Concepção de uma Ferramenta Parser para Extração de dados do metamodelo ECore

Ítalo de Pontes Oliveira[1], Lucas Araújo Ramos[1], Rhavy Maia Guedes[1]

[1] Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB) – Campina Grande, Paraíba – Brasil

italo.oliveira@ieee.org, {lucas.du.a7x@gmail.com, rhavy.maia}@gmail.com

Abstract. *The Software Engineering offers clear and objective techniques for systems building. There are model based approaches for the software development cycle, such as the MDA. The OMG defines models as graphs that provide important information about the system. It is necessary that modeling teams seeking qualify and quantify the entities that are involved in the process. For this, they need to extract information from files based on the standard MOF metamodels. Thus, this study aims to perform data extraction for ECore metamodel by creating a tool-based software that allows the analysis of the elements contained in the model.*

Resumo. *A Engenharia de Software oferece técnicas claras e objetivas na construção de sistemas. Existem abordagens baseadas em modelo para o desenvolvimento do ciclo de software como, por exemplo, o MDA. A OMG define os modelos como representações gráficas que trazem informações importantes sobre o sistema. É necessário que as equipes de modelagem busquem qualificar e quantificar as entidades que estão envolvidas no processo. Para isso, necessitam extrair informações de arquivos de metamodelos baseados no padrão MOF. Desse modo, o presente trabalho tem como objetivo realizar a extração de dados do metamodelo ECore através da criação de uma ferramenta baseada em software que permitirá a análise dos elementos contido no modelo.*

1. Introdução

A engenharia de software oferece técnicas claras e objetivas na construção de sistemas, a fim de obter melhores resultados. Para isso, o modelo de processo de software é composto por atividades de apoio como, por exemplo, o Levantamento e Análise de Requisitos, a Modelagem do Sistema, a Implementação e os Testes [Presman 2010]. A Análise de Requisitos é o processo de avaliação das funcionalidades do sistema, e tem como objetivo evitar redundância e esclarecer os procedimentos a serem realizados pelo software. A Modelagem do Sistema constrói uma visualização prévia das funcionalidades de abrangência. O Desenvolvimento do Sistema define a estrutura de código. Os Testes tratam as falhas e exceções do desenvolvimento.

Para Qiuyan (2012), a modelagem permite uma representação mais clara dos requisitos a serem implementados no sistema. Além disso, serve como documentação de

contrato entre a equipe de desenvolvimento e o cliente. A Object Management Group (OMG) [OMG 2013] define os modelos como representações gráficas que trazem consigo informações importantes sobre o sistema. Segundo Liu (2005), os modelos são estruturados a partir de seus metamodelos que contém uma sintaxe abstrata e semântica estática de uma linguagem de modelagem específica de domínio.

Além disso, Piers (2012) ressalta a possibilidade de realizar transformações automáticas em modelos com o objetivo de melhorar a qualidade, evitar erros e economizar esforços no processo de desenvolvimento de software. A técnica de transformação automatizada permite que através de um modelo de entrada seja gerado um ou mais modelos de saída. O procedimento possibilita que as alterações sofridas em processos organizacionais modelados sejam rapidamente refletidas no software. Para isso, a OMG define padrões e metodologias para o processo de transformações de modelos como, por exemplo, o *Meta Object Facility* (MOF) que define uma arquitetura para criação de metamodelos, e o *Atlas Transformation Language* (ATL) linguagem procedural capaz de executar transformações unidirecionais.

No entanto, antes da execução da transformação do modelo ou modelagem, as equipes buscam qualificar e quantificar as entidades que estão envolvidas no processo, a fim de identificar relações e dependências dos elementos e o impacto causado pelas mudanças. A análise geralmente é realizada através das imagens dos diagramas gerados pelas ferramentas. No entanto, a análise visual é passível de erro, pois os diagramas podem estar representados em um nível de abstração alto.

Para extrair as informações contidas em um metamodelo é necessário que seja interpretados a estrutura de metadados do arquivo baseado em XML *Metadata Interchange* (XMI). O XMI permite o intercâmbio de metadados entre ferramentas baseadas no padrão da OMG como, por exemplo o *Eclipse Model Framework* (EMF) que utiliza o metamodelo ECore.

Neste sentido, o presente trabalho tem como objetivo realizar a extração de dados do metamodelo ECore através da criação de uma ferramenta baseada em software que permitirá a análise dos elementos contidos no modelo. Essas informações darão subsídios para que equipes de modelagem tomem decisões acerca da transformação de modelos.

2. Background

O *Model Driven Engineering* (MDE) é uma tendência da engenharia de software que reúne todas as abordagens baseadas em modelo para o ciclo de desenvolvimento de software. [Rensink 2006]. O *Model Driven Architecture* (MDA) faz parte do conjunto proposto pela MDE. O Objetivo do MDA é definir a estrutura e o comportamento do sistema em um nível de abstração mais alto que desconsidera tecnologias e implementações subjacentes. Sua proposta principal é diminuir o contato com código-fonte, deixando o esforço voltado para a análise e desenvolvimento de modelos. Dessa forma, o modelo deixa de ser apenas um guia, uma referência para o desenvolvimento, e passa a compor o software assim como o código-fonte [Moraes 2002].

Para definir sua estrutura semântica e sintática, o modelo pode utilizar um ou mais metamodelos. Segundo a OMG, o metamodelo confere vantagens na implementação e interoperabilidade de modelos, pois define regras, restrições e teorias úteis para a modelagem.

2.1. Estrutura de um modelo

A OMG padronizou a arquitetura de metamodelo através da criação do *Meta Object Facility* (MOF), desenhada quatro meta-camadas. O MOF é capaz de estruturar sintaticamente e semanticamente um conjunto de modelos [Silva 2008]. A Figura 1 mostra as camadas que compõe o padrão MOF, é possível identifica-las de baixo para cima pelos índices M0 até o M3.

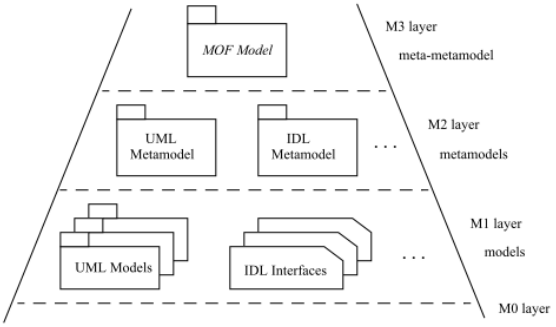

Figura 1. Arquitetura do Metamodelo MOF [OMG 2013].

O nível M0 é a camada mais baixa da arquitetura que constituem as instâncias concretas dos elementos representados num modelo; O nível M1 de Modelos responsável por representar os elementos que constituem as abstrações definidas pelo utilizador da UML no seu processo de modelagem; O nível M2 de Metamodelo é composto por elementos que constituem as abstrações definidas pelos criadores da UML, ou seja, estruturas para definição dos modelos (Por exemplo: Classe, Atributo, Método, Relacionamento, etc.); Por fim, o nível M3 de Meta-metamodelo que permite definir os elementos que serão responsáveis pela especificação dos metamodelos (Por exemplo: MetaClasse, MetaAtributo, MetaMétodo, MetaRelacionamento, etc.) [Silva 2008][Rensink 2006]. Durante a implementação do padrão MOF as ferramentas podem optar em reduzir ou aumentar a quantidade de camadas definidas pela OMF.

Baseado nas especificações da OMG o EMF permite a construção de ferramentas e aplicações baseadas em modelo de classe baseado na UML [Steinberg et al 2009]. O EMF converte os modelos criados para o formato ECore que contém informações acerca das classes definidas pelo modelo [Vogel, 2008]. É importante ressaltar que o metamodelo definido pelo ECore é baseado no padrão *Meta-Object Facilities* (MOF). Essa característica permite maior integração do EMF com outras ferramentas que também utilizam o padrão MOF.

Figura 2. Modelo de classe gerado pelo EMF [EMF 2013].

É possível observar na A Figura 2 um simples modelo de classe gerado através do EMF. A classe denominada *Book* contém o atributo "*title*" do tipo *String*, e "*pages*" do tipo int. O modelo da Figura 2 é descrito através da meta-informações presentes no documento do XML *Metadata Interchange* visto na Figura 3.

```
<xsd:schema targetNamespace="http:///library.ecore"
    xmlns="http:///library.ecore" xmlns:xsd="http://www.w3.org/2001/XMLSche
  <xsd:complexType name="Book">
    <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="pages" type="xsd:integer"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Figura 3. Representação textual XML Schema para um modelo EMF [EMF 2013].

A Figura 4 representa as quatro entidades básicas do metamodelo ECore. O ECore é composto pela entidade EClass, EAtrribute, EDataType, EReference. A EClass modela uma classe que pode possuir um identificador, cardinalidade e referências. A EAttribute modela um atributo que pode possuir nome e tipo. A EReference, representa a extremidade de uma associação entre as classes. A EDataType modela um tipo de atributo, podendo ser um tipo de dado primitivo ou composto.

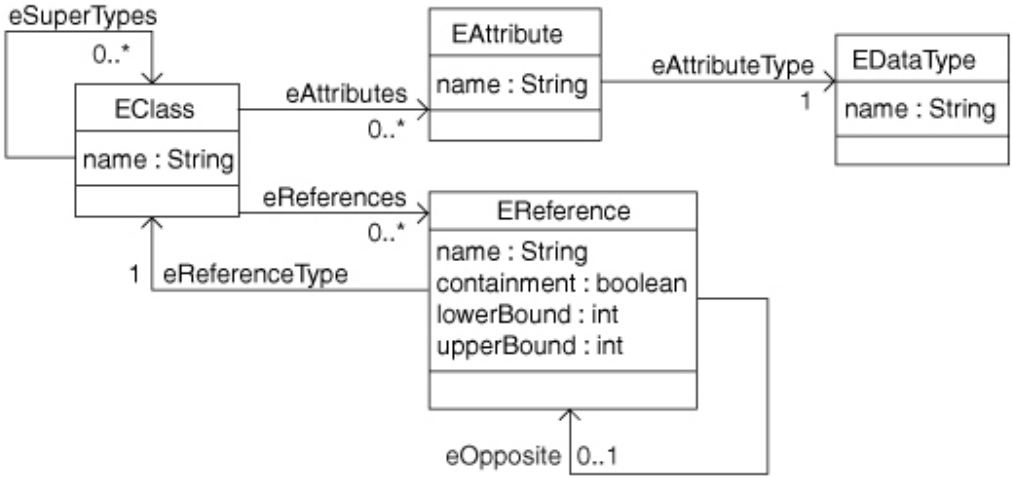

Figura 4. Diagrama de Classes para a representação feita por um modelo ECore.

3. Materiais e Métodos

O trabalho foi desenvolvido por um grupo de Iniciação Científica que investiga temáticas relacionadas a Modelagem Dirigida a Modelos (MDD) no Instituto Federal de Educação,

Ciência e Tecnologia da Paraíba campus Campina Grande. O grupo é formado por estudantes do curso Superior em Telemática e pelo professor orientador. Durante o período de revisão da literatura foram levantados aspectos importantes sobre a transformação automática de modelos como, por exemplo, a análise estática e extração de dados de metamodelos.

3.1 Ferramentas de modelagem

A pesquisa utilizou o ambiente de modelagem baseado no (EMF) para desenvolver o modelo presente na Figura 5. O modelo representa o Sistema de Orientação Docente-Discente que servirá como artefato de análise da ferramenta de análise estática e extração de informações de modelos.

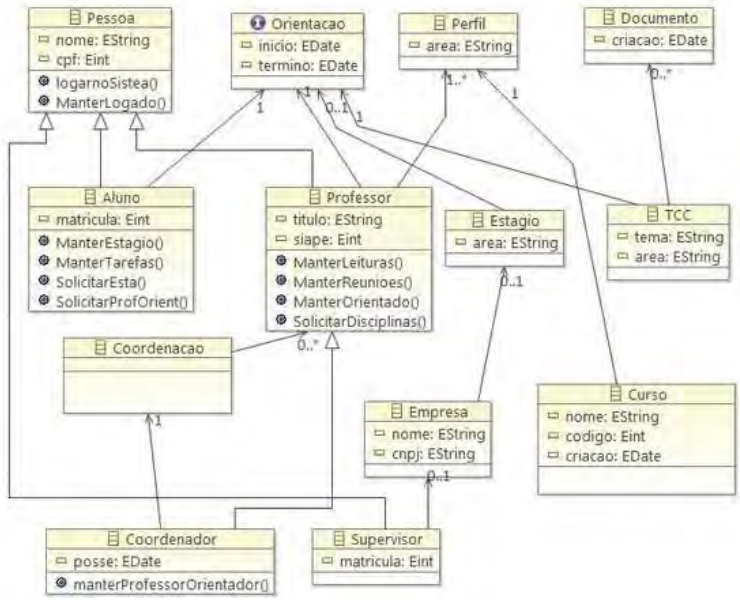

Figura 5. Modelo Sistema de Orientação Docente-Discente.

3.2. Especificação de Requisitos

Tabela 1 - Lista de requisitos para construção do sistema

Requisito	Descrição
Extrair dados do ECore	Extrair as informações contidas no arquivo XMI que representa o modelo de classe baseado em ECore. A extração resultará na criação de uma instância de objeto Java composto por uma lista de Classes que seguirá a definição do modelo EClass.
Exibir Relacionamento	Exibir informações sobre as classes que compõe o modelo. Será possível visualizar os atributos, agregações e

	heranças presentes nos relacionamentos bem como sua cardinalidade.
Gerar Relatório para Análise	Gerar um relatório capaz de quantificar e analisar as caracteristicas do modelo. O documento é sintetizado e de fácil interpretação, pois auxiliará no processo de transformação de modelo.

3.3. Implementação das funcionalidades

O sistema foi estruturado a partir da arquitetura prevista pelo Model-View-Controller (MVC) e codificado através da linguagem de programação Java. A codificação dos requisitos de Extração de dados do ECore e Geração de relatório realizou-se no ambiente de desenvolvimento do Eclipse. No entanto, para a construção das interfaces gráficas baseadas em Swing utilizou-se a ferramenta NetBeans.

Como forma de modularizar os comportamento referentes a Extração de dados do ECore a equipe resolveu criar uma API base responsável por transformar o arquivo XMI, baseado no ECore, em instância de objeto que representa o diagrama de classe mostrado na Figura 4.

4. Resultados

A aplicação inicia o procedimento de interpretação da estrutura do modelo de classe através da leitura de um arquivo XMI disponibilizado pela equipe de modelagem. A interpretação desde arquivo gera a instância de objeto que contem a lista de classe com suas características e comportamento. É possível observar na Figura 6 a exibição de todas as entidades existentes no modelo da Figura 5. A interface permite que a equipe de modelagem acesse as opções de atributos, métodos, relacionamentos inerente a cada classe. A opção Gerar Relatórios fornece um detalhamento quantitativo de relacionamentos, heranças, quantidades de atributos e comportamentos presentes em todo o modelo.

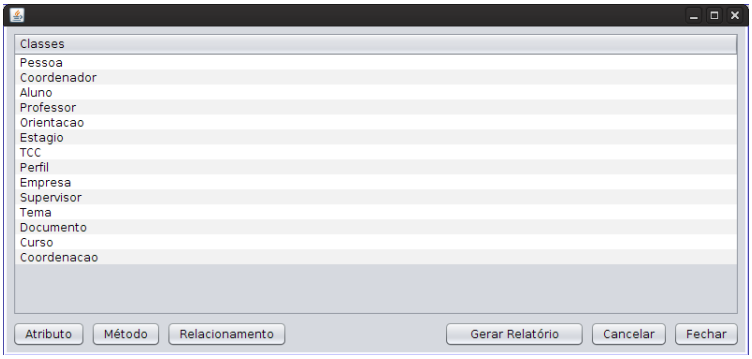

Figura 6. Tela exibindo as classes de um modelo de entrada.

5. Considerações Finais

A ferramenta poderá auxiliar empresas de desenvolvimento ou pesquisadores durante a realização da avaliação de estrutura de modelos baseados em ECore. Será possível que

uma equipe de engenharia de software avalie, através da ferramenta desenvolvida, a maturação de um modelo em linha de desenvolvimento de software, evitando assim, erros durante o processo de codificação de sistemas humanos.

Referências

Piers, W. and Bruneliere, H. (2012). ATL Concepts. Disponível em: <http://wiki.eclipse.org/ATL/Concepts>. Acessado em: 10 fev. 2013.

Vogel, L. (2008). Eclipse EMF. Disponível em: <http://www.vogella.com/articles/EclipseEMF/article.html>. Acessado em: 19 fev. 2013.

Burbeck, S. (1987). Applications programming in smalltalk-80(tm): How to use model-view-controller (mvc).

Castro, S. (2008/2009). Modelos de código java eclipse java model e emf java model. Technical report, Instituto Superior de Engenharia do Porto.

EMF (2013). Emf. the eclipse modeling framework (emf) overview. Disponível em: http://help.eclipse.org/indigo/nav/22 0. Acessado em: 07 jan. 2013.

Félix, R. (2010). Dsltranslator - ferramenta para transformacãoo de modelos. Master's thesis, Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia, Departamento de Informática.

Kleppe, A. G., Warmer, J., and Bast, W. (2003). MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Kreutz, M. and Silva, D. R. C. (2013). Metrópole digital. Disponível em: http://www.metropoledigital.ufrn.br/aulas avancado/web/disciplinas/desktop/aula03.html. Acesso em: 18 jan. 2013.

Lapenda, R., Madruga, P., and Loniewski, G. (2008). Utilizando transformacão de modelos para o desenvolvimento de softwares de qualidade.

Liu, J., He, K., Li, B., He, C., and Liang, P. (2005). A transformation definition metamodel for model transformation. In Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on, volume 2, pages 373 – 378 Vol. 2.

Lucrédio, D. (2009). Uma Abordagem Orientada a Modelos para Reutilização de Software. PhD thesis, Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo.

Moraes, T. (2012). Extração do comportamento especificado em modelos uml usando o eclipse modeling framework.

Oliveira, Í. P., Bezerra, T. S., Ramos, L. A., Guedes, R. M. (2012). Uso de modelos de integração para transformações de modelos baseados em metamodelos. VII CONNEPI.

OMG (2013). Documents associated with mof version 2.0. Disponível em: http://www.omg.org/spec/MOF/2.0/. Acesso em: 07 jan. 2013.

Pressman, R. (2010). Software engineering: a practitioner's approach. McGraw-Hill higher education. McGraw-Hill Higher Education.

Qiuyan, L., Jie, T., Qiuhong, P., Ji, W., and Chao, L. (2011). Automatic transformation technology from aadl model to uml model. In Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, pages 255–258.

Rahimi, S. (2010). Specification of uml model transformations. In Software Testing, Verification and Validation (ICST), 2010 Third International Conference on, pages 323–326.

Selic, B. (2003). The pragmatics of model-driven development. IEEE Softw., 20(5):19–25.

Silva, A. d., Videira, C., and Atlântico, C. (2008). UML, metodologias e ferramentas CASE: linguagem de modelação UML, metodologias e ferramentas CASE na concepção e desenvolvimento de sistemas de informação. Number v. 1. Centro Atlântico.

Sommerville, I. (2003). Engenharia de Software. Addison Wesley, São Paulo, SP, 6 edition. Tradução André Maurício de Andrade Ribeiro; Revisão técnica Kechi Hirama.

Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E. (2009). EMF: Eclipse Modeling Framework 2.0. Addison-Wesley Professional, 2nd edition.

Zhang, J., Liu, S., Wang, X., and Qin, T. (2008). A model transformation classification method used in qvt. In IT in Medicine and Education, 2008. ITME 2008. IEEE International Symposium on, pages 464 –468.