MDA

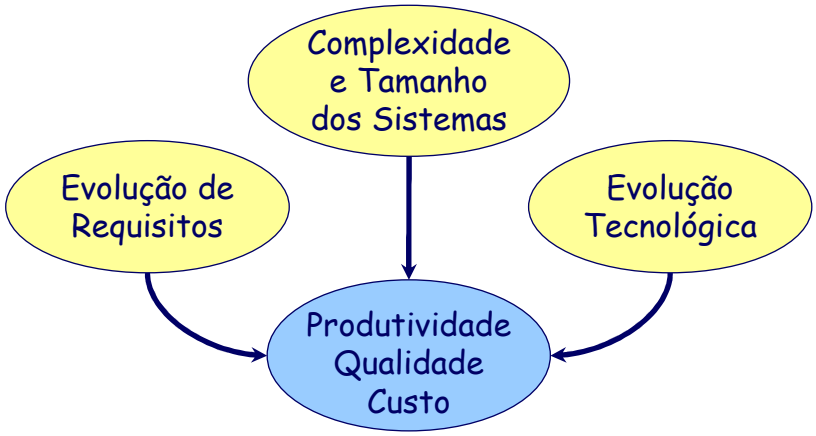
Padrões, Tecnologias e Desafios

Glêdson Elias
gledson@compose.ufpb.br
gledson@di.ufpb.br

SBCARS 2007

Motivação

- Realidade Atual



Motivação

- Abordagens Tradicionais
  - Ainda adotam processos de desenvolvimento com pouca abstração, reuso e automação
    - Abstração e reuso ajudam a tornar o software um artefato (asset), similar aos de outras engenharias
    - Automação pode gerar código a partir de modelos
  - Mapeamento de modelos para código executável é realizado de forma manual
    - Foco não exclusivo no domínio do problema
    - Detalhes complexos das plataformas alvo representam grande parte do esforço de desenvolvimento

Motivação

- Abordagens Tradicionais
  - Modelos desenvolvidos não são propriamente mantidos, atualizados e reusados
    - Modelos são usados apenas como documentação na fase inicial
    - Codificação torna-se o foco do desenvolvimento
    - Mudanças no código não são refletidas e sincronizadas com os modelos
    - Modelos tornam-se obsoletos
    - Manutenção e evolução dos sistemas tornam-se difíceis, caras e demoradas

Motivação

- Abordagens Tradicionais
  - Adaptar um sistema a uma nova plataforma é quase tão difícil quanto desenvolver desde o princípio
  - Adotam pouca ou nenhuma transformação automática de modelos
    - Poucas linguagens de transformação definidas
    - Poucas ferramentas

Motivação

- MDD (Model-Driven Development)
  - É uma abordagem para usar modelos no desenvolvimento de sistemas software
    - Incrementa o poder dos modelos
  - Explora modelos para guiar o entendimento, projeto, construção, implantação, operação, manutenção e evolução dos sistemas
  - Explora a separação de interesses arquiteturais para alcançar os objetivos
  - Promove a separação da arquitetura do sistema de sua implementação

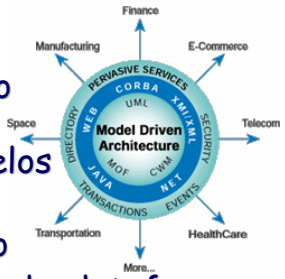
MDA é um framework para desenvolvimento dirigido a modelos

Agenda

- Introdução
- Conceitos
- Visões & Tipos de Modelos
- Transformação de Modelos
- Abordagens de Transformação
- Sistema Multiplataforma
- Múltiplas Transformações
- Reuso de Mapeamentos
- Processo MDA
- Padrões & Ferramental
- Considerações Finais

Introdução

- MDA (Model-Driven Architecture)
  - Um framework para um conjunto de padrões de suporte ao desenvolvimento dirigido a modelos
  - Modelos são centrais em todo o ciclo de vida do desenvolvimento
  - Foco em modelos independentes de plataforma
    - Abstrai detalhes de tecnologias e plataformas específicas
    - Middleware alvo não é importante ou o foco
  - Está sendo padronizada pela OMG (Object Management Group)

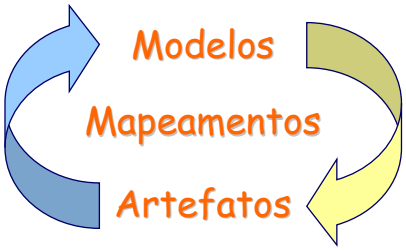


Introdução

- MDA (Model-Driven Architecture)
  - É apenas um outro passo evolucionário no desenvolvimento da área de software
  - Automação do software a partir de modelos é apenas um outro nível de compilação
    - Compiladores podem traduzir modelos para código em linguagens de alto nível
    - Abordagem pode ser usada para plataformas existentes, como também para novas plataformas futuras

Introdução

- MDA (Model-Driven Architecture)
  - Tem por objetivo incrementar o nível de reuso e abstração
  - Artefatos (código, testes, documentação ...) são automaticamente derivados dos modelos através de mapeamentos



- ✓ Portabilidade
- ✓ Interoperabilidade
- ✓ Reusabilidade
- ✓ Qualidade
- ✓ Redução de custos

Introdução

- MDA (Model-Driven Architecture)
  - Separação de Interesses
    - Separa a especificação do sistema de sua operação e dos detalhes da forma como o sistema usa as facilidades da plataforma
    - MDA provê uma abordagem para...
      - Especificar um sistema de forma independente da plataforma adotada
      - Especificar plataformas
      - E, uma vez selecionada uma dada plataforma, transformar a especificação do sistema em uma específica da plataforma

Introdução

- Benefícios MDA
  - Protege investimento da evolução tecnológica
    - Modelos independentes de plataforma são preservados e reusados em novas plataformas
  - Facilita tratar a evolução de requisitos
    - Modelos são mais fáceis de manipular do que código em função do maior nível de abstração
    - Modelos podem ser simulados e transformados automaticamente em código
  - Permite manipular sistemas grandes e complexos
    - Separação de interesses abstrai detalhes facilitando o desenvolvimento
    - Integra aspectos independentes (segurança, confiabilidade, desempenho) uniformemente

Conceitos

- Sistema
  - É qualquer coisa de interesse como um todo ou como composto de partes (RM-ODP, 1996)
    - Um componente de um sistema pode também ser considerando um sistema (subsistema)
  - Pode ser . . .
    - Um sistema existente
    - Um sistema que está sendo planejado
    - Um sistema a ser modificado

Conceitos

- Modelo
  - É uma descrição ou especificação de alguma coisa com algum propósito (Miller, 2003)
  - É uma descrição de um sistema ou parte dele, escrito usando uma linguagem bem definida (Kleppe, 2003)
  - Um conjunto de sentenças (true/false) sobre um sistema (Seidewitz, 2003)
  - É uma representação de uma parte da função, estrutura ou comportamento de um sistema (MDA Guide, 2001)

> É uma descrição ou especificação de um sistema e seu ambiente para um determinado propósito
>
> (MDA Guide, 2003)

Conceitos

- Modelo
  - Geralmente é apresentado como uma combinação de desenhos e textos
    - Pode ser em linguagem de modelagem ou natural
  - Permite especificar diferentes aspectos de um sistema em diferentes modelos e níveis de abstração
  - Torna possível entender um sistema existente
  - Define um vocabulário comum que facilita a comunicação dos envolvidos

Conceitos

- Modelo
  - Permite validar um sistema
    - Erros de projeto podem ser detectados
      - Erros são mais baratos neste estágio
    - Protótipo do sistema pode ser avaliado
      - Modelos também podem ser executados
    - Propriedades do sistema podem ser formalmente analisadas
      - Quanto mais exato o modelo, maiores as chances do sistema construído ser o esperado
  - Pode guiar a implementação gerando esqueletos e programas completos

Conceitos

Selic, 2003

- Modelo
  - Características
    - Abstrato
      - Enfatiza aspectos relevantes e esconde aspectos irrelevantes
    - Inteligível
      - Expresso de uma forma fácil de ler e entender
    - Preciso
      - Representa o sistema de forma confiável
    - Dedutível
      - Pode ser usado para derivar conclusões sobre o sistema
    - Barato
      - É mais barato construir e estudar o modelo

Conceitos

- Arquitetura
  - É uma especificação das partes e conectores do sistema e as regras para as interações das partes usando os conectores (Shaw and Garlan, 1996)
  - Um conjunto de regras para definir a estrutura de um sistema e os relacionamentos entre as partes (RM-ODP, 1996)

O que significa
ARQUITETURA
em MDA?

Conceitos

- Arquitetura em MDA
  - Separa a arquitetura do sistema de sua implementação
  - Estende o significado central de arquitetura
  - Considera a arquitetura de um sistema de modelos
    - Modelos são conectados e estruturados por relacionamentos de transformação
    - Especifica os tipos de modelos a serem usados e como podem ser preparados
    - Define os relacionamentos entre os diferentes tipos de modelos

Conceitos

- Ponto de Vista (Viewpoint)
  - É uma técnica de abstração que usa um conjunto selecionado de conceitos arquiteturais e regras estruturais
    - Foca em aspectos de interesse do sistema
    - Esconde detalhes irrelevantes do sistema
    - Os conceitos e as regras formam a linguagem da visão (viewpoint language)

Abstração é o processo de remover detalhes selecionados irrelevantes para obter um modelo simplificado

Conceitos

- Ponto de Vista (Viewpoint)
  - Cada ponto de vista é uma forma diferente de preparar um modelo de um sistema
  - O mesmo sistema pode ser visto sob diferentes pontos de vista
- Visão (View / Viewpoint Model)
  - É uma representação do sistema na perspectiva de um determinado ponto de vista

O termo visão é geralmente usado para representar o ponto de vista (viewpoint) e a visão (view/viewpoint model)

Conceitos

- Plataforma
  - É um conjunto de subsistemas e tecnologias que provêem um conjunto coerente de funcionalidades através de interfaces e padrões de uso especificados
    - Qualquer aplicação suportada pela plataforma pode usar as funcionalidades da plataforma
    - Aplicação não precisa conhecer os detalhes de como as funcionalidades são implementadas

Conceitos

- Plataforma
  - Plataforma Genérica
    - Objetos, Batch, Dataflow
  - Plataforma Específica de Tecnologia
    - CORBA, CCM, Java 2 Components
  - Plataforma Específica de Fabricante
    - CORBA
      - Borland Visibroker, Iona Orbix
    - Java 2 Components
      - BEA Weblogic, IBM WebSphere
    - Microsoft .NET

Conceitos

- Independência de Plataforma
  - É uma qualidade que um modelo pode possuir de ser independente das características de qualquer tipo de plataforma
  - Pode ter diferentes graus de independência
    - Considera características de um determinado tipo de plataforma genérica, plataforma específica de tecnologia ou plataforma específica de fabricante
    - Considera uma, algumas ou todas as características da plataforma alvo

Conceitos

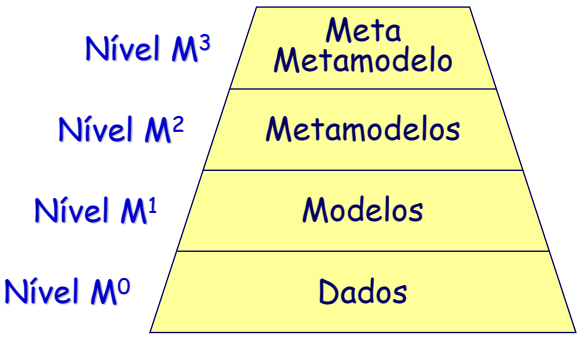
- Metamodelo
  - É um modelo de modelos (MDA Guide, 2001)
  - É um modelo de uma linguagem bem definida (Kleppe, 2003)
  - É um modelo que define uma linguagem de especificação de modelos (MOF, 2000)
  - É um modelo de uma linguagem de modelagem (Seidewitz, 2003)
  - Define o que pode ser expresso em modelos válidos especificados com uma dada linguagem de modelagem
  - É um modelo cujas instâncias são tipos em outro modelo

Conceitos

- Metametamodelo
  - É um modelo que define a linguagem de especificação de um metamodelo
  - Relacionamento entre um metametamodelo e um metamodelo é análogo ao relacionamento entre um metamodelo e um modelo

Conceitos

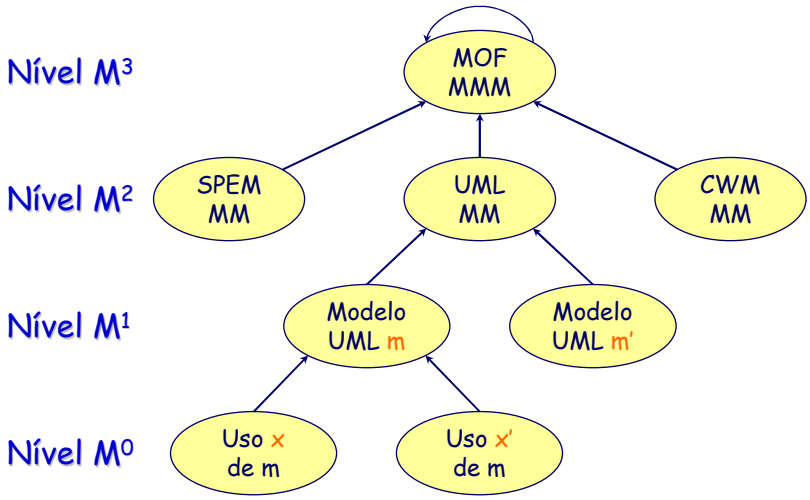
- Arquitetura em Camadas



Padrões para representar modelos, metamodelos e metametamodelos formam a espinha dorsal da infra-estrutura MDA

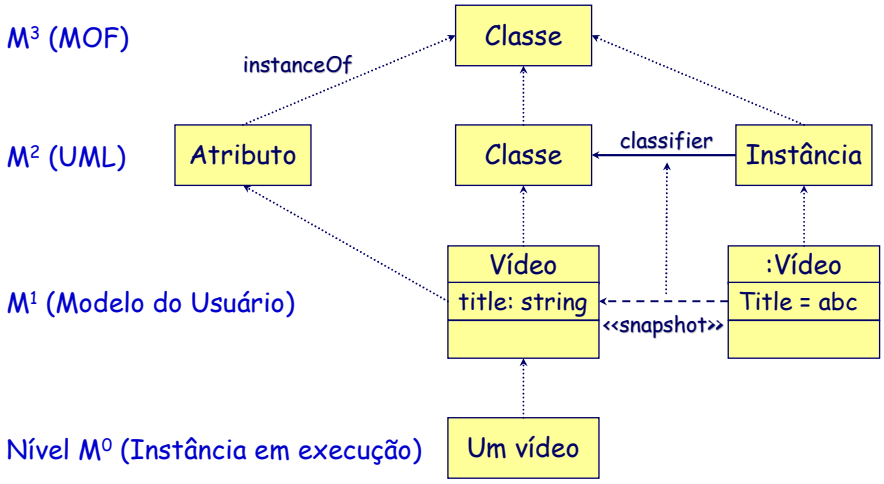
Conceitos

- Hierarquia de Metamodelos



Conceitos

- Hierarquia de Metamodelos



Visões MDA

- Computation Independent Viewpoint
  - Visão Independente de Computação
- Platform Independent Viewpoint
  - Visão Independente de Plataforma
- Platform Specific Viewpoint
  - Visão Dependente de Plataforma

Visões MDA

- Computation Independent Viewpoint
  - Foca no sistema, no ambiente do sistema e nos requisitos do sistema
  - Esconde detalhes da arquitetura do sistema
  - Detalhes da estrutura e processamento do sistema são escondidos ou ainda indeterminados
  - Define e usa um vocabulário que é familiar aos envolvidos no domínio do problema

Visões MDA

- Platform Independent Viewpoint
  - Foca nas operações do sistema enquanto esconde os detalhes necessário para uma determinada plataforma
  - Permite que parte da especificação de um sistema não mude de uma plataforma para outra
  - Pode adotar uma linguagem de modelagem de propósito geral ou uma linguagem específica da área na qual o sistema será usado

Visões MDA

- Platform Specific Viewpoint
  - Combina a visão independente com os detalhes do uso de uma plataforma específica
    - Adiciona os detalhes inerentes a uma dada plataforma
  - Representa tudo que é representado na visão independente
    - Não significa que o que aparece em uma visão também aparece na outra
    - As visões podem ser bastante diferentes, mas representam as mesmas coisas
  - Inclui representações mais detalhadas e representações de aspectos não representados na visão independente

Tipos de Modelos

- CIM (Computation Independent Model)
  - Modelo Independente de Computação
- PIM (Platform Independent Model)
  - Modelo Independente de Plataforma
- PSM (Platform Specific Model)
  - Modelo Específico de Plataforma
- PM (Platform Model)
  - Modelo de Plataforma

Tipos de Modelos

- CIM
  - É uma visão do sistema do ponto de vista independente de computação
  - Mostra o sistema no ambiente no qual irá funcionar ou será usado
  - Auxilia a entender e apresentar o que o sistema deve fazer (entender o problema)
  - Modela os requisitos funcionais do sistema
    - Requisitos devem ser posteriormente refletidos no PIM e PSM
  - Também denominado modelo de domínio ou modelo de negócio

Tipos de Modelos

- CIM
  - Considera apenas as regras de negócio
  - Não mostra detalhes da estrutura do sistema, que ainda podem ser indeterminados
  - Não precisa indicar a adoção de sistemas computacionais
  - É independente de como o sistema será implementado
  - Adota e define um vocabulário compartilhado entre os profissionais do domínio
    - Vocabulário pode passar a ser usado nos demais modelos

Tipos de Modelos

- CIM
  - O principal usuário do CIM é o profissional do domínio
    - Usuário não domina os modelos e artefatos usados para concretizar as funcionalidades
  - Aproxima especialistas do domínio e especialistas de projeto de software
    - Especialistas do domínio conhecem os requisitos funcionais do sistema
    - Especialistas de projeto de software sabem concretizar os requisitos funcionais em artefatos de software

Tipos de Modelos

- PIM
  - É uma visão do sistema do ponto de vista independente de plataforma
  - Não mostra detalhes da plataforma ou tecnologia adotada para concretizar o sistema
  - Exibe um certo grau de independência de plataforma
  - É adequado para uso em diferentes plataformas de um tipo similar
  - Pode ser adequado a um ou vários estilos arquiteturais
  - É construído baseado no CIM

Tipos de Modelos

- PSM
  - É uma visão do sistema do ponto de vista específico da plataforma
  - Especifica como o sistema faz uso da plataforma alvo
    - Inclui informações sobre a plataforma específica usada na concretização do sistema
    - Possivelmente contém elementos que são específicos da plataforma
  - Provê mais ou menos detalhes da plataforma alvo em função do seu propósito
  - Especifica o mesmo sistema que o PIM

Tipos de Modelos

- PSM
  - Pode ser uma implementação
    - Especificação que provê toda a informação necessária para construir e colocar o sistema em operação
  - Pode incluir . . .
    - Código do programa
    - Tipos requeridos pela implementação
    - Descritores de implantação
    - Especificações de configuração

Tipos de Modelos

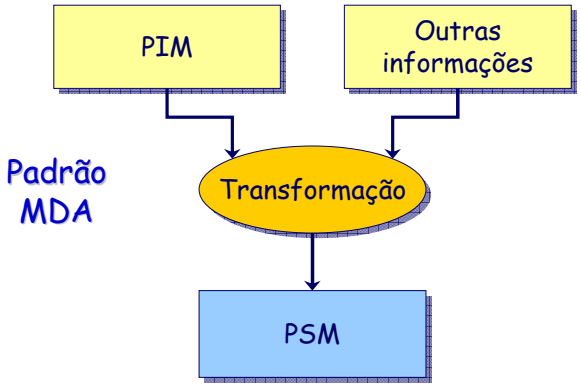
- PM
  - Modelo detalhado que descreve a plataforma
  - Provê um conjunto de conceitos técnicos, representando os diferentes tipos de partes e serviços que compõem a plataforma
  - Também especifica requisitos na conexão e uso de partes da plataforma
  - Descreve os tipos de elementos que podem ser usados na especificação do uso da plataforma

Tipos de Modelos

- PM
  - Deve ser escolhida uma plataforma (ou várias) que permita a implementação do sistema com as qualidades arquiteturais desejadas
  - Atualmente, encontra-se na forma de manuais de software ou hardware, ou até mesmo na mente do arquiteto
  - Um modelo de plataforma genérico pode focar em um estilo arquitetural específico

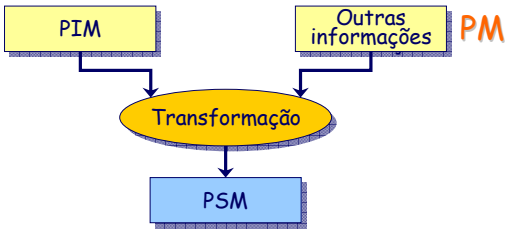
Transformação de Modelos

- O que é?
  - É o processo de converter um modelo de um sistema em outro modelo do mesmo sistema



Transformação de Modelos

- Padrão MDA
  - PIM e outras informações são combinadas pela transformação para produzir o PSM
  - Transformação é realizada de forma manual, automática ou com assistência de ferramenta
  - A transformação produz um modelo específico para uma determinada plataforma

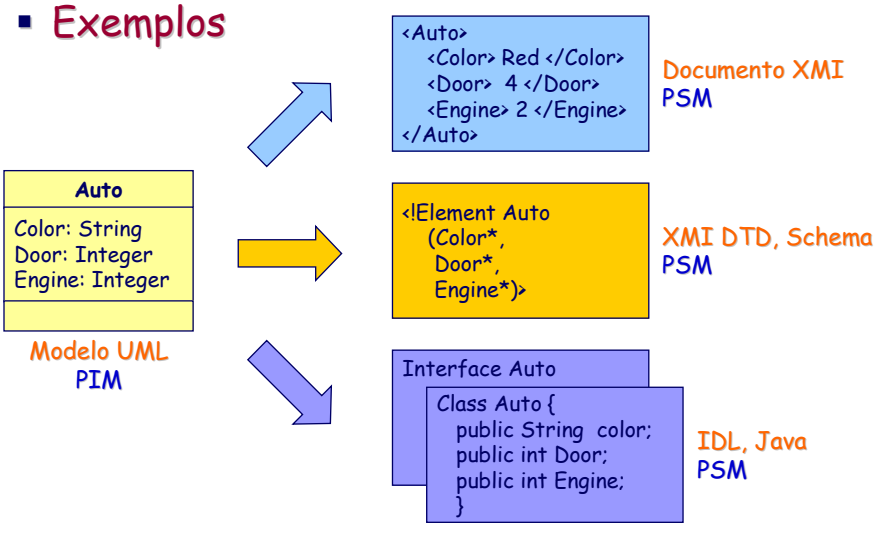


Transformação de Modelos

- Padrão MDA
  - Transformação de PIM para PSM adiciona detalhes específicos da plataforma
    - PIM pode ser transformado em código, sem produzir o PSM
    - PSM pode ser produzido para entendimento ou depuração do código gerado
  - Um PIM especifica uma classe de sistemas equivalentes para diferentes plataformas
    - Sistemas são equivalentes pois correspondem ao mesmo PIM
  - Um PSM produzido usando um PIM é membro de sua classe de sistemas

Transformação de Modelos

- Exemplos



Transformação de Modelos

- Função de Mapeamento
  - Provê especificações para transformação de um PIM para um PSM de uma dada plataforma
  - Um PM determina a natureza do mapeamento

✓ Um modelo de plataforma para EJB inclui as interfaces Home e Remote como também as classes Bean e Container Managed Persistence

✓ Um mapeamento de PIM UML para EJB provê marcas que devem ser usadas para guiar a transformação de PIM para PSM

✓ Marcar uma classe com a marca Session resulta na transformação da classe em um bean de sessão e outras classes de suporte

Transformação de Modelos

- Registro da Transformação
  - Resultado da transformação é o PSM e o registro da transformação
  - Mostra o mapeamento de cada elemento PIM para os correspondentes elementos PSM
    - Mostra quais partes do mapeamento foram usadas para cada parte da transformação

✓ Um registro de transformação pode mostrar que uma dada classe no PIM tornou-se três classes no PSM, relacionadas de alguma forma

✓ Um registro de transformação pode mostrar que dois objetos conectados diretamente no PIM são conectados no PSM via dois objetos de protocolo e um objeto interceptador

Transformação de Modelos

- Registro da Transformação
  - Pode ser disponibilizado para os profissionais que manipulam o PIM ou o PSM
  - Pode ser usado para manter sincronizado o PIM e o PSM em relação a mudanças em ambos
    - Mudanças no PIM são refletidas no PSM
    - Mudanças no PSM são refletidas no PIM

Agenda

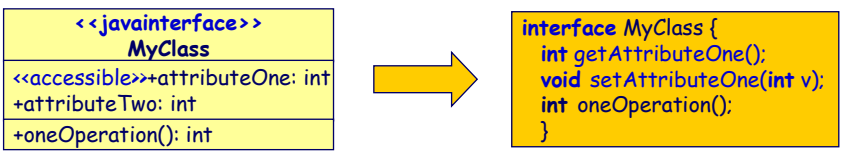
- Introdução
- Conceitos
- Visões & Tipos de Modelos
- Transformação de Modelos
- Abordagens de Transformação
- Múltiplas Transformações
- Reuso de Mapeamentos
- Processo MDA
- Padrões & Ferramental
- Considerações Finais

Abordagens de Transformação

- Transformação de Instâncias
  - Identifica elementos no PIM que devem ser transformados de uma determinada forma para o PSM de uma dada plataforma alvo
  - Adota anotações ou marcas (marks) para indicar os elementos no PIM que devem ser transformados
  - A colocação de marcas no PIM origina um novo modelo do sistema chamado PIM anotado ou PIM marcado (marked PIM)

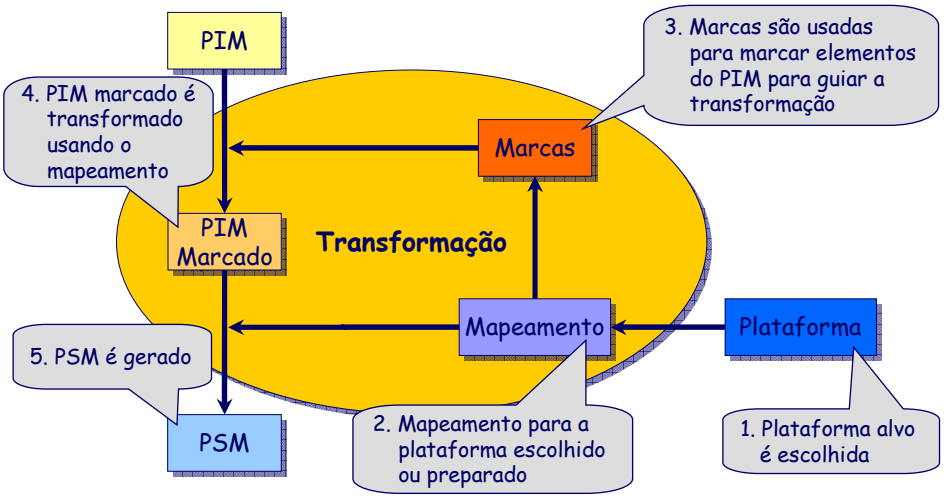
Abordagens de Transformação

- Transformação de Instâncias
  - Marca
    - Representa um conceito no PSM
    - É aplicada a um elemento do PIM para indicar como o elemento deve ser transformado
    - Não é parte do PIM, pois é específica de plataforma



Abordagens de Transformação

- Transformação de Instâncias

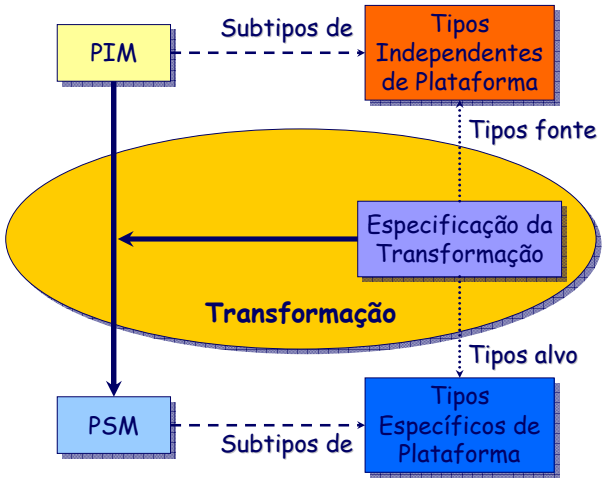


Abordagens de Transformação

- Transformação de Modelo
  - Especifica o mapeamento de modelos construídos usando tipos independentes de plataforma para modelos expressos usando tipos específicos de plataforma
  - Define regras e ou algoritmos expressos em termos de tipos independentes e específicos de plataforma

Abordagens de Transformação

- Transformação de Modelo

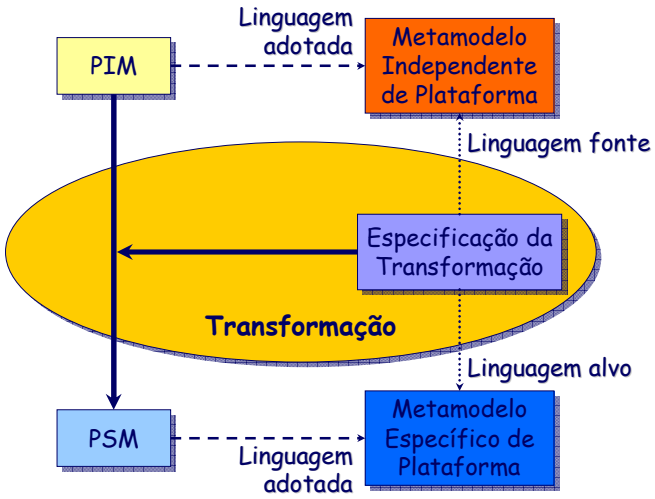


Abordagens de Transformação

- Transformação de Metamodelo
  - Caso especial de transformação de modelo
  - Tipos de elementos do PIM e do PSM são especificados como metamodelos MOF
  - Define regras e ou algoritmos expressos em termos de todas as instâncias de tipos nos metamodelos das linguagens PIM e PSM

Abordagens de Transformação

- Transformação de Metamodelo

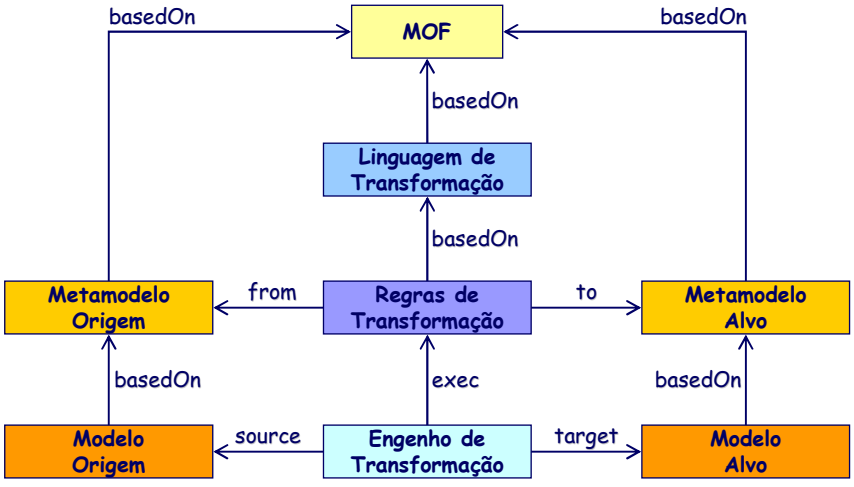


SBCARS 2007 Glêdson Elias

Abordagens de Transformação

Bezivin, 2004

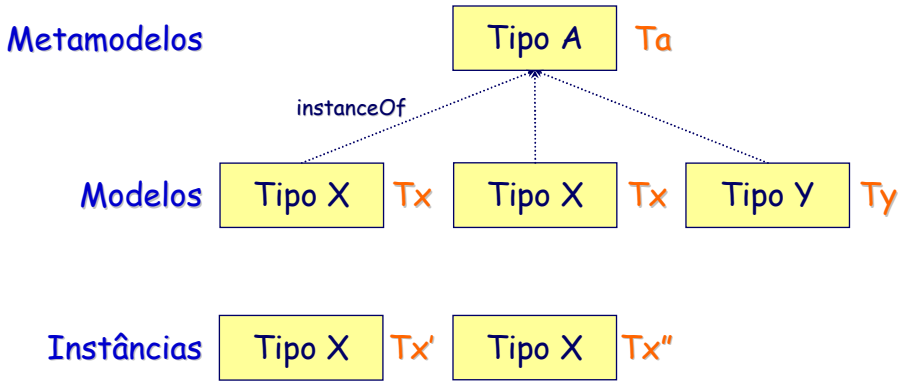
- Transformação de Metamodelo



SBCARS 2007 Glêdson Elias

Abordagens de Transformação

- Visão das Abordagens

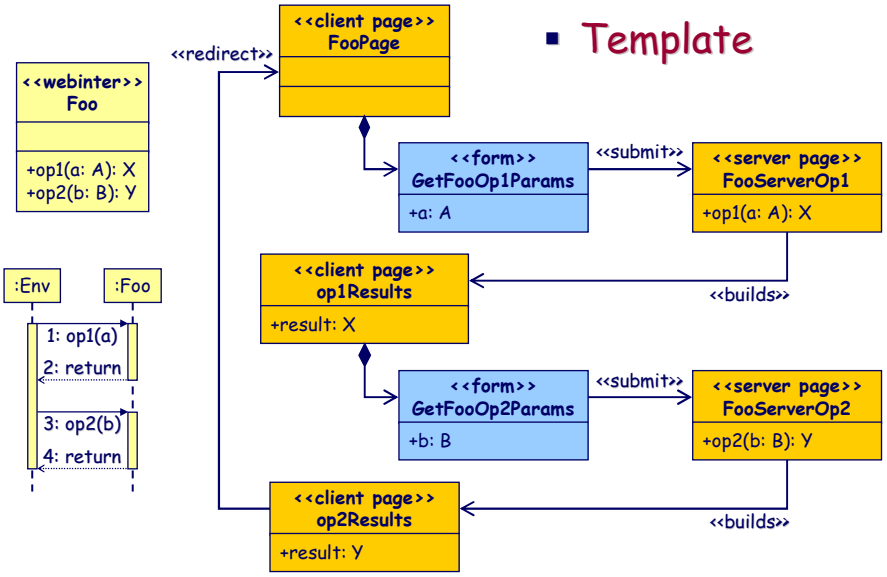


Abordagens de Transformação

- Template
  - É um modelo parametrizado que especifica um determinado tipo de transformação
  - É usado em regras para transformar um padrão de elementos de modelo em outro padrão (transformação de modelo)
  - Um conjunto de marcas pode ser associado ao template
    - Indica o template a ser aplicado
    - Associa valores aos parâmetros do template
    - Indica instâncias transformadas pelo template

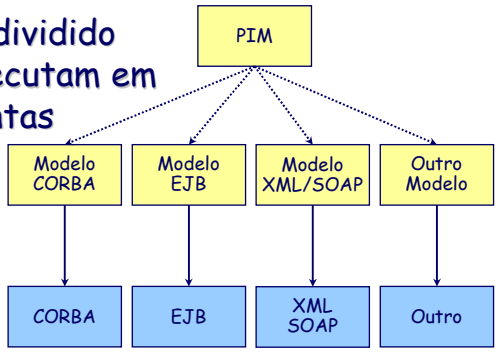
Abordagens de Transformação

- Template



Abordagens de Transformação

- Modelo Multiplataforma
  - PIM pode ser subdividido em partes que executam em plataformas distintas
  - Transformação pode usar marcas de diferentes modelos de plataforma

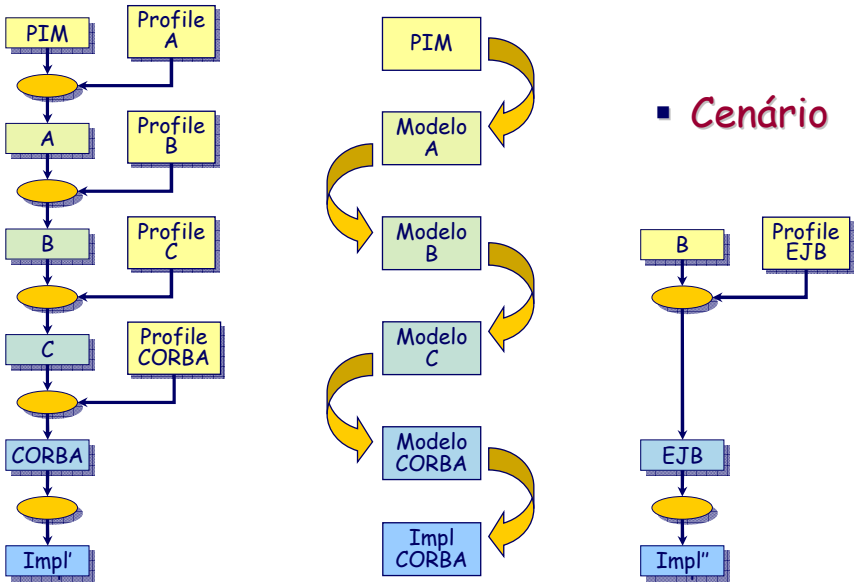


✓ Partes do sistema precisam se comunicar via IIOP, RMI e SOAP. Arquiteto escolhe o mecanismo mais adequado para cada conector, marcado-o com uma marca que identifica o mecanismo de comunicação a ser adotado

Múltiplas Transformações

- Características
  - A abordagem MDA pode ser aplicada múltiplas vezes sucessivas
  - PIM inicial é projetado para ser independente de escolhas de plataformas
  - PSM resultante de uma transformação é um PIM para a próxima transformação
  - Não é requerida a inclusão de todos os detalhes da plataforma no PSM

Múltiplas Transformações

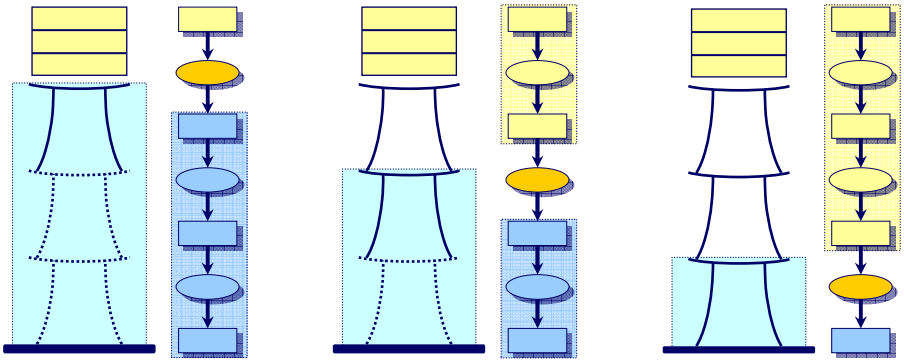


- Cenário

Múltiplas Transformações

- Visão da Plataforma

Implementação da Plataforma



Visão de interesse depende das necessidades do usuário do modelo

✓ Para muitos usuários MDA, o middleware é a plataforma
✓ Para o desenvolvedor de middleware, o sistema operacional é a plataforma

Reuso de Mapeamento

- Extensão
  - Adota um mapeamento base para criar um mapeamento derivado por modificação incremental
  - Modificações podem adicionar propriedades ou alterar propriedades do mapeamento base
  - Mapeamentos podem ser organizados em uma estrutura hierárquica com herança

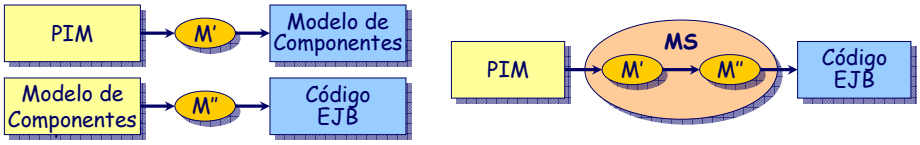
✓ Um mapeamento de diagramas de classe UML para modelos CORBA genéricos pode ser estendido para mapear para uma plataforma CORBA de um fabricante específico

Reuso de Mapeamento

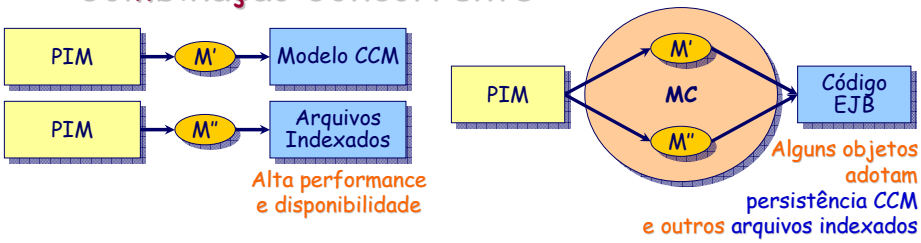
- Combinação
  - Usa dois ou mais mapeamentos para criar um novo mapeamento
  - Características do novo mapeamento são determinadas pelos mapeamentos combinados e pela maneira como são combinados
    - Efeito do novo mapeamento corresponde aos efeitos dos mapeamentos originais
  - Formas de Combinação
    - Seqüencial - aplica mapeamentos sucessivamente
    - Concorrente - aplica mapeamentos simultaneamente

Reuso de Mapeamento

- Combinação Seqüencial



- Combinação Concorrente

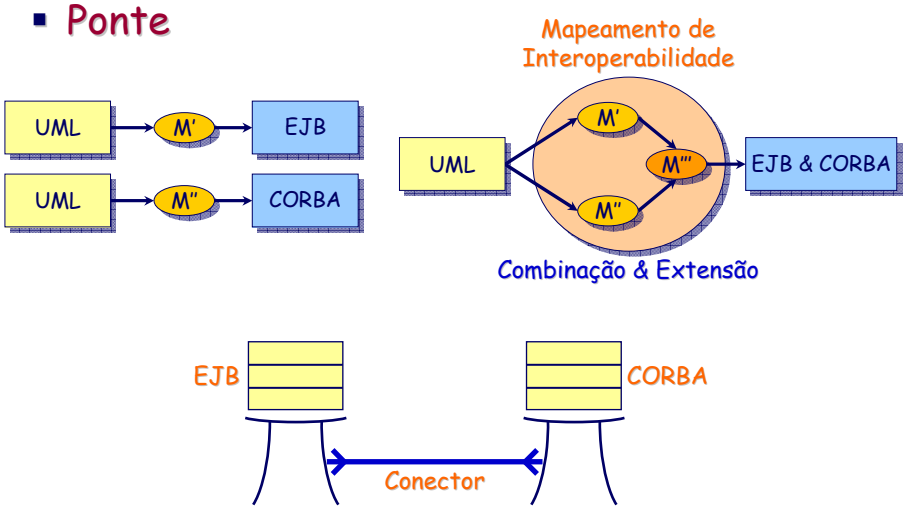


Reuso de Mapeamento

- Ponte
  - Transforma o PIM em um PSM multiplataforma e viabiliza a interoperabilidade dos subsistemas nas diferentes plataformas
  - Mapeamento de interoperabilidade é derivado dos mapeamentos das plataformas alvo
    - Combinação dos mapeamentos das plataformas mapeia alguns objetos para uma plataforma e outros para outra plataforma
    - Extensão do mapeamento combinado inclui conectores que definem as pontes entre as plataformas

Reuso de Mapeamento

- Ponte



Processo MDA

- Etapas
  - CIM é preparado mostrando o sistema no ambiente no qual opera
    - Auxilia a entender o que o sistema faz, independentemente de como é implementado
  - PIM é construído descrevendo o sistema
    - Não mostra detalhes de implementação
  - Arquiteto escolhe uma plataforma (ou várias) que viabilize a implementação com as qualidades arquiteturais desejadas

Processo MDA

- Etapas
  - Na transformação de instâncias . . .
    - PIM é marcado
    - PIM marcado é transformado em PSM
  - Na transformação de modelos & metamodelos
    - Engenho de transformação é usado para transformar o PIM em PSM
  - Processo pode ser aplicado iterativamente
    - PSM provê mais ou menos detalhes, dependendo do propósito
    - PSM pode ser o PIM da próxima iteração, até que uma implementação adequada é alcançada

Padrões

- Base Tecnológica
  - OMG adota tecnologias que conjuntamente viabilizam a abordagem dirigida a modelos
    - MOF (Meta Object Facility)
    - XMI (XML Model Interchange)
    - UML (Unified Modelling Language)
    - CWM (Common Warehouse Meta-model)
    - SPEM (Software Process Engineering Meta-model)
    - QVT (Query-View-Transformation)
    - Profiles UML
      - CORBA, CCM, EJB, EDOC, ...

Padrões

http://www.omg.org/technology/documents/formal/MOF_Core.htm

- MOF (Meta Object Facility)
  - É um framework de integração extensível dirigido a modelos para definir, manipular e integrar metadados e dados de uma forma independente de plataforma
  - Permite a definição de linguagens de modelagem
  - Padrões baseados em MOF são usados para integrar ferramentas, aplicações e dados
  - Adotado em diversos padrões OMG (XMI, UML, CWM, SPEM)

Padrões

http://www.omg.org/technology/documents/formal/xmi.htm

- XMI (XML Model Interchange)
  - É um framework de integração XML dirigido a modelos para definir, intercambiar e integrar objetos e dados XML
  - Facilita o intercâmbio de modelos via documentos XML, definindo regras para . . .
    - Derivar esquemas XML para metamodelos (linguagens de modelagem) baseados em MOF
    - Gerar documentos XML para modelos cujos metamodelos são baseados em MOF
  - XMI é usado para integrar ferramentas, repositórios, aplicações e dados
  - MOF/XMI compõem a espinha dorsal da infra-estrutura MDA

Padrões

http://www.omg.org/technology/documents/formal/uml.htm

- UML (Unified Modeling Language)
  - É uma linguagem visual para especificar, construir e documentar a estrutura e o comportamento de sistemas
  - É uma linguagem de propósito geral que pode ser usada . . .
    - Na maioria dos métodos baseados em componentes e objetos
    - Em todos os domínios de aplicação
    - Em todas as plataformas de implementação

Padrões

http://www.omg.org/technology/documents/formal/cwm.htm

- CWM (Common Warehouse Metamodel)
  - É uma especificação OMG para integração de repositórios de dados
  - Permite a modelagem de metadados para sistemas multidimensionais, relacionais e não-relacionais de data warehousing
  - Permite intercâmbio de metadados entre ferramentas de data warehousing e repositórios de metadados em ambientes heterogêneos distribuídos
  - Viabiliza mapeamentos de esquemas de bancos de dados

✓ CWM permite a modelagem de dados
✓ UML permite a modelagem de sistemas

Padrões

http://www.omg.org/technology/documents/formal/spem.htm

- SPEM (Software Process Engineering Meta-model)
  - É um metamodelo usado para descrever um processo de desenvolvimento de software ou uma família de processos relacionados
  - Provê uma linguagem para modelar e documentar processos de desenvolvimento de software, incluindo processos MDA
  - Define a colaboração entre entidades que realizam atividades sobre artefatos
  - É uma proposta neutra em relação a metodologias

Padrões

http://www.omg.org/cgi-bin/doc?ptc/2005-11-01

- QVT (Query-View-Transformation)
  - É um padrão para especificação de transformações de modelos
    - Viabiliza a transformação de um modelo em outro modelo
  - Transformações podem ser representadas na forma de modelos
  - Torna as ferramentas adaptáveis
    - Transformações podem ser incluídas, trocadas, removidas ou estendidas
    - Permite o intercâmbio de transformações entre diferentes ferramentas

Padrões

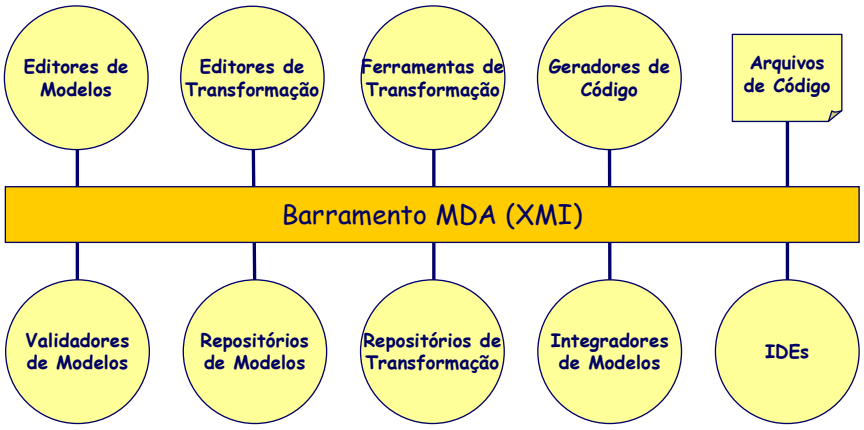
- QVT (Query-View-Transformation)
  - Query – é uma expressão aplicada ao modelo para identificar elementos de interesse
    - Resulta em instâncias de tipos definidos no modelo fonte ou na linguagem de consulta
  - View – é um modelo derivado a partir de um outro modelo
    - É gerada por transformações
  - Transformation – é um mapeamento de um modelo fonte para um modelo alvo

Padrões

- Profile UML
  - Representa um mecanismo de extensão da UML que a especializa para domínios específicos
  - Especifica uma nova linguagem de modelagem, adicionando e restringido elementos da linguagem fonte
  - Podem ser aplicados vários profiles a um modelo existente, estendendo ou restringindo elementos do modelo
  - Aplicação de um profile a um modelo pode ser removida

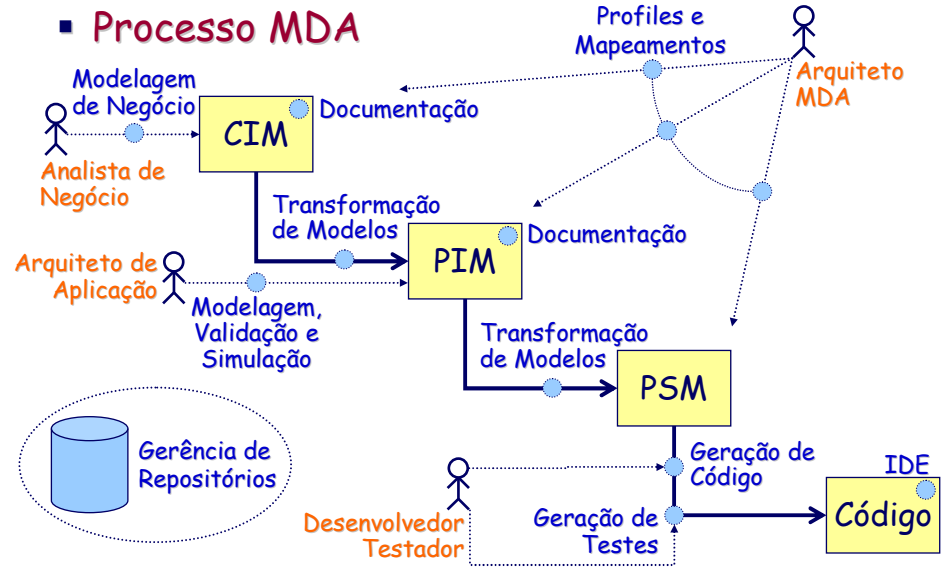
Ferramental

- Componentes MDA



Ferramental

- Processo MDA



Ferramental

- Linguagens de Transformação
  - MTL (Model Transformation Language)
  - ATL (ATLAS Transformation Language)
  - UMLX - linguagem gráfica
  - GReAT - linguagem baseada em grafos
- Ferramentas de Transformação
  - MDA Transf – regras expressas em XML
  - UMT (UML Model Transformation) – também é gerador de código de modelos UML/XMI
  - GMT (Generative Modeling Technologies) - projeto para construir ferramentas MDA
  - PathMATE - ferramentas para análise e engenho de transformação de modelos

Ferramental

- Geradores de Código
  - MIA (Model-in-Action) – suporta várias linguagens
  - OptimalJ - ferramenta MDA para J2EE
  - ArcStyler - ferramenta MDA para J2EE e .NET
  - Middlegen - engenho para banco de dados
- Frameworks
  - ModFact - projeto para MDA na LIP6
  - AndroMDA - arcabouço extensível de geração de código para Spring, EJB, .NET, Hibernate e Struts
  - OpenModel - geração de aplicações para UML
  - MCC – ferramental que suporta J2EE e .NET
  - Codagen - ferramental para J2EE e .NET
  - JAMDA - construção de geradores Java

Considerações Finais

- Realidade Atual
  - MDA parece estar no caminho certo
    - Conceitualmente claro e bem definido
    - Protege investimentos e propriedade intelectual separando os modelos de negócio e as tecnologias
  - MDA não é apenas uma visão
    - Muitos sistemas já são desenvolvidos usando MDA
  - MDA não é uma panacéia
    - Alguns céticos e críticos
    - Codificação automática é 100% possível e viável ?

Considerações Finais

- Realidade Atual
  - Mas ainda existe um longo caminho
    - Padronizar processos e linguagens, e disponibilizar ferramental que suportem de forma integrada todo o ciclo de vida de desenvolvimento de software e de MDA
    - Nenhum suporta o ciclo de vida completamente
  - Ferramental e padronização de processos e linguagens continuam em expansão
    - Muita pesquisa está sendo realizada
    - Ainda requer importantes avanços
  - Mercado ainda não é maduro

Considerações Finais

- Sucesso MDA
  - É preciso desmistificar a percepção que modelos são esforços extra e artefatos supérfluos/descartáveis
  - Redução da necessidade de codificação pode não ter aceitação dos desenvolvedores
    - Receio infundado de perder os empregos
    - Automação de transformações de modelos não representa incapacidade dos desenvolvedores
      - Torna os desenvolvedores mais eficientes
      - Evita trabalho repetitivo e não criativo

Considerações Finais

- Cenário Ideal
  - Papel dos Engenheiros de Software
    - Definir o processo MDA
    - Reusar modelos
    - Adaptar os modelos, caso necessário
    - Reusar modelos das plataformas
    - Descrever como os modelos são integrados
    - Marcar os modelos fonte
    - Gerar o sistema
  - Modelo torna-se um artefato (asset) reusável
  - Manutenção e evolução representam mudanças em modelos, marcações e mapeamentos

Considerações Finais

- Desafios
  - Suporte integrado ao ciclo de vida completo
    - Processos, linguagens, transformações, ferramental
  - Geração de código customizado e completo
  - Facilidades de sincronização, versionamento, simulação e validação
  - Modelos não representam adequadamente comportamento e coreografia (Moreno, 2004)
    - Maioria de modelos UML capturam apenas a estrutura
  - Adoção de padrões públicos x proprietários
    - Evitar a adoção de vários padrões complexos

Considerações Finais

- Desafios
  - Manutenção de código gerado (desconhecido) é mais cara e desencoraja o uso de modelos
    - Facilidades de depuração e reporte de erros devem ser incorporadas nas ferramentas
  - Código gerado é eficiente em termos de desempenho e tamanho?
    - Requer identificar domínios nos quais os modelos podem ser efetivamente usados e desenvolver ferramentas específicas

Considerações Finais

- Desafios
  - Como reusar componentes e sistemas legados?
    - MDA é uma abordagem top-down
    - Adequado ao desenvolvimento para reuso
    - Deve incorporar um visão bottom-up para desenvolvimento com reuso
  - Como manter e evoluir sistemas existentes?
    - Reengenharia?
    - Transformações reversas?
  - Facilidade de uso?
  - Custo de adoção, incluindo treinamento

Referências

- Anneke Kleppe, Jos Warmer, Wim Bast. "MDA Explained: The Model Driven Architecture, Practice and Promise" Addison-Wesley, 2003.
- Bran Selic. "The Pragmatics of Model-Driven Development." IEEE Software, Volume 20, Issue 5, pp. 19-25, September 2003.
- Ed Seidewitz. "What Models Mean" IEEE Software, Volume 20, Issue 5, pp. 26-32, September 2003.
- Jean Bézivin. "In Search of a Basic Principle for Model Driven Engineering." The European Journal for the Informatics Professional 5(2):21-24, 2004.
- Mary Shaw and David Garlan, "Software Architecture", Prentice Hall, 1996.

Referências

- OMG, MDA Guide Version 1.0.1, June 2003. <http://www.omg.org/docs/omg/03-06-01.pdf>
- OMG. "Meta Object Facility Core Specification, v2.0", January 2006. <http://www.omg.org/docs/formal/06-01-01.pdf>
- OMG. "XMI Mapping Specification, v2.1", September 2005. <http://www.omg.org/docs/formal/05-09-01.pdf>
- OMG. "Unified Modeling Language: Infrastructure", Version 2.1.1, February 2007. <http://www.omg.org/docs/formal/07-02-04.pdf>
- OMG, "Common Warehouse Metamodel Specification", Version 1.1, March 2003. <http://www.omg.org/docs/formal/03-03-02.pdf>

http://www.omg.org/cgi-bin/doc?ptc/2005-11-01

Referências

- OMG, "Software Process Engineering Metamodel Specification", Version 1.1, January 2005. <http://www.omg.org/docs/formal/05-01-06.pdf>
- OMG, "MOF QVT Specification", November 2005. <http://www.omg.org/docs/ptc/05-11-01.pdf>
- Open Distributed Processing Reference Model (RM-ODP), ISO/IEC IS 10746 / ITU-T X.900. <http://www.joaquin.net/ODP>.