Desenvolvimento Dirigido por Modelos (MDD)

Eduardo Figueiredo
http://www.dcc.ufmg.br/~figueiredo
reuso.software@gmail.com

21 Maio 2012

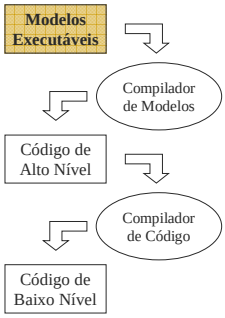
Agenda Atual do Curso

Aula	Data	Assunto
23	28/05	Avaliação Experimental de Reuso
24	30/05	Semana da PPGCC (**ñ há aula**)
25	04/06	Apresentações de Monografia (1)
26	06/06	Apresentações de Monografia (2)
27	11/06	Apresentações de Trabalhos (TP)
		... Revisão + Prova 2 (P2) ...

Agenda da Aula

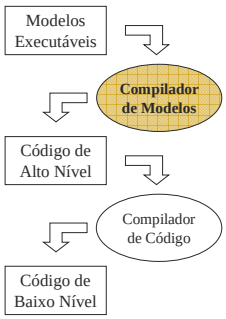
- Conceitos fundamentais
- O Processo MDD
- Transformações de modelos
- UML Executável
- Ferramentas para MDD

Abordagem MDD



- Os modelos são independentes de software
  - Assim como, código de alto nível é independente de hardware

Abordagem MDD



- Modelos podem ser compilados para várias linguagens de programação
  - Modelos podem ser parcialmente ou totalmente reusados em diferentes contextos

Manutenção e Geração

- Quando for necessário fazer manutenção do software
  - A manutenção deve ser feita nos modelos
- Se for necessário mudar a tecnologia de implementação
  - Basta re-gerar o sistema a partir dos modelos para uma tecnologia diferente

Argumentos a Favor

- Permite que desenvolvedores pensem em alto nível de abstração
  - Reduz a probabilidade de erros
  - Acelera o projeto e implementação
- Criação (de parte) do sistema independente de plataforma
  - Facilidade de portar o sistema para uma nova plataforma

Argumentos Contrários

- Bibliotecas de código estão disponíveis
  - Elas podem não se adequarem aos modelos construídos
- Independência de plataforma somente é importante em sistemas de vida útil longa
- O principal custo de sistemas complexos não está na implementação
  - Engenharia de requisitos é mais cara

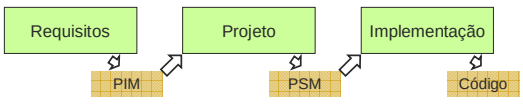
O Processo MDD

O Processo MDD

- O processo de desenvolvimento é focado na atividade de modelagem
- Três tipos de modelos
  - Modelos independentes de plataforma (**PIM)**
  - Modelos específicos de plataforma (**PSM**)
  - Código

Modelo de Processo

- O modelo MDD parece com processos tradicionais de desenvolvimento
  - A diferença crucial está no grau de automação (transformação de modelos)
  - Em particular, de PIM para PSM



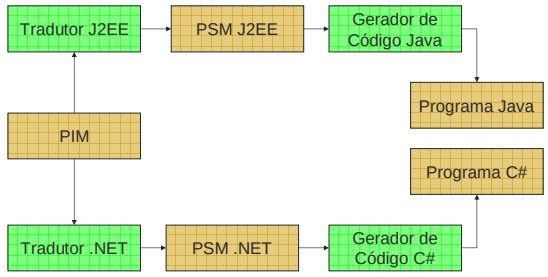
PIM

- Modela abstrações importantes do domínio
- Vários PIMs podem ser criados
  - Refletem diferentes visões do sistema
- Geralmente são criados em UML

PSM

- Criados geralmente pela transformação de PIM
- Podem haver camadas de PSM
  - Cada camada acrescenta mais detalhes aos modelos
- Exemplo
  - Camada independente de BD
  - Camada específica para o BD

Exemplo de Tradutores



O Processo na Visão MDD

1. Selecionar modelos existentes
2. Escolher partes dos modelos que interessam ao sistema
   - Pode ser necessário projetar novos modelos ou adaptar os modelos existentes
3. Integrar as partes selecionadas dos modelos
4. Pegar uma tecnologia de implementação
5. Descrever (ou reusar) o mapeamento dos modelos para a implementação
6. Gerar o sistema

Benefícios Esperados

- Produtividade
  - Geração de código
  - Trabalho em um nível mais abstrato
- Custo menor
  - Reuso em larga escala
- Maior vida útil
  - Modelos têm vida útil maior que código

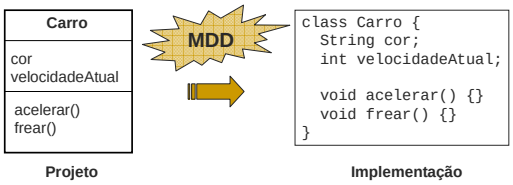
Benefícios Esperados

- Portabilidade
  - PIM são portáveis
- Interoperabilidade
  - MDD gera não somente PIM, mas os canais de comunicação
- Documentação
  - Modelos são documentos
- Manutenção mais barata

Exemplo de MDD

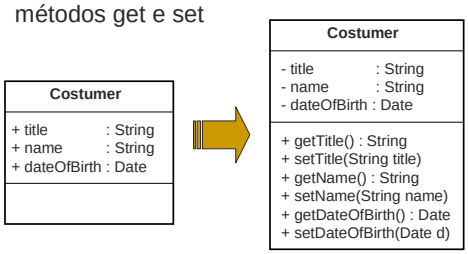
Do Projeto para Implementação

- 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



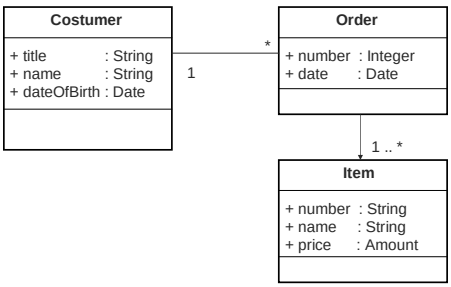
Justificativas da Transformação

- 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

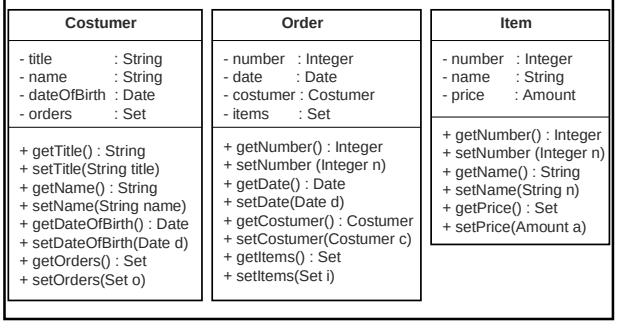
Regras de Transformação

- 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)



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 0 ou 1
  - O tipo deste atributo é Set se a multiplicidade for maior que 1
  - Os novos atributos terão métodos *get* e *set* (vide regras anteriores)
- Para associações direcionadas, os passos acima só se aplicam na direção da aresta

UML Executável (xUML)

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?

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

Ferramentas para MDD

Existe MDD na prática?

- Poucos sistemas ainda são desenvolvidos usando a filosofia MDD
  - A expectativa é de aumentar a adoção nos próximos anos a medida que MDD amadurece
- Desde que MDD foi proposta, vários ferramentas afirmam apoiar MDD
  - Na verdade, as ferramentas apóiam alguns aspectos de MDD

Dificuldades de Automação

- Ferramentas MDD devem considerar casos particulares
- Ambiente de execução inclui
  - Plataforma de programação (ex. J2EE)
  - Bibliotecas específicas da empresa ou do domínio
  - Bibliotecas específicas de interface com o usuário, etc.

Tipos de Ferramentas

- Transformação de PIM para PSM
- Transformação de PSM para Código
- Transformação de PIM para Código
- Ferramentas para definir transformações
  - Outras

Ferramentas: PIM para PSM

- Tipo de ferramenta que recebe PIM de alto nível e transforma em um ou mais PSM
  - Ferramentas deste tipo quase não existem

PSM para Código

- As ferramentas mais conhecidas para suporte a MDD
  - Recebem um ou vários modelos como entrada
  - Geram código em uma determinada linguagem (modelo de código)
- Algumas ferramentas mantém a consistência entre modelos e código

PIM para Código

- Tipo de ferramenta que suporta
  - Transformação de PIM para PSM
  - Transformação de PSM para Código
- Os usuários podem ver somente a transformação PIM para Código
- UML é geralmente usada como uma linguagem para PIM
  - Comportamento nem sempre é expresso em UML (manual ou OCL)

Exemplos de Ferramentas

- xUML-Compiler
- IBM Rational Rhapsody
- AndroMDA

xUML Compiler

http://code.google.com/p/xuml-compiler/

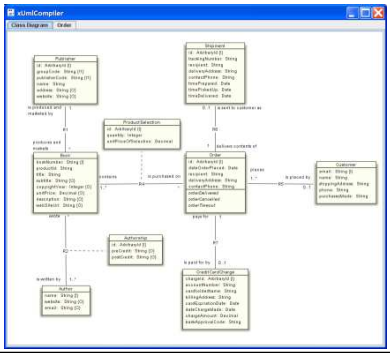
xUML Compiler

- É um projeto *open source*
- Visão gráfica
  - Diagrama de Classes
  - Diagrama de Estados
- Gera código executável e testável (Java)
  - Gera documentação parcial (Java Doc)

Linguagem de Ações

- É escrita em sintaxe Java
  - Linguagem de Ações é tão poderosa e expressiva quanto Java
- Elementos da Linguagem de Ações são inseridos nas classes

Diagrama de Classes



IBM Rational Rhapsody

http://www.ibm.com/developerworks/rational/products/rhapsody/

Rhapsody

- Ambiente de desenvolvimento para MDD
  - Suporta C, C++ e Java
- Versão atual 7.5.3
  - Permite analisar e verificar rastreabilidade entre requisitos
  - Validar as funcionalidades antecipadamente no desenvolvimento
  - Conduzir testes nos modelos

Demonstração

- Modelos de alarme para uma casa

http://www.ibm.com/developerworks/rational/products/rhapsody/

  - Getting started with IBM Rational Rhapsody
    (Start a project and create diagrams with IBM Rational Rhapsody)

AndroMDA

http://www.andromda.org/

AndroMDA

- Framework para intensa geração de código
  - Adere aos principais conceitos de MDD
- Modelos UML são transformados em componentes implantáveis
- Gera código compatível com tecnologias atuais
  - J2EE, Spring, Struts, JSF, Spring and Hibernate, etc.

Modelos de Entrada (UML)

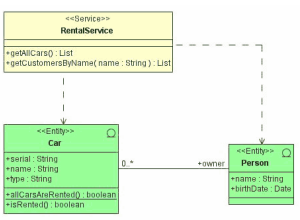
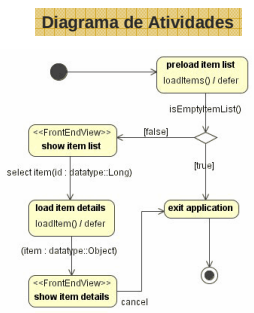
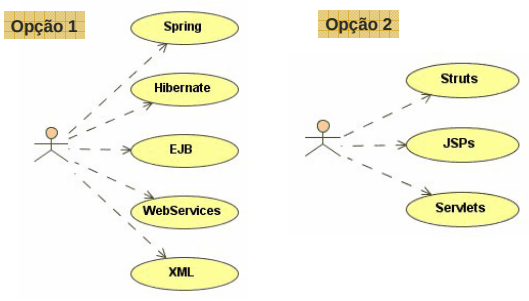


Diagrama de Classes



Tecnologia de Implementação



Bibliografia da Aula

- A. Kleppe, J. Warmer, W. Bast. **MDA Explained: The Model Driven Architecture: Practice and Promise**. Addison-Wesley, 2003.
  - Capítulos. 1 e 2
- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - 5.5 Engenharia Dirigida por Modelos

Próxima Aula...

- Laboratório 2011
- Assunto
  - Desenvolvimento Dirigido por Modelos