Transformações de Modelos

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

A Linguagem UML

- É uma notação gráfica (visual) para modelar sistemas
  - Não é uma linguagem de programação
  - Não há preocupação com detalhes semânticos
- Possui muitos diagramas
  - E é extensível

UML para PIM

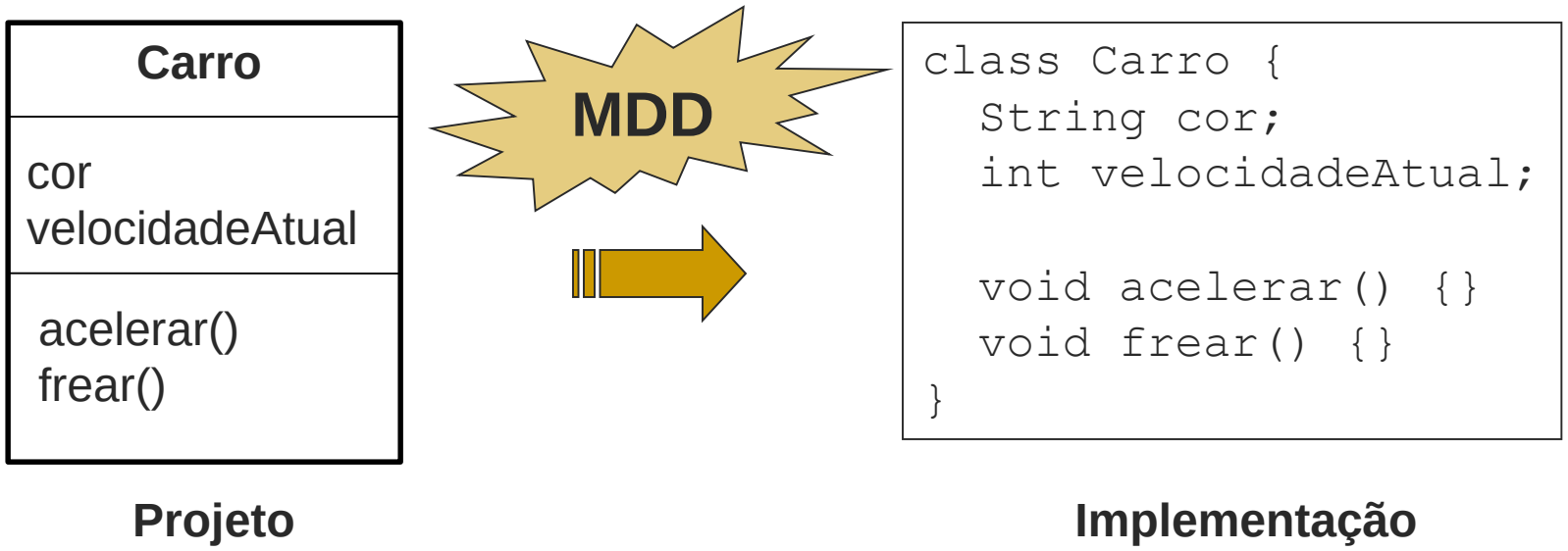
- Para obter PSM a partir de PIM, é preciso que a linguagem permita
  - Completude dos modelos
  - Consistência dos modelos
  - Modelos não ambíguos
- A UML é boa para modelar a parte estrutural
  - O ponto fraco da UML está nos diagramas comportamentais

Limitações da UML

- A UML oferece algum suporte para modelar o comportamento do sistema
  - Diagramas de Sequência, Estados, Atividades, Colaboração, etc.
- Entretanto, as definições destes diagramas não são suficientemente formais, completas e consistentes
  - Que tipo de código poderia ser gerado a partir de um Diagrama de Colaboração?

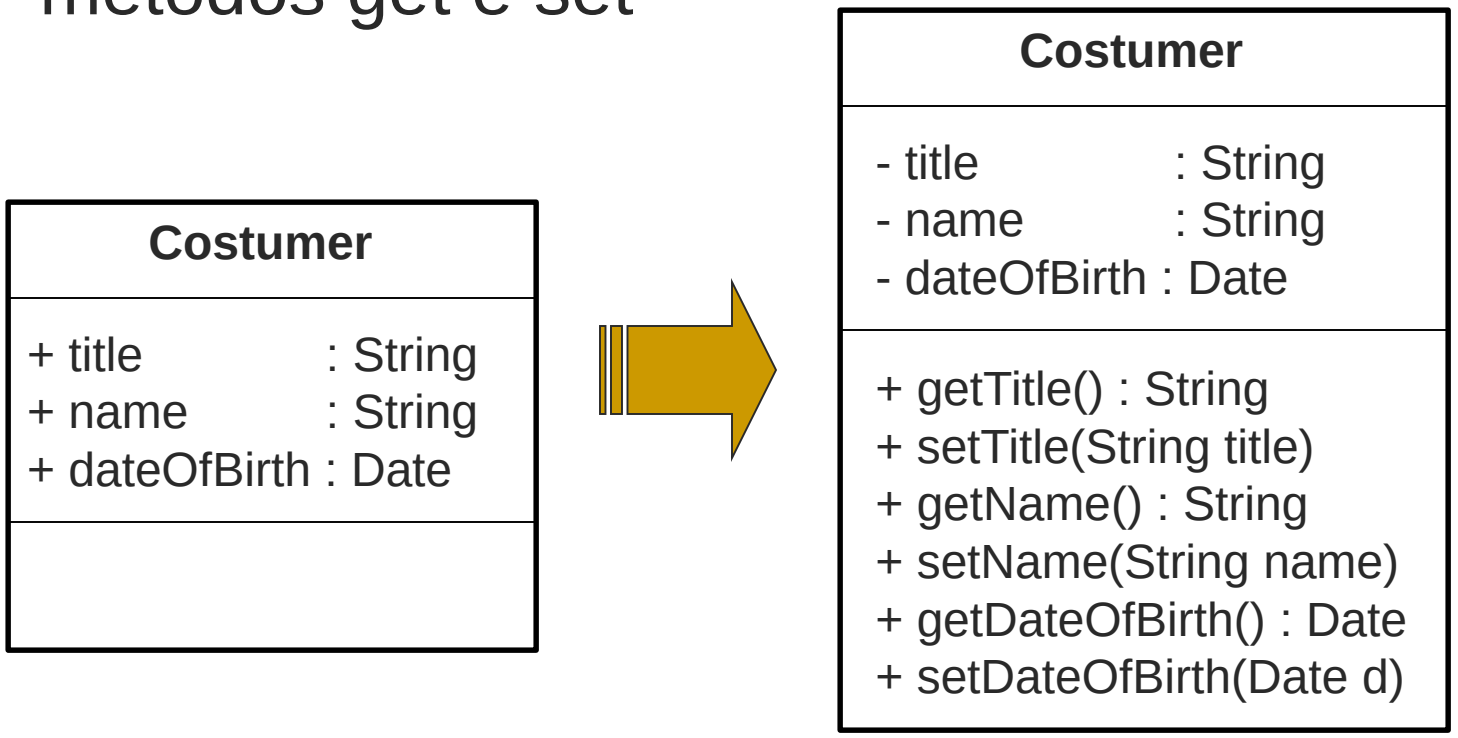
Do Projeto para Implementação

- O objetivo de MDD é obter o código a partir de modelos



Transformação de Modelos

- Transformação entre modelos UML
  - Atributos público passam a ter métodos get e set



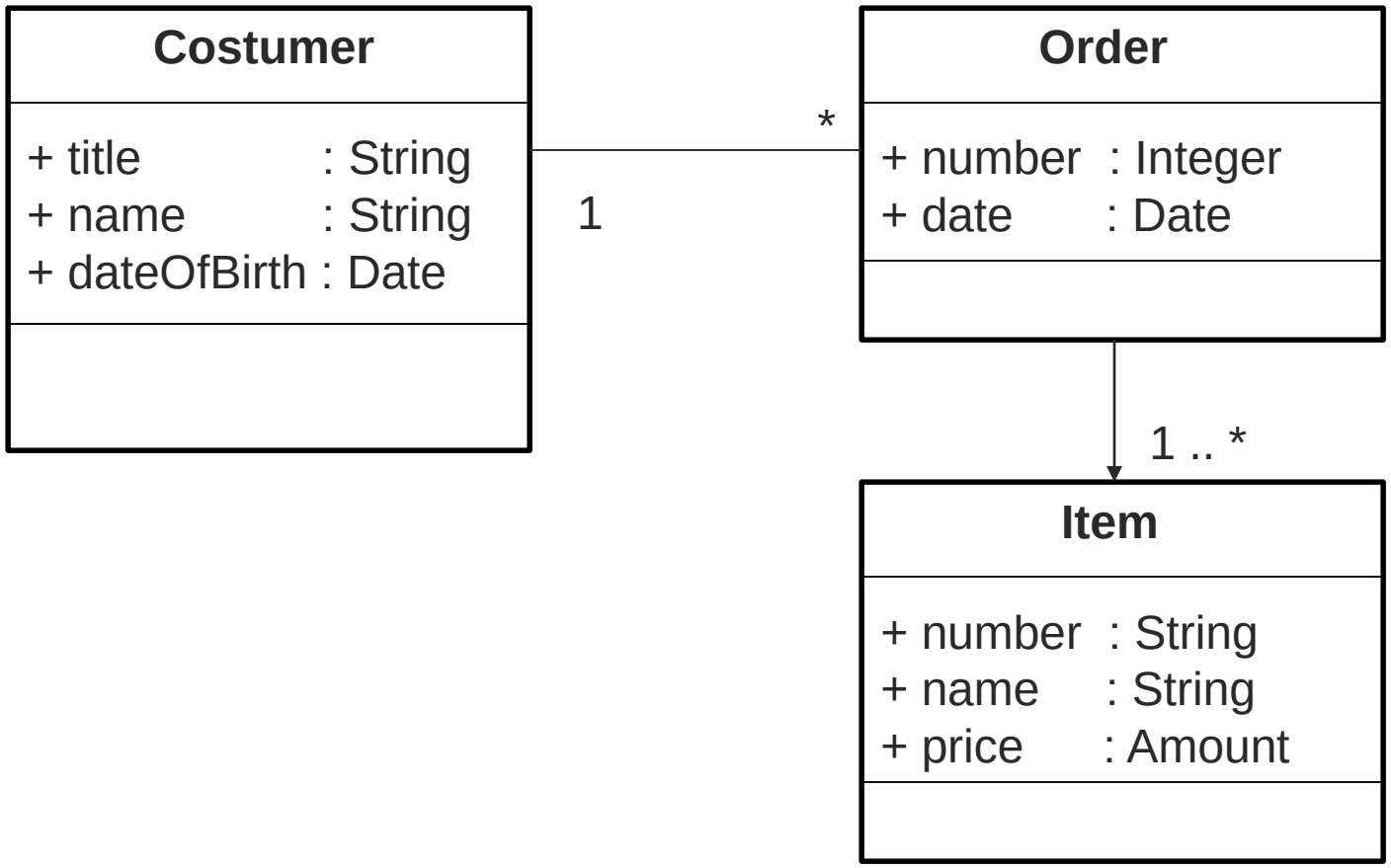
Justificativas da Transformação

- Em PIM de mais alto nível, é normal ter atributos públicos
  - Facilitam a leitura do diagrama
  - Significam propriedades que podem ter seus valores alterados
- Em PSM, os modelos são mais próximos do código
  - Atributos públicos são considerados um projeto ruim

Regras de Transformação

- Para cada classe C1 em PIM, existe uma classe C1 em PSM
- Para cada atributo público *a1 : Tipo1* em PIM, existe em PSM
  - um atributo privado *a1 : Tipo1*
  - um método público *getA1(): Tipo1*
  - um método público *setA1(Tipo1 t)*

Associação (PIM)



Associação (PSM)

Costumer
- title : String - name : String - dateOfBirth : Date - orders : Set
+ getTitle() : String + setTitle(String title) + getName() : String + setName(String name) + getDateOfBirth() : Date + setDateOfBirth(Date d) + getOrders() : Set + setOrders(Set o)

Order
- number : Integer - date : Date - costumer : Costumer - items : Set
+ getNumber() : Integer + setNumber (Integer n) + getDate() : Date + setDate(Date d) + getCostumer() : Costumer + setCostumer(Costumer c) + getItems() : Set + setItems(Set i)

Item
- number : Integer - name : String - price : Amount
+ getNumber() : Integer + setNumber (Integer n) + getName() : String + setName(String n) + getPrice() : Set + setPrice(Amount a)

Transformação de Associação

- Transformação de PIM para PSM
  - Transforma atributos públicos em privados
  - Elimina associações
- O diagrama resultante (PSM) é bem mais complexo que o diagrama PIM
  - É difícil identificar os relacionamentos e suas direções
  - Alguns relacionamentos não podem ser revertidos de PSM para PIM

Regras para Associação

- Para cada associação em PIM
  - Para cada alvo da associação, há um atributo privado na classe oposta
  - O tipo deste atributo é do tipo da classe alvo se a multiplicidade for 1
  - O tipo deste atributo é Set se a multiplicidade for maior que 1
  - Novos atributos terão métodos *get* e *set*
- Para associações direcionadas, os passos acima só se aplicam em uma direção

UML Executável (xUML)

UML Executável (xUML)

- xUML é um subconjunto UML que podem ser considerados modelos executáveis
  - Modelos executáveis agem exatamente como o código
- O objetivo de xUML é definir semântica precisa aos modelos UML

Definições de xUML

- Modelos de Domínio
  - Identificam os principais elementos do domínio do sistema e suas dependências
- Diagrama de Classes
  - Define as classes e associações entre elas
  - Detalha os atributos e métodos das classes

Definições de xUML

- Diagrama de Estados
  - Usado para descrever o ciclo de vida de uma classe
  - Detalha estados, eventos e transições
- Linguagem de Ações
  - Define operações que fazem algum processamento no modelo
  - É a principal forma de especificam a parte dinâmica dos modelos

Vantagens

- Modelos xUML podem ser compilados para uma linguagem de programação abstrata
- Modelos xUML podem ser testados
- Facilita a transformação de PIM para PSM

Problemas

- Diagrama de Estados somente é útil em alguns domínios
- A Linguagem de Ações não é de muito alto nível
  - Não difere muito do código em uma linguagem de programação
- A Linguagem de Ações não tem sintaxe e notação padronizadas

UML e OCL

- OCL (*Object Constraint Language*) é uma linguagem declarativa
  - Permite descrever regras que se aplicam a diagramas UML
- Parte do comportamento dinâmico do sistema pode ser especificado em OCL
  - Pré-condições e pós-condições são descritas para as operações

Bibliografia da Aula

- A. Kleppe, J. Warmer, W. Bast. **MDA Explained: The Model Driven Architecture: Practice and Promise**. Addison-Wesley, 2003.
  - Capítulo 2
- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - 5.5.2 UML Executável