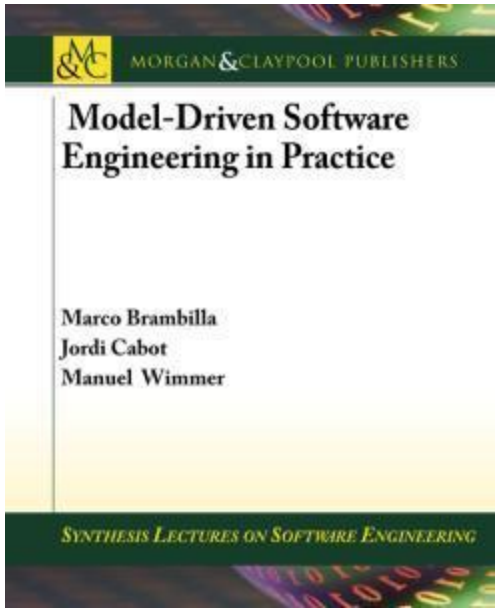
MORGAN & CLAYPOOL PUBLISHERS

Capítulo - 10

GERENCIANDO MODELOS

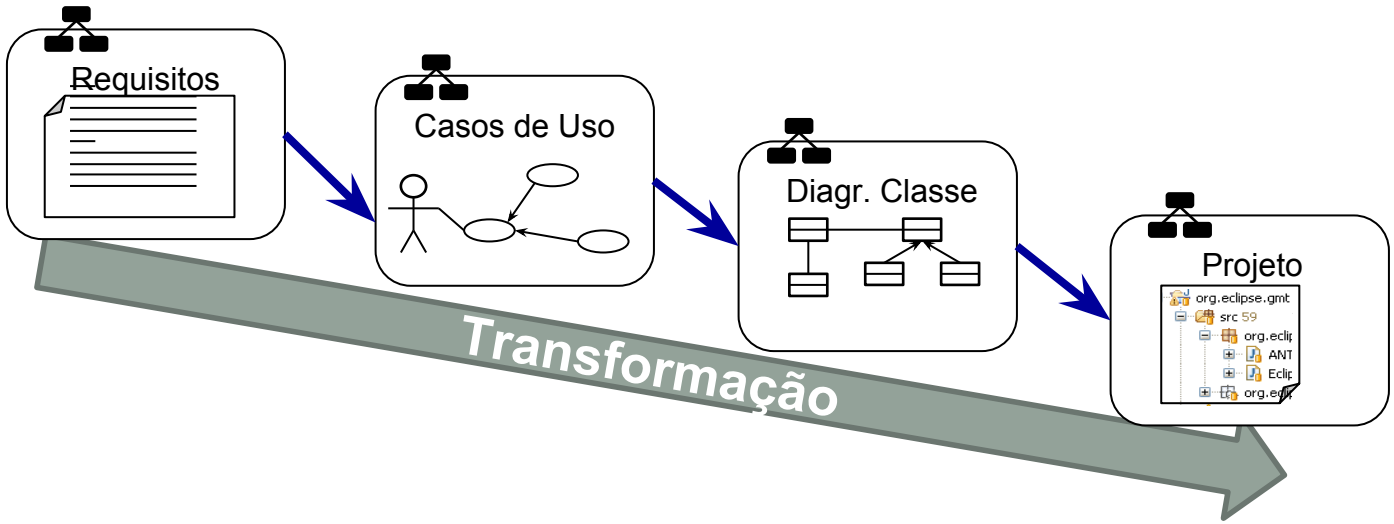


Model-Driven Software Engineering in Practice
by Marco Brambilla, Jordi Cabot, Manuel Wimmer.
Morgan & Claypool, USA, 2012.

Motivação

Porque Gerenciar Modelos?

- Em Engenharia Dirigida por Modelos (MDE) tudo são considerados modelos mas o mais importante é que um modelo não é uma ilha isolada.
- Todos os artefatos de modelagem em um projeto MDE estão inter-relacionados. Estes relacionamentos devem ser corretamente gerenciados durante o ciclo de vida do projeto.



Conteúdo

1. Modelo de Intercâmbio
2. Modelo de Persistência
3. Modelo de Comparação
4. Modelo de Versionamento

RONALDO

5. Modelo de Co-Evolução
6. Gerenciamento Global dos Modelos
7. Modelo de Qualidade
8. Modelagem Colaborativa

VALDEMAR

Implementação dos Modelos usando o Eclipse

THIAGO

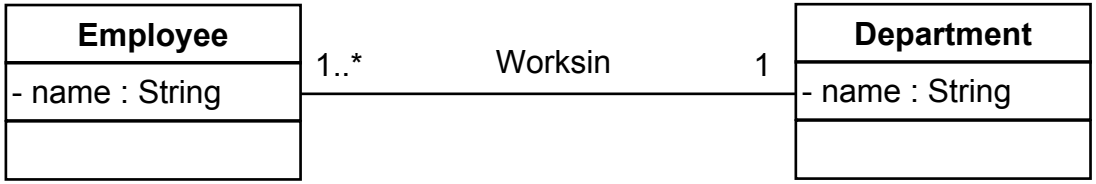
Modelo de Intercâmbio

Modele Uma Vez, Abra em Qualquer Lugar

- Existe uma necessidade clara da capacidade de diferentes ferramentas reconhecerem um mesmo modelo
  - Em outras palavras, pode-se escolher:
    - Ferramenta **A** para especificação do modelo
    - Ferramenta **B** para verificação da qualidade do modelo
    - Ferramenta **C** para executar o modelo...
- Ainda estamos muito longe deste objetivo – FALTA DE PADRÕES
- Tentativa de Solução: XMI (XML Metadata Interchange), um padrão adotado pela OMG para serialização e troca de modelos UML e MOF
- O problema é que cada ferramenta parece entender este padrão de maneiras diferentes

Exemplo XMI

(Versões parciais e simplificadas dos arquivos reais XMI)

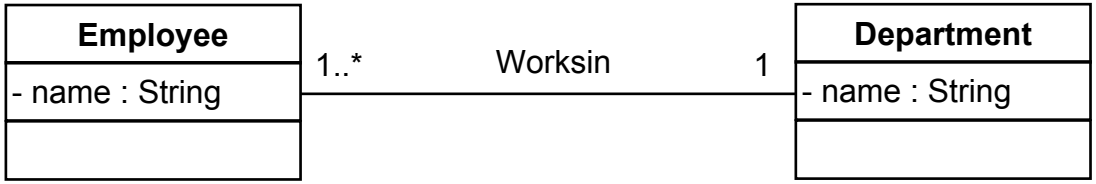


XMI do ECLIPSE ≠ **XMI do ArgoUML** ≠ **XMI do StarUML** ≠ - - -

Exemplo XMI

(Versões parciais e simplificadas dos arquivos reais XMI)

ECLIPSE

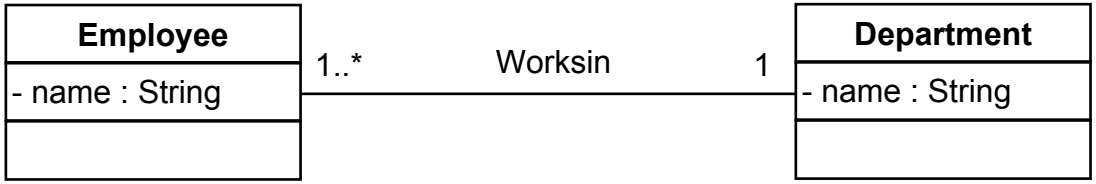



```
<packagedElement xmi : type="uml : Class " xmi : id=" c001 " name="Employee">
      <ownedAttribute xmi : id=" a001 " name="name"/>
</packagedElement>
<packagedElement xmi : type="uml : PrimitiveType " xmi : id=“t001" name="String "/>
<packagedElement xmi : type="uml : Class " xmi : id=" c002 " name="Department">
      <ownedAttribute xmi : id=“a002 " name="name" type=“t001"/>
</packagedElement>
<packagedElement xmi : type="uml : Association " xmi : id=“as001" name="WorksIn"
memberEnd=" e001 e002">
      <ownedEnd xmi : id=“e001" type=“c002 " association=“as001"/>
      <ownedEnd xmi : id=“e002" name=" type=“c001" association=“as001">
      <upperValue xmi : type="uml : LiteralUnlimitedNatural " xmi : id="un001" value=""/>
      </ownedEnd>
</packagedElement>
```

Exemplo XMI

(Versões parciais e simplificadas dos arquivos reais XMI)

ArgoUML

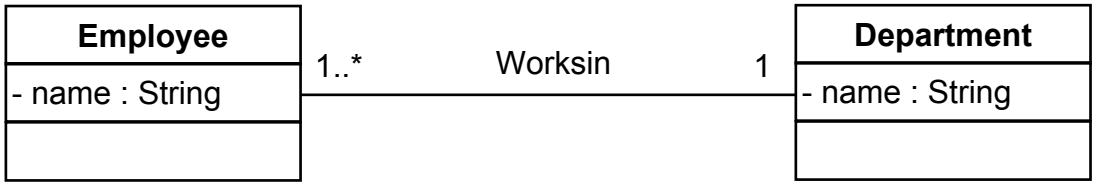



```
<UML: Classxmi.id = ‘c001 '
      name = 'Employee ' visibility = ‘public ' isSpecification = ' false ' isRoot = ‘false '
      isLeaf = ‘false' isAbstract = ‘false' isActive = ‘false'>
      <UML: Classifier.featur e>
      <UML: Attributexmi.id = ‘a001 '
            name = 'name' visibility = ‘public ' isSpecification = ‘false’
            ownerScope = ‘instance' changeability = ‘changeable ' targetScope = ‘instance '>
      <UML: StructuralFeature.multiplicity >
            <UML: Multiplicityxmi.id = ‘m001'>
                  <UML: Multiplicity.range>
                  <UML: MultiplicityRangexmi.id = ‘mr001 ‘ lower = '1 ' upper = ‘1'/>
                  </UML: Multiplicity.range>
            </UML: Multiplicity>
      </UML: StructuralFeature.multiplicity >
</UML: Class>
```

Exemplo XMI

(Versões parciais e simplificadas dos arquivos reais XMI)

StarUML




```
<XPD:ATTR name="#OwnedElements" type="integer">3</XPD:ATTR>
<XPD:OBJ name="OwnedElements[0]" type="UMLClass" guid="Au9Z38kjrkqA+1JT5/3omQAA">
<XPD:ATTR name="Name" type="string">Employee</XPD:ATTR>
...
<XPD:ATTR name="Name" type="string">name</XPD:ATTR>
<XPD:ATTR name="Visibility" type="UMLVisibilityKind">vkPrivate</XPD:ATTR>
<XPD:ATTR name="TypeExpression" type="string">String</XPD:ATTR>
<XPD:REF name="Owner">Au9Z38kjrkqA+1JT5/3omQAA</XPD:REF>
</XPD:OBJ>
<XPD:OBJ name="OwnedElements[1]" type="UMLClass" guid="yNZA6VEYQEucgL5SWa834wAA">
<XPD:ATTR name="Name" type="string">Department</XPD:ATTR>
...
<XPD:ATTR name="Name" type="string">name</XPD:ATTR>
<XPD:ATTR name="Visibility" type="UMLVisibilityKind">vkPrivate</XPD:ATTR>
<XPD:ATTR name="TypeExpression" type="string">String</XPD:ATTR>
<XPD:REF name="Owner">yNZA6VEYQEucgL5SWa834wAA</XPD:REF>
</XPD:OBJ>
<XPD:OBJ name="OwnedElements[2]" type="UMLAssociation" guid="DbTzeqHRIUq2NQRvv6pZeQAA">
<XPD:ATTR name="Name" type="string">WorksIn</XPD:ATTR>
...
```

Avanços Recentes

- **Model Interchange Working Group (MIWG)**
  - Tem o objetivo de permitir a avaliação da capacidade de intercâmbio do modelo das ferramentas de modelagem pela comparação dos arquivos XMI dos fornecedores com um conjunto de testes previamente definidos.
  - Casos de teste (total de 20) para demonstrar o intercâmbio entre **UML**, **SysML**, **SoaML** and **UPDM** (*Unified* Profile for DoDAF and MODAF)
    - **DoDAF** - Department of Defense Architecture Framework
    - **MODAF** - UK Ministry of Defence Architecture Framework

- Os novos padrões de definição de diagramas vão permitir não somente o intercâmbio dos conteúdos de modelagem como também a disposição gráfica dos modelos.

Avanços Recentes

- **Model Interchange Working Group (MIWG)**
  - Principais fornecedores participantes:

Vendor	Point of Contact	Tool	Version
Atego	Simon Moore	Artisan® Studio	7.2m
IBM	Maged Elaasar	RSx	8.0.3
IBM/Sodius	Eldad Palachi/Mickael Albert	IBM Rhapsody	7.6.x
No Magic	Lonnie VanZandt	MagicDraw	17.0
SOFTEAM	Etienne Brosse	Modelio	2.4.19
Sparx Systems	J. D. Baker	Enterprise Architect	9.1

  - Outros participantes:
    - Roy Bell, Raytheon (MIWG Co-Chair)
    - Peter Denno, NIST (Validator)
    - Sandy Friedenthal, SAF Consulting (MIWG Chair)
    - Leonard Levine, DISA
    - Pete Rivett, Adaptive
    - Ed Seidewitz, Model Driven Solutions

Modelo de Persistência

Modelo Persistência

- Normalmente os modelos são serializados em arquivos simples, seguindo o formato XMI ou outro formato XML proprietário.

- Não funcionam muito bem com grandes modelos. Problemas de escalabilidade
  - A carga de todo o modelo na memória pode não ser uma boa opção
  - Estratégias de acesso aleatório mais carregamento sob demanda são necessários

Modelo Persistência

Alternativas

- Repositório modelos CDO (Connected Data Objects – Objetos de Dados Conectados)
  - Framework de persistência em tempo real otimizado para consultas escaláveis e suporte transacional para grandes objetos gráficos
  - Back-ends: objetos, NoSQL e base de dados relacionais
  - Para base de dados relacional, CDO depende do Teneo6, um modelo relacional de mapeamento de presistencia da base de dados em tempo real.

- Soluções NoSQL puras: Morsa e MongoEMF. Os dois usam o MongoDB como backend.

- Novas alternativas objetivam a utilização da computação nas nuvens (Cloud) como solução para o modelo de armazenamento.

MODELO DE COMPARAÇÃO

Modelo de Comparação

- A comparação de dois modelos é uma operação fundamental em muitas operações de gerenciamento de modelos tais como modelos de versionamento.

- O ponto inicial da comparação de modelos é a identificação do conjunto de diferenças entre os dois modelos

- Estas diferenças são normalmente representadas como um novo modelo , chamado de modelo de diferenças.

Modelo de Comparação: Modelo de Correspondência

Fase 1 do processo de comparação de modelos

- Identificar os elementos comuns dos dois modelos
- Como deve ser definido quais elementos possuem a mesma identidade?
  - Identidade estática: id's explícitos identificando os elementos
  - Identidade de assinatura: identidade baseada nas características dos elementos dos modelo (nome, elementos próprios,…)
- A identidade pode ser uma função probabilística (correspondência por similaridade)
- Funciona melhor se os usuários redefinirem o conceito de correspondência para DSLs específicas (de modo que a sua semântica específica possa ser levado em conta)

Modelo de Comparação: Modelo de Diferenciação

Fase 2 do processo de comparação de modelos

- Os elementos correspondentes são procurados pelas diferenças
- As diferenças correspondem a uma operação atômica (**add** / **delete** / **update** / **move**) executada em um destes elementos
- Estas diferenças são coletadas e armazenadas em diferentes modelos

Ferramentas de Modelo de Comparação

- **EMF compare**:
  - A mais popular de todas
  - Facilidade de comparação genérica para qualquer tipo de modelo EMF
  - As diferenças podem ser exportadas como um modelo de correções

- **SiDiff**:
  - Principalmente trata da correspondência baseada em similaridade
  - Adaptável para qualquer modelo no padrão gráfico

- **Epsilon Comparison Language**:
  - Incluem uma DSL para realizar as implementações das mudanças especializadas de alto nível
  - Com ela, mudanças de alto nível tais como refatoração podem ser igualmente detectadas

Modelo de Versionamento

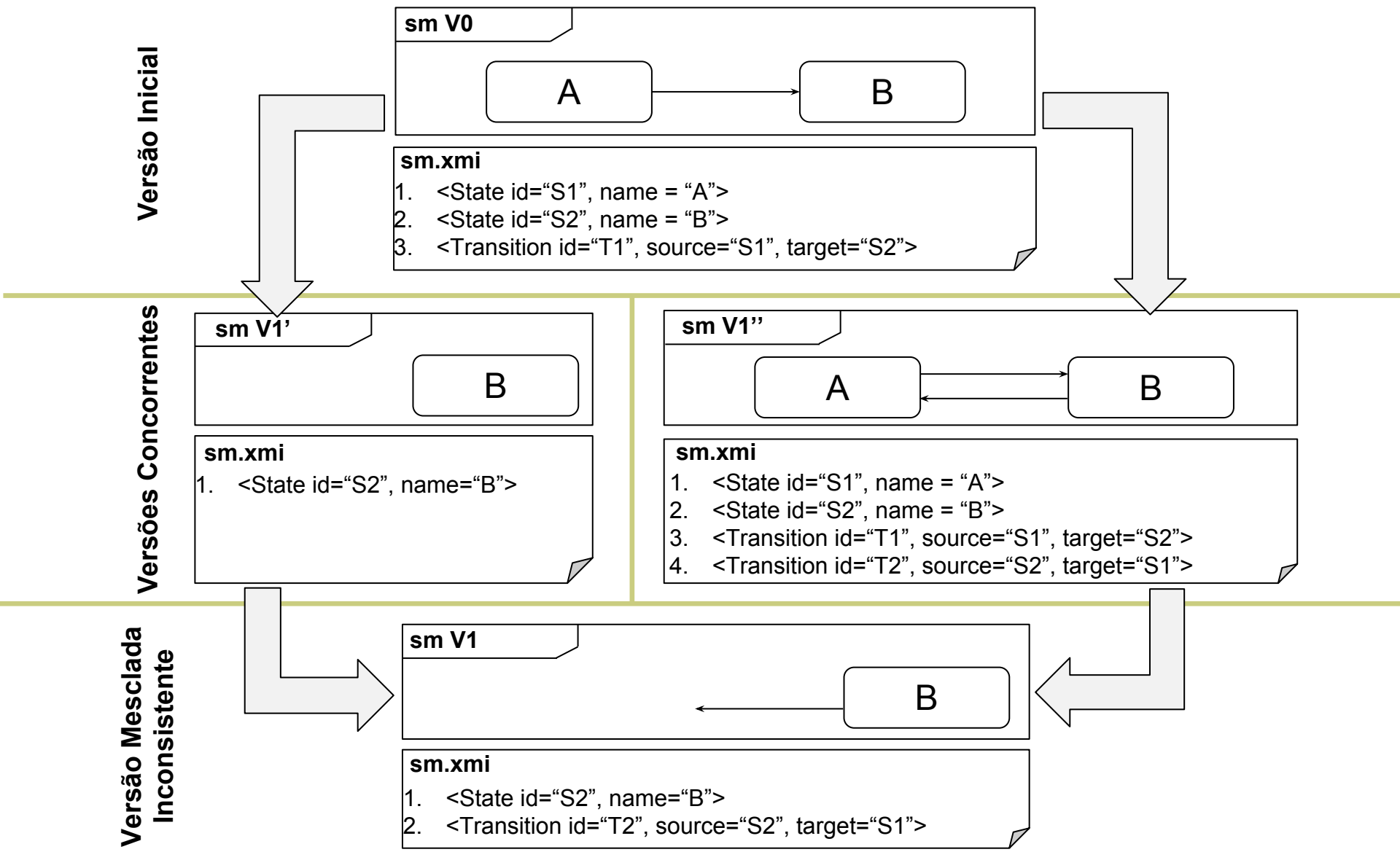
Modelo de Versionamento

- Programadores não vivem sem sistemas de controle de versionamento tais como SVN ou GIT. Os projetistas também precisam destes sistemas de controles de versionamento (VCS - Version Control System) para trabalhar com modelos .

- Os VCSs ajudam a detectar, gerenciar e resolver conflitos que surgem ao mesclar modelos

- VCSs atuais são baseados em texto. Usá-los para mesclar modelos podem resultar em resultados inconsistentes devido à semântica baseados em grafos dos modelos.

Modelo de Versionamento



Modelo de Versionamento

Ferramentas

- São necessários VCSs baseado em modelos dedicados

- Algumas primeiras tentativas:
  - **EMFStore**: Projeto o ficial do Eclipse para repositório de modelos. Segue os mesmos protocolos de interações do SVN no nível de modelos.
  - **AMOR** (Adaptable model versioning): Detecção de conflitos graves e possibilidade de estratégias de resolução.Processo de mesclagem visual por meio de anotações de conflitos diretamente no ambiente gráfico dos modelos
  - **CDO:** inclui suporte de ramificação para modelos
  - **Epsilon Merging Language:** é uma linguagem baseada em regras para mesclagem (heterogênea) de modelos

- Versionamento do layout gráfico ainda é um problema
  - Mover uma classe duas polegadas para a direita conta como uma mudança?

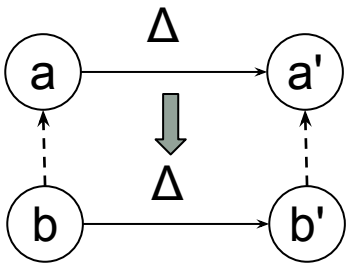
Co-EVOLUção de Modelos

Co-Evolução de Modelos

Ferramentas

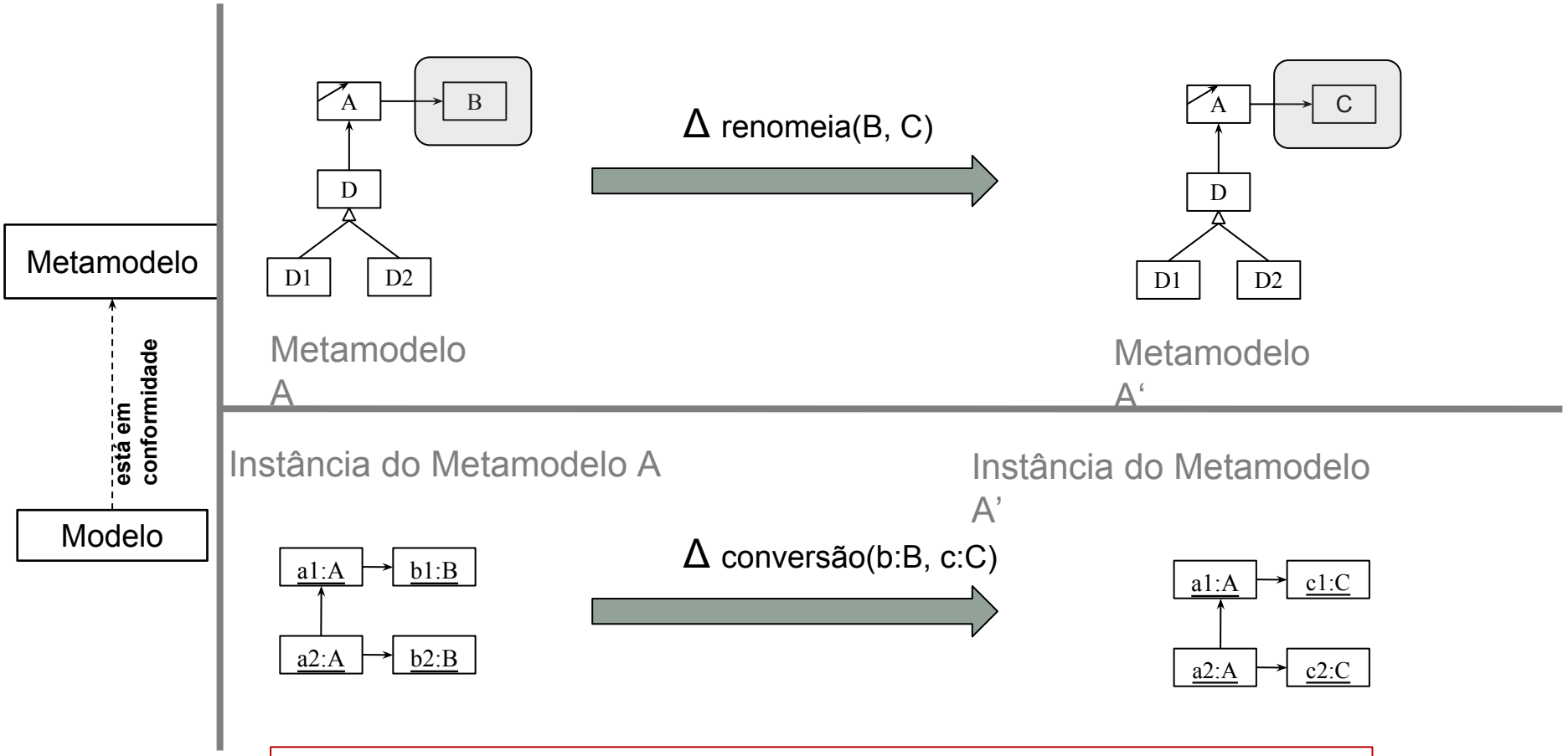
- Versionamento de Modelos trata das mudanças em um único artefato mas cada mudança pode afetar muitos outros artefatos relacionados.

- Co-Evolução em MDE
  - Co-evolução é a **mudança** de um modelo **disparada** por uma **mudança** no modelo **relacionado.**
  - Visão Atual
    - Relação: r(a,b)
    - a → a'
    - b → b' | r(a',b')
    - **Desafio: Reconciliação da Relação**
  - Pesquisa Atual foca em relações um-para-um:
    - Evolução Modelo / Metamodelo
    - Evolução Metamodelo/ Transformação
    - …



Co-Evolução de Modelos e Metamodelos

Exemplo



Suposição: Classes renomeadas não representam um novo conceito!

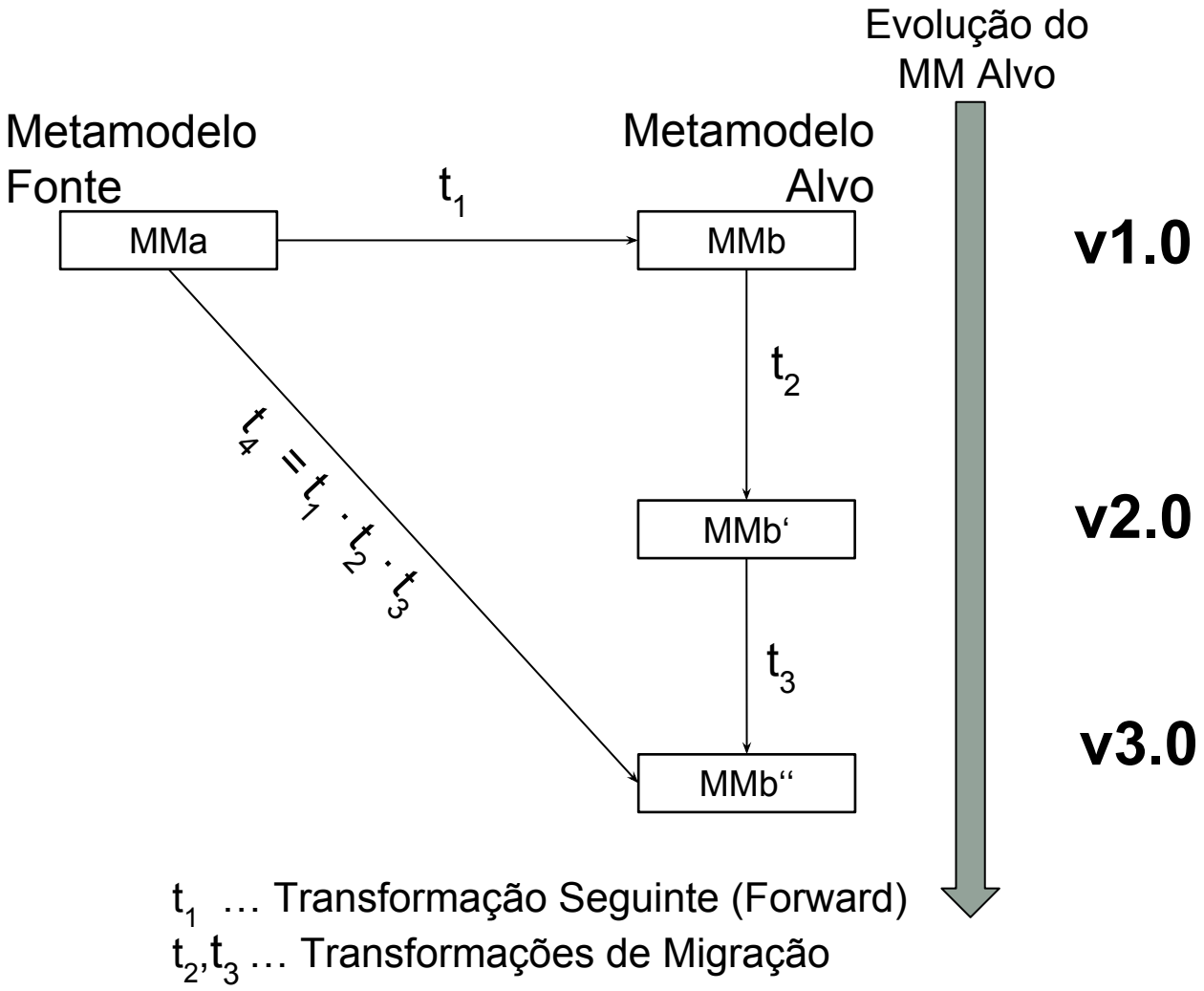
Co-Evolução Modelo / Metamodelo

Processo

- Classificação de mudanças no metamodelo
  - Operações Sem quebra: não exige migrar os modelos
  - Quebra e Soluciona: Migração automática de modelos existentes é possível.
  - Quebra e Não-Soluciona (automaticamente): Intervenção do Usuário é necessária
- Ferramentas como Edapt e Epsilon Flock podem derivar uma transformação de migração para adaptar modelos correntes para uma nova estrutura de metamodelo quando possível.

Coevolução Metamodelo/Transformação

Outros Cenários de Coevolução



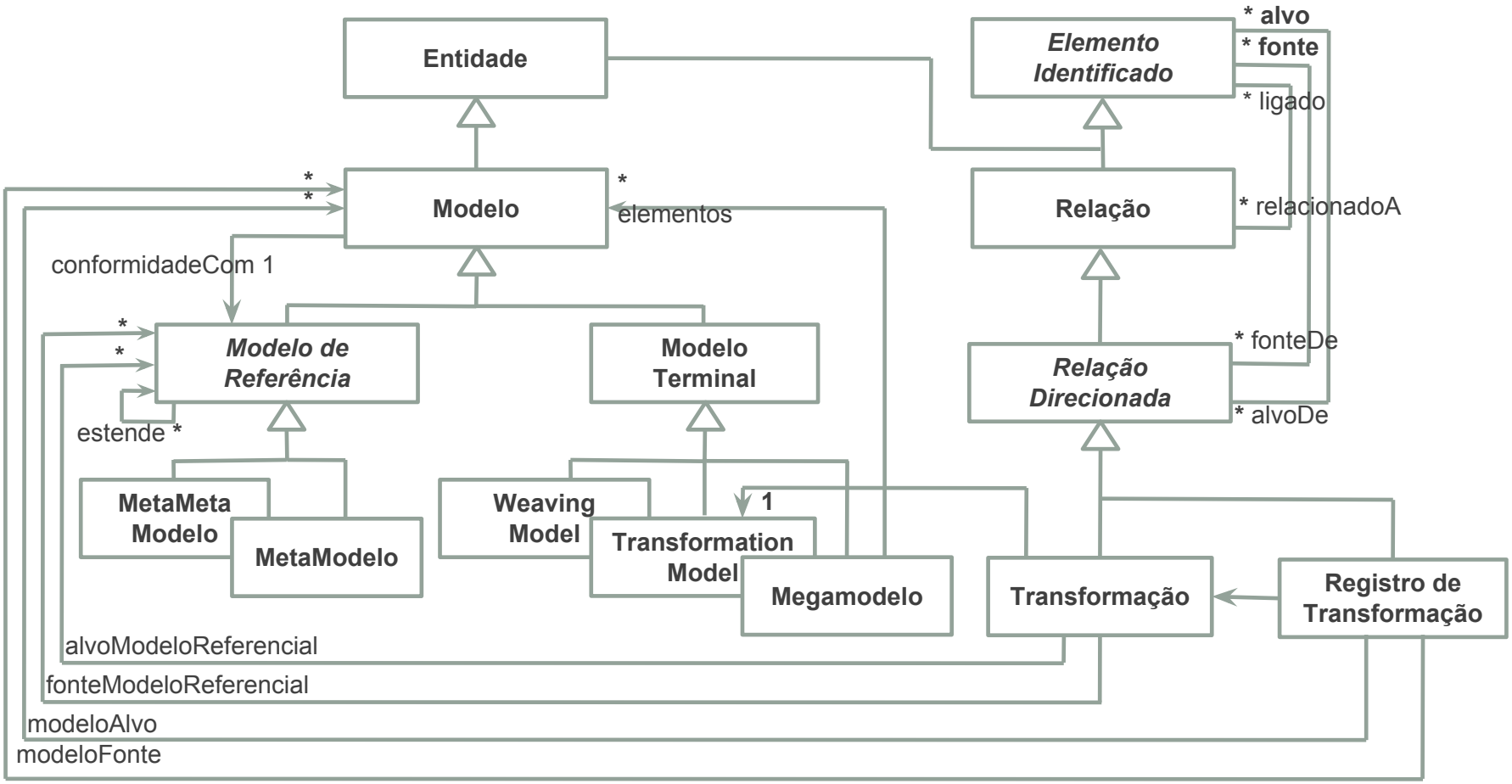
Gerência de Modelos Global

Gerência de Modelos Global

- Solução baseada em modelos para o problema de gerenciar todo o ecossistema de modelos que aparece em um projeto MDE;
- Representa-se com um modelo, *o megamodelo*, todos os modelos (e artefatos relacionados como arquivos de configuração) e relações no ecossistema;
- Um megamodelo pode ser visto como um repositório de metadados para o projeto;
- Um megamodelo é um modelo cujos elementos são de fato outros modelos;
- Como um modelo, um megamodelo pode ser diretamente manipulado usando as mesmas ferramentas empregadas para manipular modelos normais.

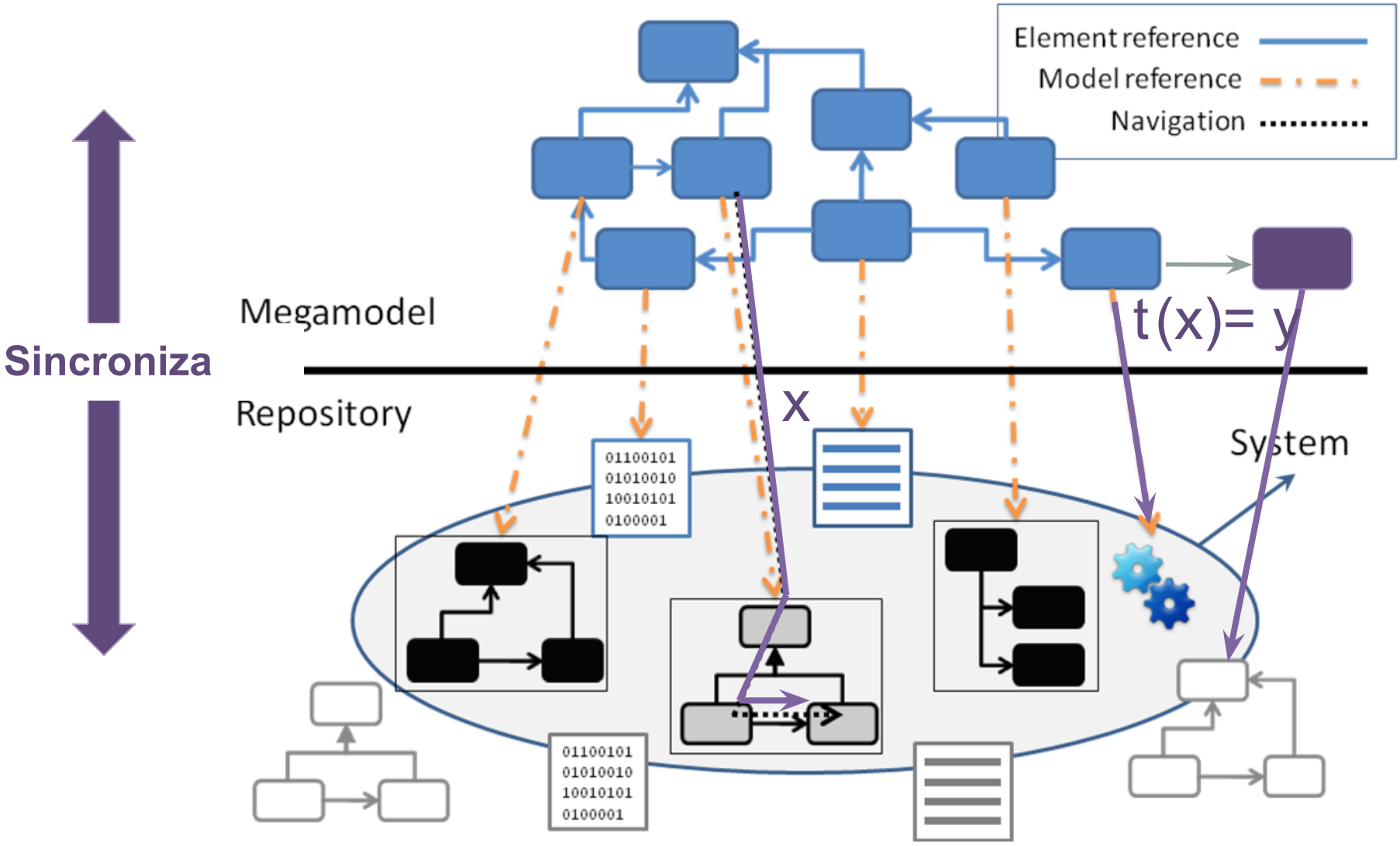
Gerência de Modelos Global

O metamodelo de um megamodelo



Gerência Global de Modelos

Usando megamodelos



Gerência de Modelo Global

Repositório de Artefatos de MDE

- Dissertação de Mestrado
  - http://www.inf.ufg.br/mestrado/sites/www.inf.ufg.br.mestrado/files/uploads/Dissertacoes/dissertacao_valdemar_.pdf
  - Já constava a ideia de Repositório de Regras (generalizando: artefatos)

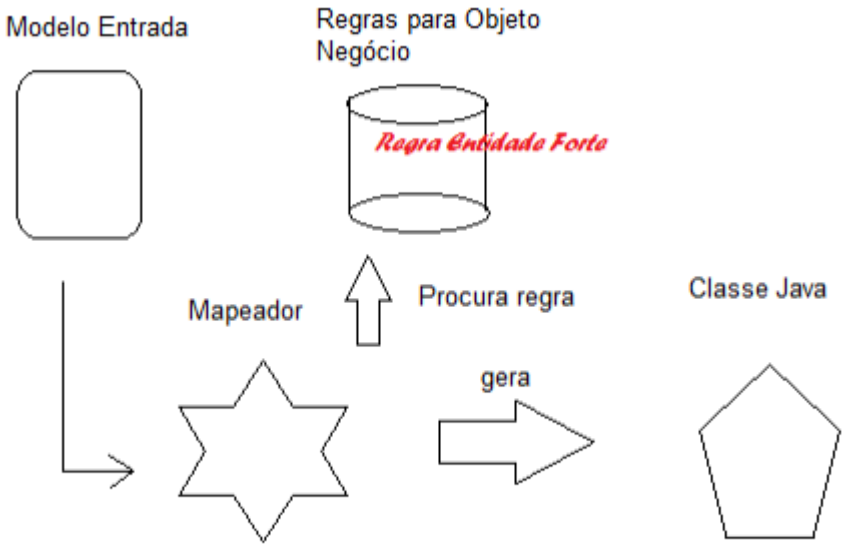


Ilustração do processo de mapeamento.

Gerência de Modelo Global

MoScript

- DSL para escrever scripts de gerência de modelos sobre megamodelos
- Permite a automação de tarefas de modelagem complexas, envolvendo várias manipulações consecutivas sobre um conjunto de modelos.

Gerência de Modelos Global

Exemplo de MoScript

- Operações de Consulta

```
Model::allInstances()->any(m | m.indentifier ='SimpsonFamily')
->allContents()->collect(el | el.name))
```

Collection {'Bart', 'Homer', 'Lisa', 'Maggie', 'Marge'}

- Transformações Modelo Para Modelo (M2M)

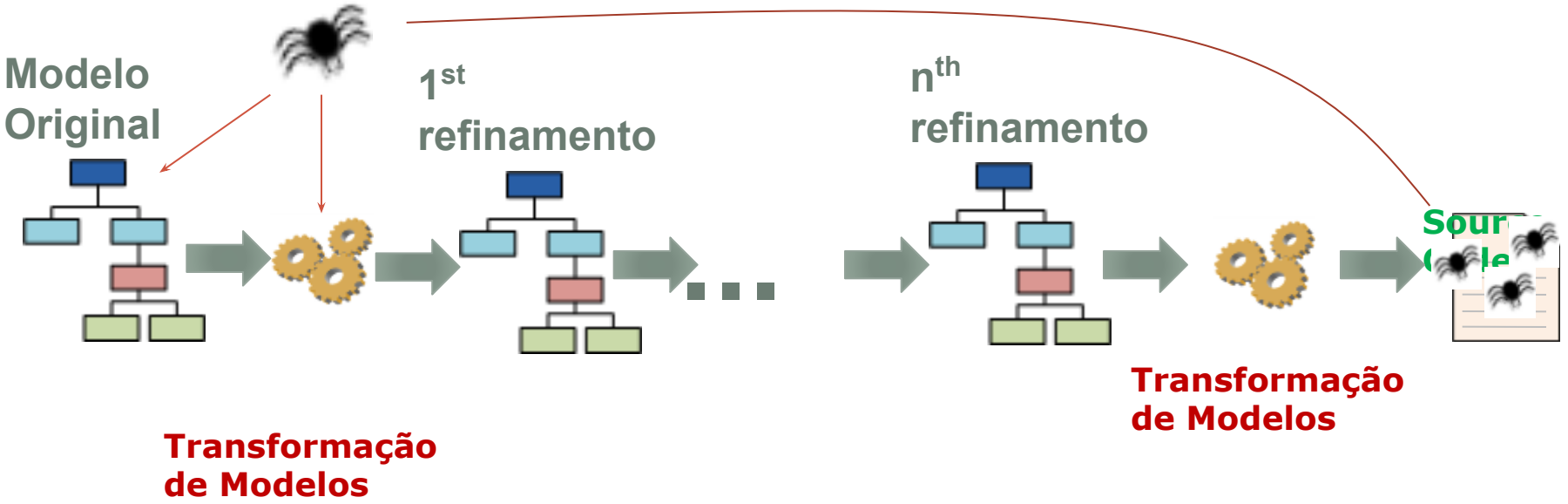
```
1  let j2dNet : Transformation = Transformation::allInstances()
2      ->any(t | t.identifier ='j2dNet')
3  in
4
5  Model::allInstances()
6      ->select(m | m.conformsTo.kind ='Java'))
7      ->collect (jModel | j2dNet.applyTo(jModel))
```

```
TransformationRecord::allInstances()->collect(tr | tr.run())
```

Qualidade de Modelo

Motivação

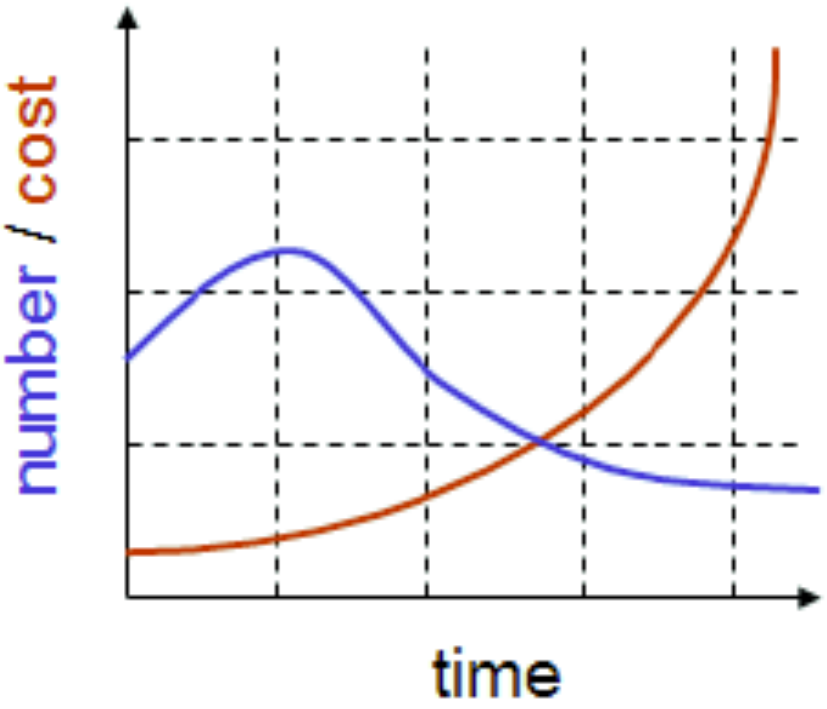
Processo de Desenvolvimento de Software baseado em MDE



Erros no modelo levam a erros no software resultante.

Qualidade de Modelo

Processo baseado em MDE



Qualidade de Modelo

- Ferramentas de Modelagem só checam a boa formação (*well-formedness*)
  - Um modelo está em conformidade com seu metamodelo? i.e. o modelo é uma instância válida do metamodelo?

- Mas esta é apenas a ponta do *iceberg* quando inicia-se a avaliação da qualidade de um modelo. Há muitas propriedades para verificar:
  - em modelos estáticos: satisfatibilidade, vivacidade, redundância, consistência,*subsunção (subsumption - acrescentar algo a um contexto)* …
  - em modelos dinâmicos: ausência de impasses (deadlocks), alcançabilidade, …
- A verificação pode ser feita através do uso de métodos formais ou teste.

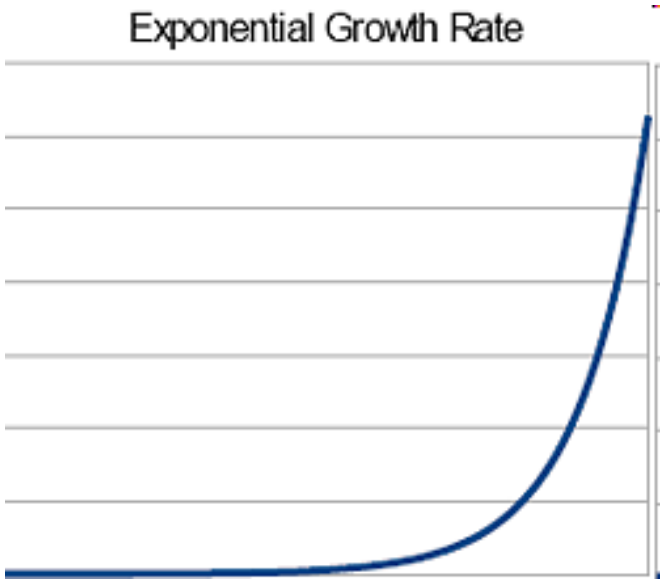
Qualidade de Modelo

- Outro tipo de consistência citada no livro: consistência entre visões arquiteturais

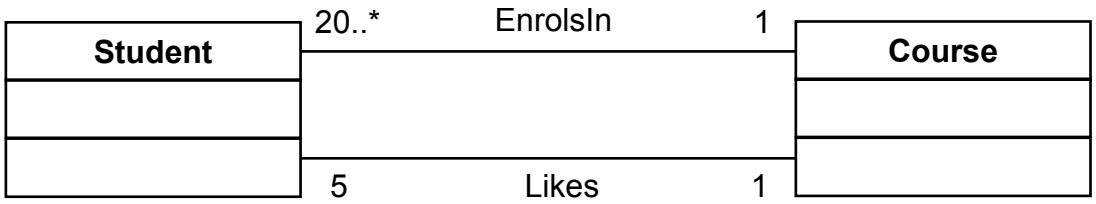
Exemplo de propriedade: satisfatibilidade

- Um modelo é satisfatível se é possível criar uma instância válida daquele modelo. Uma instanciação é válida se satisfaz todas as restrições do modelo.
- Mais difícil do que parece.

MDE

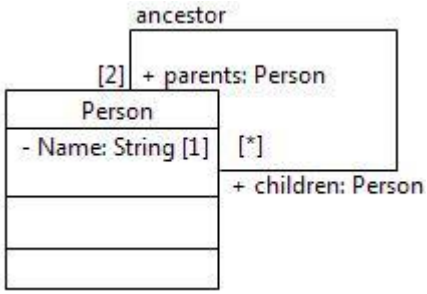


Exemplo de não satisfatibilidade (1)

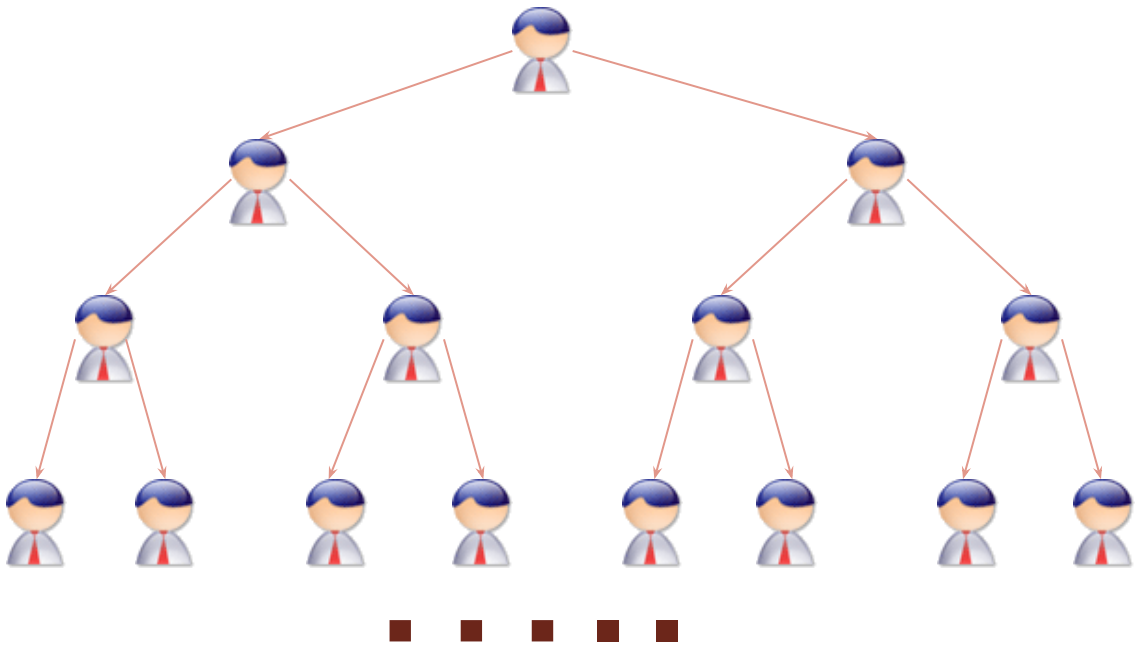


- Devido a relação matriculaEm |student|>=20*|course|
- Devido aos *Likes* |student|=5*|course|

Exemplo de problema não satisfatível (2)

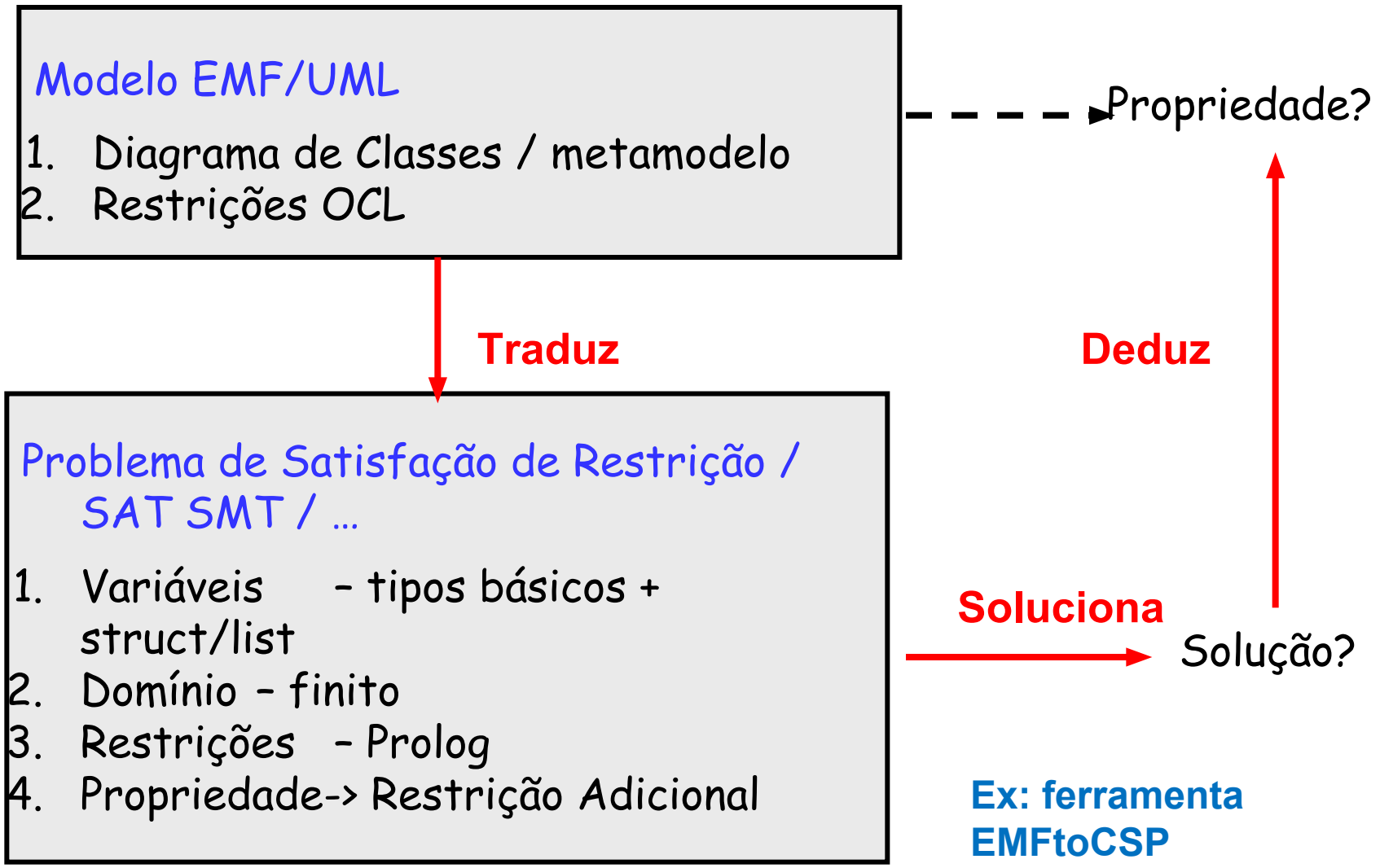


e nenhuma pessoa é
seu próprio ancestral



Satisfatibilidade forte

Abordagem de verificação formal típica



Testando modelos

Derive testes de seus modelos

- Da mesma forma que se pode testar código, modelos podem também ser testados
  - Ferramentas como USE podem criar fotografias de um sistema e avaliar restrições OCL sobre elas para testar expressões OCL
- Operações de Transformações de modelo & operações do tipo transformação de modelos
  - E.g. nós podemos querer checar se uma transformação gera o modelo de saída válido toda vez que um modelo de entrada válido é provido (VALIDAÇÃO DA TRANSFORMAÇÃO)
- Várias técnicas caixa-preta e caixa-branca para testes de modelos tem sido propostas

Modelagem Colaborativa

Modelagem Colaborativa

- Modelagem é por definição uma atividade de equipe
- Sincronização Offline de modelos podem ser manipuladas usando ferramentas de versionamento vistas anteriormente
- Modelagem colaborativa Online (vários usuários atualizando o mesmo modelo ao mesmo tempo) é muito problemático
  - Baseado em um modelo de transação pequeno onde mudanças podem ser imediatamente propagadas para todo mundo
  - Mecanismos de gerenciamento de conflito muito leves(e.g. tranca voluntária)
  - Resolução de conflitos através de consenso explícito entre todas as partes

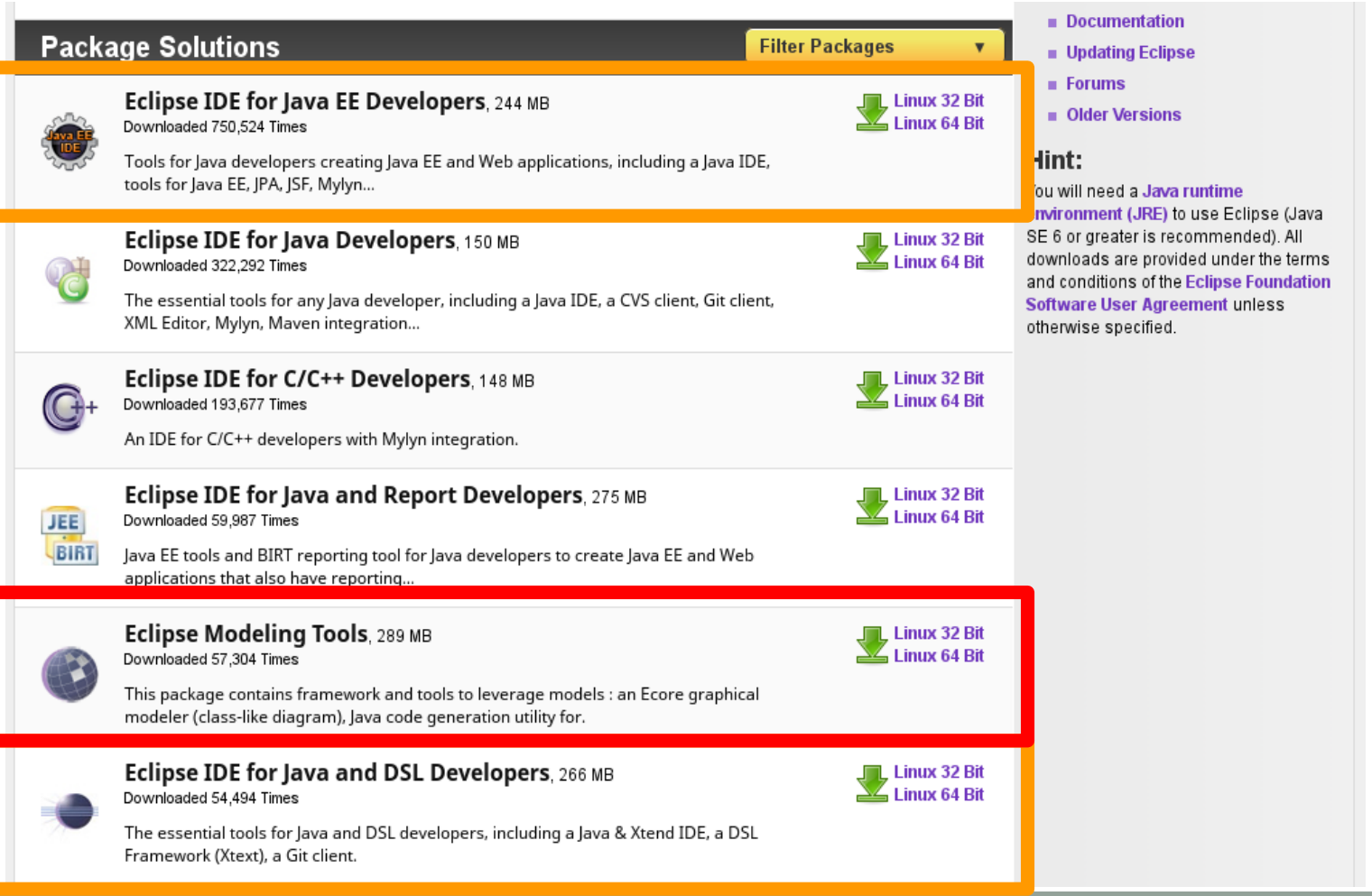
Modelagem Colaborativa

Ferramentas

- EMFCollab
  - Cópia mestre no servidor. Uma cópia em cada cliente.
  - Comandos para modificar os modelos são serializados e distribuídos através da rede
- SpacEclipse-CGMF
  - Integração de funcionalidades colaborativas em editores baseados em GMF;
  - Esta funcionalidade pode ser gerada como parte da geração de seu próprio editor GMF e workspace
- Dawn
  - Subprojeto de CDO
  - Visa prover acesso colaborativo a diagramas GMF.

Parte Prática

Instalação - Distribuição Eclipse

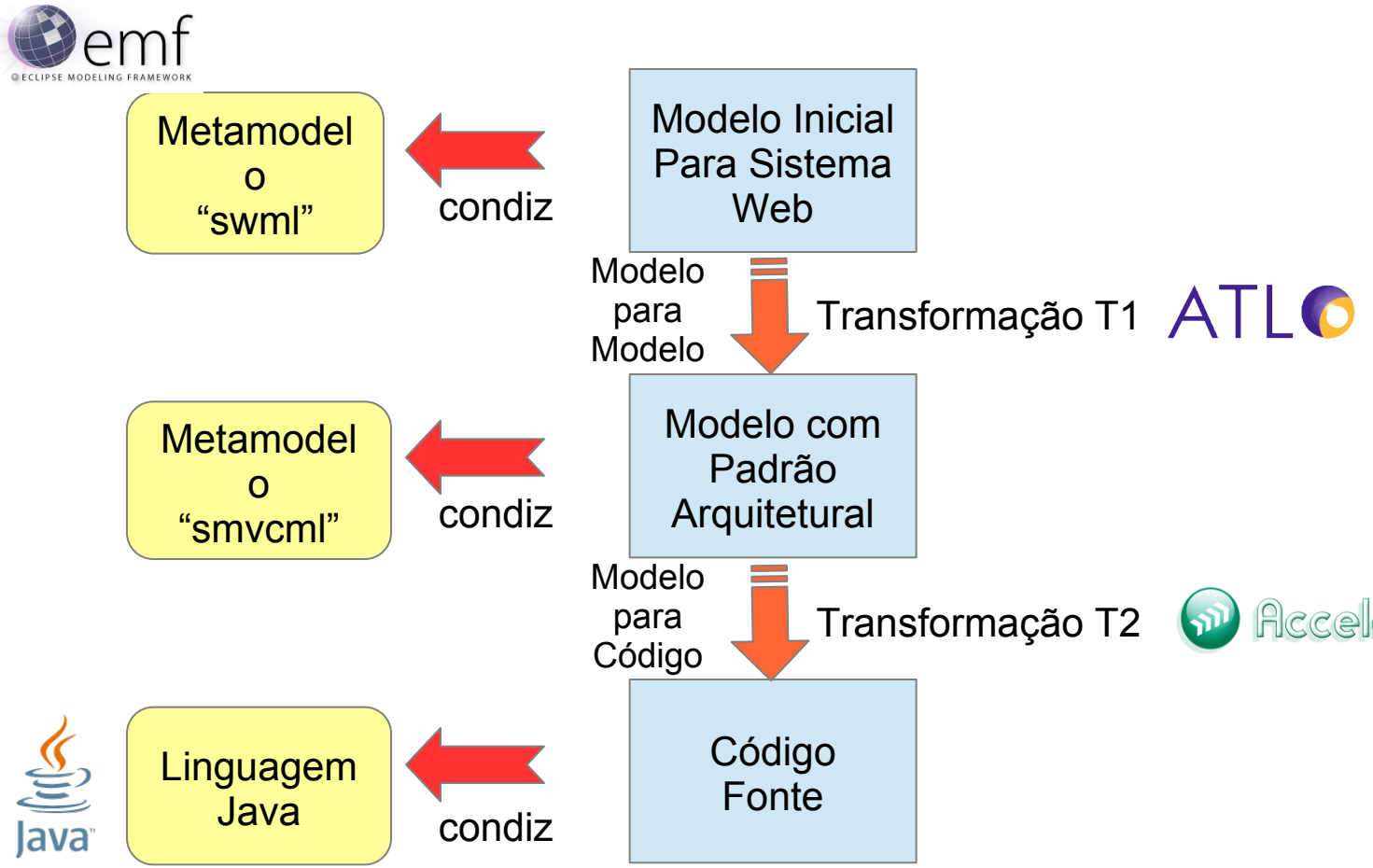


Instalação - Plugins Eclipse

- Ferramentas necessárias:

- Eclipse Modeling Project
  - Conjunto de ferramentas para modelagem com a plataforma Eclipse;
- Eclipse Modeling Framework
  - Arcabouço para desenvolvimento rápido de editores e transformadores de modelos;
- ATLAS Transformation Language
  - Linguagem de transformação de modelo para modelo;
- Acceleo Transformation Language
  - Linguagem de transformação de modelo para texto;

Arquitetura dos Modelos



Capítulos

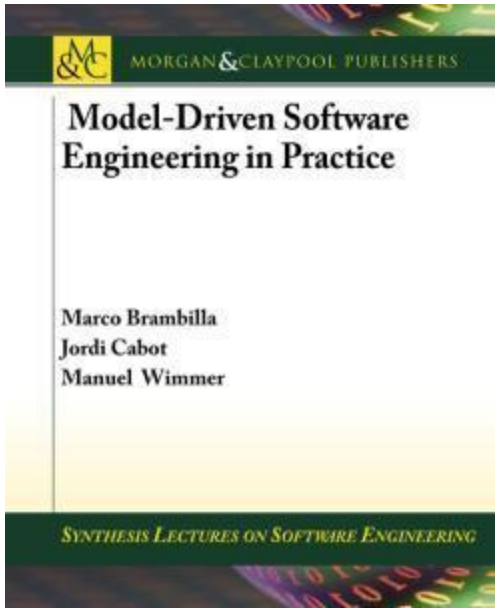
7
Metamodelo

8
Transformação Modelo-Modelo

9
Transformação Modelo-Código

MORGAN & CLAYPOOL PUBLISHERS

MODEL-DRIVEN SOFTWARE ENGINEERING IN PRACTICE

Marco Brambilla,
Jordi Cabot,
Manuel Wimmer.
Morgan & Claypool, USA, 2012.

www.mdse-book.com
www.morganclaypool.com
or buy it at: www.amazon.com