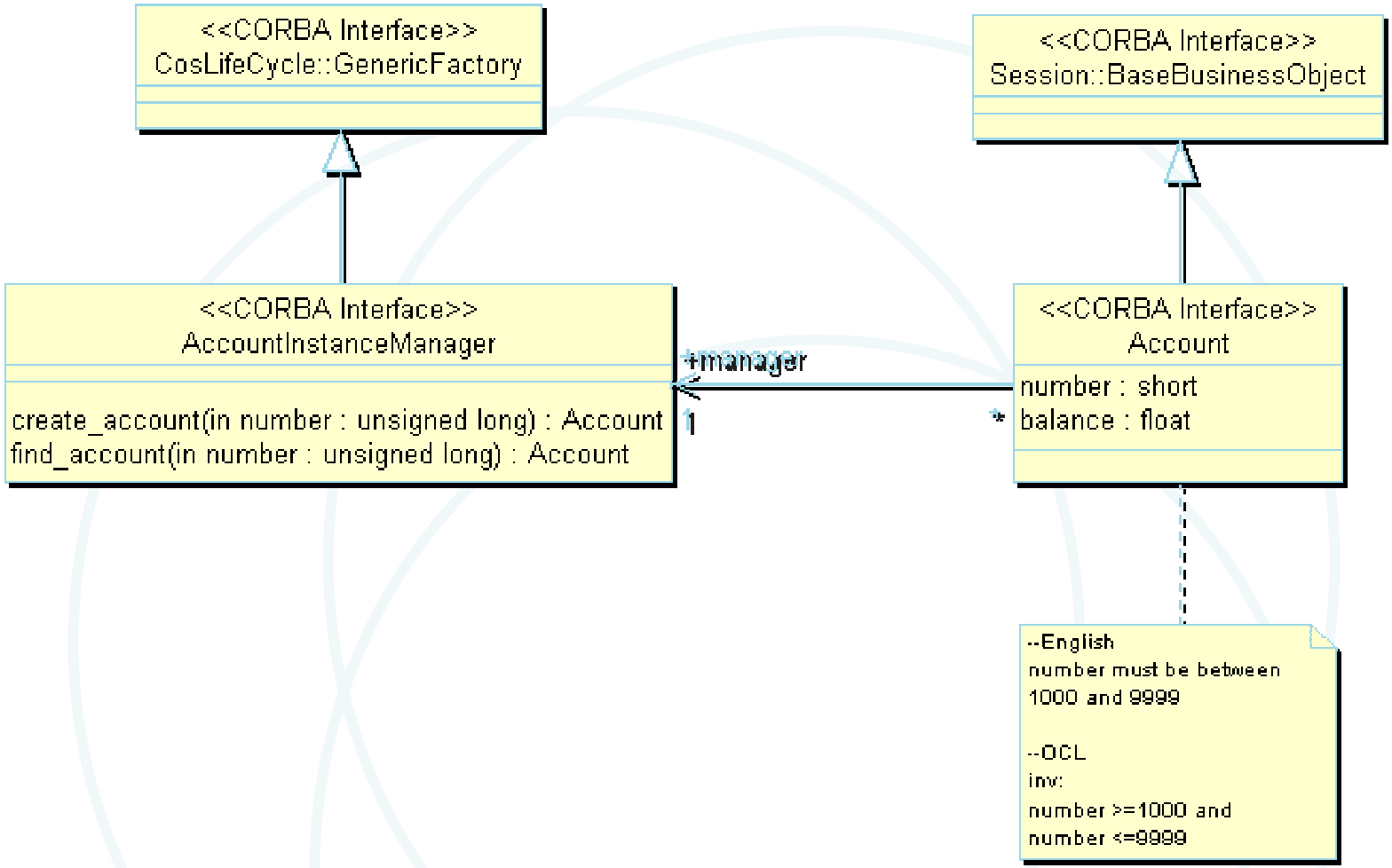


Model Driven Architecture

- ***PSM - Platform Specific Model*:**
  - Representam um PIM, voltado para alguma plataforma específica
    - CORBA, Enteprise JavaBeans, Microsoft .NET
  - Uso de UML Profiles
    - Profiles – Extensões UML para representar uma plataforma específica
      - CCM Profile
      - EJB Profile

Model Driven Architecture

- **Exemplo PSM**



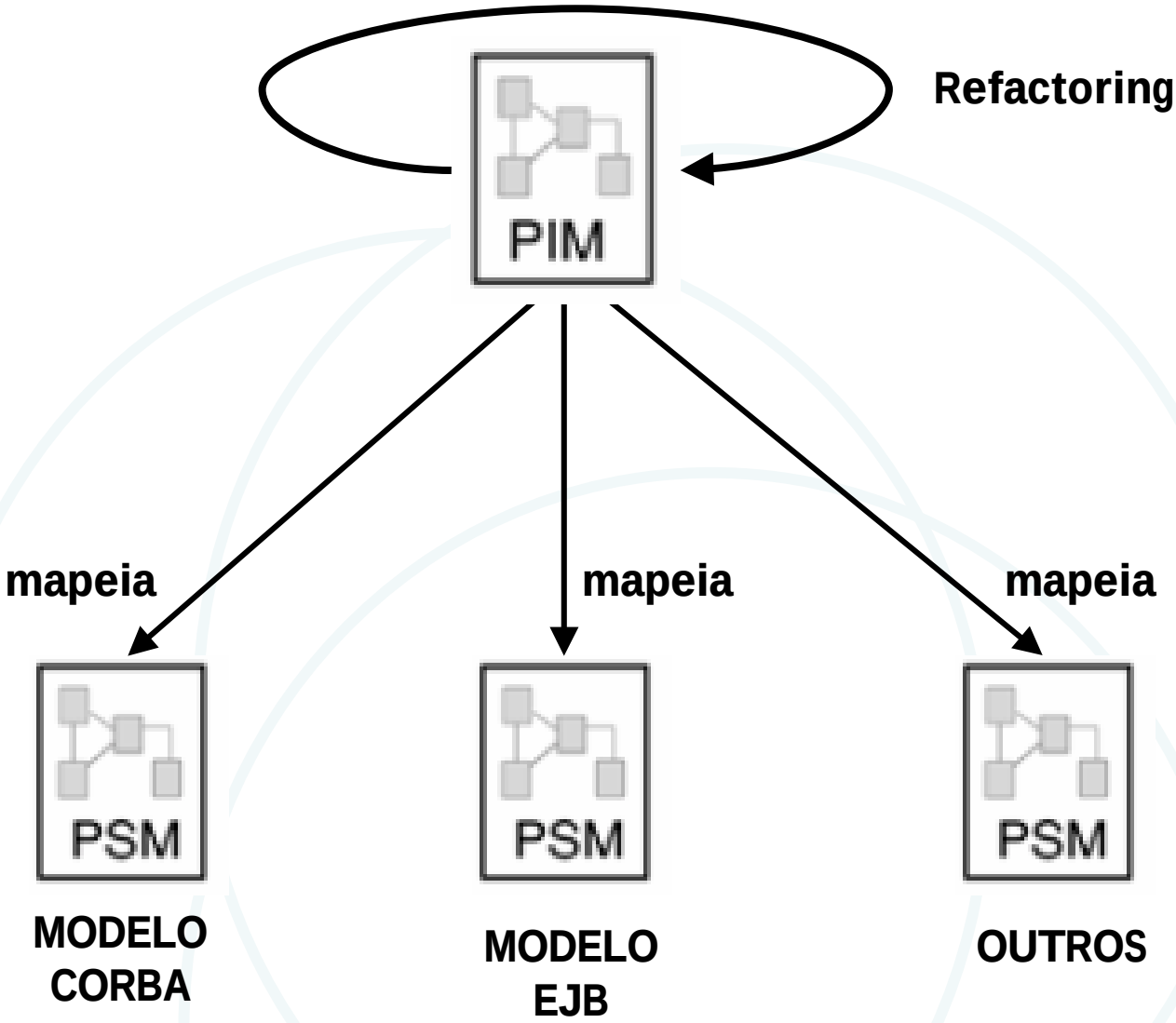
Model Driven Architecture

- *Código* – Java, C#, SQL

- *Transformações*: geração automática de um modelo a partir de outro modelo, baseado em uma definição de transformações

  - Aplicações: Refactoring, Normalização

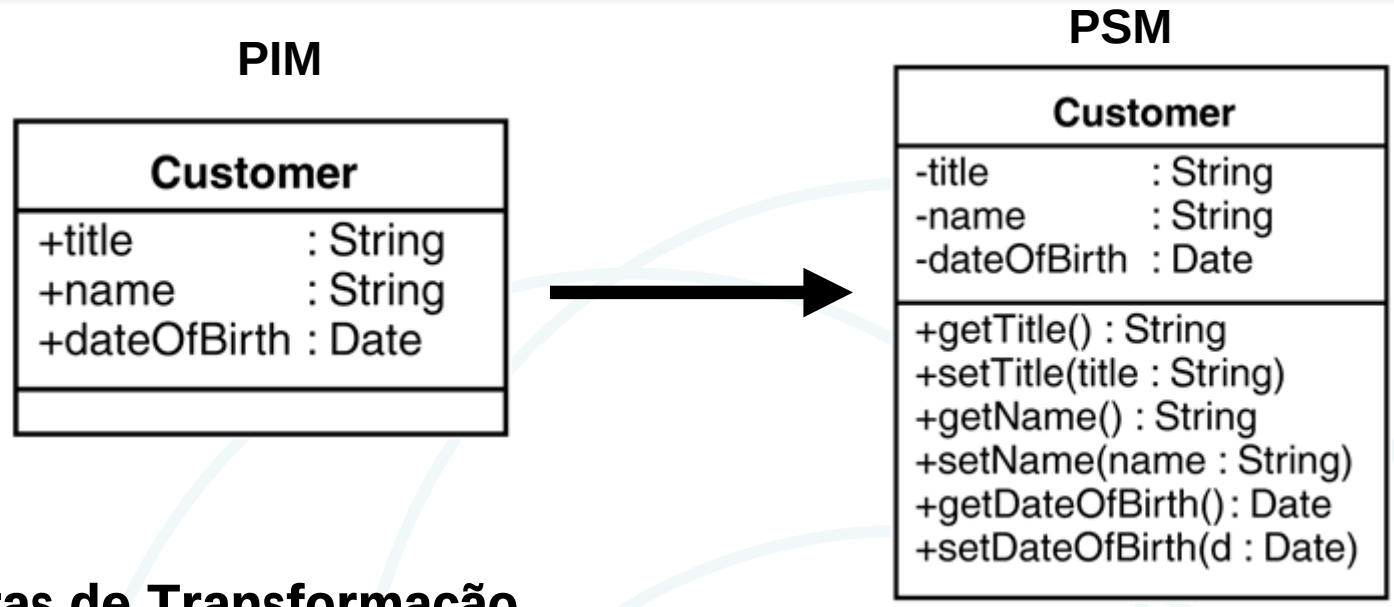
Model Driven Architecture



Transformações

- *Definição de transformação* : conjunto de regras que descrevem como um modelo original é transformado em outro modelo
- *Regra de transformação* : descrição de como um ou mais construtores na linguagem do modelo original deve(m) ser transformado(s) em um ou mais construtores na linguagem do modelo destino
- *Linguagem de definição de transformação* : A linguagem na qual as definições das transformações são descritas.

Exemplo Transformação

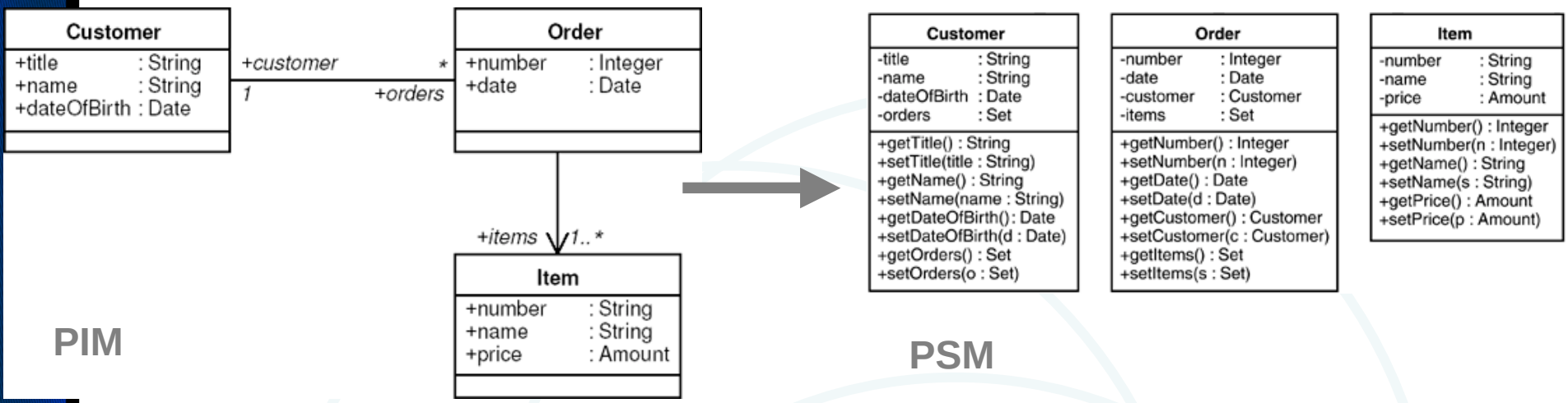


Regras de Transformação

(1) Para cada atributo público *attributeName : Type* da classe *className* no PIM, devem ser gerados atributos e operações no modelo destino seguindo a regras abaixo::

- **Um atributo privado com o mesmo do nome/tipo original: *attributeName : Type***
- **Uma operação pública cujo nome deve ser precedido pela palavra "get", seguido pelo nome do atributo com a primeira letra maiscula. Esta operação deve ter tipo de retorno igual ao tipo do atributo original *getAttributeName() : Type***

Outro Exemplo



Regras de Transformação:

(1) **Para cada associação, há um atributo privado com o mesmo nome na classe destino**

(2) **Se a multiplicidade da associação for igual a zero ou um, o tipo deste atributo é igual a classe do lado original da associação. Se a multiplicidade for igual a zero ou mais, o tipo do atributo deve ser igual ao tipo Set.**

(3) **...**

Exemplo Linguagem de Transformação

Transformation PublicToPrivateAttributes (UML, UML) {

params

setterprefix: String = 'set'; getterprefix: String = 'get';

source

sourceAttribute : UML::Attribute;

target

targetAttribute : UML::Attribute; getter : UML::Operation; setter : UML::Operation;

source condition

sourceAttribute.visibility = VisibilityKind::public;

target condition

targetAttribute.visibility = VisibilityKind::private and

setter.name = setterprefix.concat(targetAttribute.name) and

setter.parameters->exists(p | p.name = targetAttribute.name and

p.type = targetAttribute.type) and setter.type = OclVoid and

getter.name = getterprefix.concat(targetAttribute.name) and

getter.parameters->isEmpty() and getter.type = targetAttribute.type and
targetAttribute.class = setter.class and

targetAttribute.class = getter.class;

bidirectional;

mapping

sourceAttribute.name <~> targetAttribute.name; sourceAttribute.type <~> targetAttribute.type;

}

Características desejáveis para Transformações MDA

1. **Tunability (Controle)**
   - Controle manual, condições sobre as transformações, parâmetros sobre transformações
2. **Rastreabilidade**
3. **Consistência Incremental**
4. **Bidirecionalidade (difícil!)**
   - Apenas uma transformação.
   - Duas transformações, sendo uma a inversa da outra.

- Tranformações podem ocorrer ambos sentidos, mas é necessário manter os modelos origem e destino consistentes

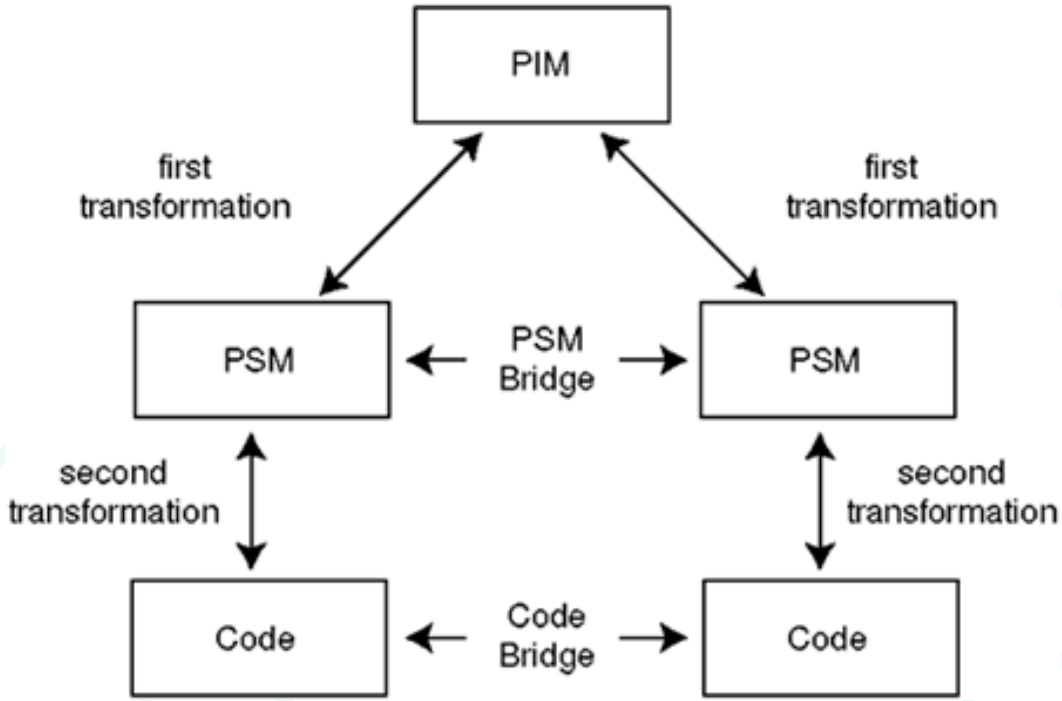
Modelos MDA

- **Uso de linguagens bem-definidas**
  - Forma (sintaxe) e significado (semântica) bem definidas.
  - Requisitos necessários para interpretação automática por computadores, ie, sem ambiguidades
  - MDA não é restrita apenas à UML
  - Segundo OMG, UML + OCL é a melhor opção para definição de modelos
    - Opções como “só UML” ou semântica de ações seriam alternativas, porém apresentam problemas

Benefícios - MDA

- **Produtividade**
  - A maior parte do código pode ser gerado
- **Portabilidade (Preservação do conhecimento)**
  - Transformação para diversos PSM
- **Manutenção e Documentação**
  - Modelos não abandonados (rastreabilidade 100%)
- **Separação de interesse**
  - Foco no desenvolvimento de um PIM
  - Transformações guardam detalhes técnicos
- **Interoperabilidade**
  - Geração de bridges entre os modelos

Interoperabilidade



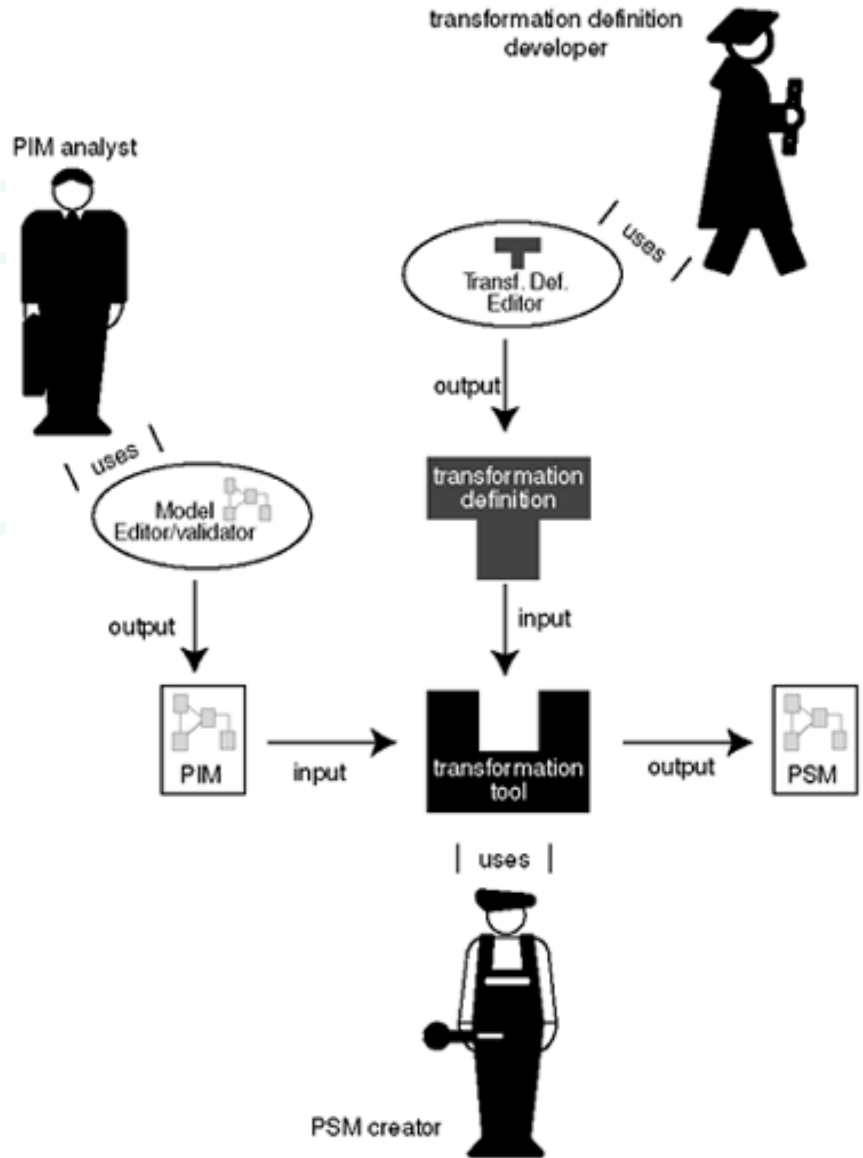
Classes Entidades	**Tabelas em Bancos de Dados Relacionais**
Class Pessoa	**Tabela PESSOA**

MDA & Processos de Desenvolvimento

- **MDA não específica um processo específico para seu uso**
  - OMG não se preocupa com padronização de processos
- **Modelos já possuem um importante papel no RUP**
- **Extremme Programming poderia se tornar Extremme Modelling**

Novos papeis

- **Analista PIM**
  - Necessidades do negócio
  - Modelo de negócio
- **Construtor PSM**
  - Detalhes da plataformas
  - Arquiteturas
- **Desenvolvedor de definição de transformações**
  - Escrita e compra



MDA hoje

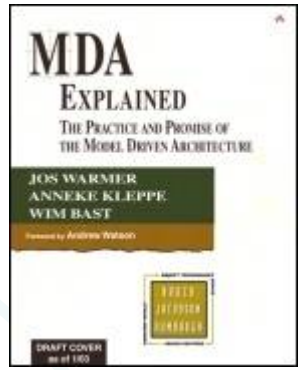
- **Uso de Padrões OMG para implementação MDA**
  - UML (Definições de Modelos)
  - OCL – Object Constraint Language
    - Linguagem para melhoria da definição de modelos UML através de restrições
  - MOF – Meta Object Facility
    - Definições de linguagens
- **OMG está num processo para definição de uma linguagem padrão para definição de transformações**

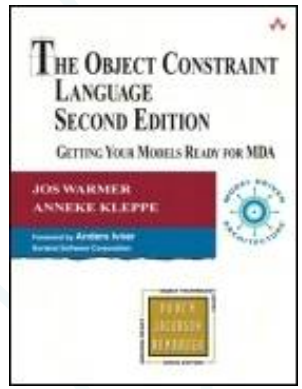
Ferramentas MDA

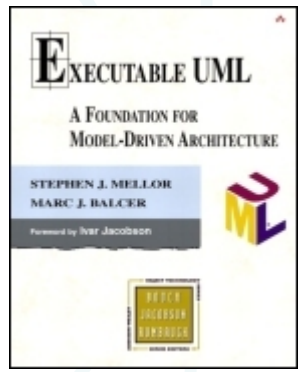
- **Nenhuma implementa MDA por completo**
  - IO Software ArcStyler
    - Disponível para download (Trial)
    - Uso de Cartridges
  - ComponentX
    - Versão Full & Free
  - OptimalJ
    - Versão Trial
  - MDA Metanology
    - Plugin para uso com Eclipse

Referências - Livros

- MDA Explained: The Pratice and Promise of Model Driven Architecture
- Convergent Architecture—Building Model-Driven J2EE Systems with UML
- The Object Constraint Language – 2nd Edition
- Executable UML – A Foundation for Model-Driven Architecture







Referências – Artigos e Sites

- **MDA website**
  - http://www.omg.org/mda

- **Artigos**
  - MDA Guide (Versão Maio 2003)