VI CONGRESSO DE INICIAÇÃO CIENTÍFICA DA UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

PIBIC/CNPq/UFCG-2009

INVESTIGAÇÃO E IMPLEMENTAÇÃO DE TÉCNICAS DE MINERAÇÃO DE DADOS PARA DESCOBERTA DE PADRÕES EM MDA

Fábio de Sousa Leal[1], Franklin de Souza Ramalho[2]

RESUMO

Este trabalho teve o objetivo de investigar a existência de padrões no âmbito de Model-Driven Architechture (MDA), que trata-se de um novo paradigma proposto pela Object Management Group – OMG para a Engenharia de Software. Como método auxiliar para realizar a descoberta de padrões utilizamos a abordagem de Mineração de dados, que trata-se de uma maneira automatizada para se extrair padrões que não eram conhecidos previamente através da análise de grandes massas de dados.

Palavras-chave: MDA, padrões, Model Driven Development, Data Mining.

INVESTIGATION AND IMPLEMENTATION OF DATA MINING TECHNIQUES FOR PATTERNS DISCOVERY IN MDA

ABSTRACT

The goal of this project is to investigate the existence of patterns in the context of Model-Driven Architecture – MDA, a new paradigm proposed by the Object Management Group – OMG. In order to perform the discovery of patterns we adopted data mining approach that allows to automatically extract patterns that were not previously known through the analysis of large amounts of data.

Keywords: MDA, patterns, Model Driven Development, Data Mining.

INTRODUÇÃO

Cada vez mais surgem novas abordagens e tecnologias no âmbito da engenharia de *software* com intuito de oferecer, dentre outros aspectos, uma maior automação e, conseqüentemente, maior produtividade no processo de desenvolvimento de um *software*.

Como exemplo de uma abordagem que atua nesse sentido temos o MDA (*Model-Driven Arquitecture*) [11], uma arquitetura proposta pela OMG [20] que tem como objetivo principal deslocar o esforço e tempo durante o ciclo de vida de um *software* das tarefas de implementação e testes para tarefas de modelagem, meta-modelagem e transformações de modelos. Com o uso de MDA, podemos construir sistemas consistentes e independentes de plataforma focando principalmente em suas tarefas de modelagem.

[1] Aluno do Curso de Ciência da computação, Depto. de Sistemas e computação, UFCG, Campina Gande , PB, E-mail: fabiosl@dsc.ufcg.edu.br

[2] Cientista da Computação, Prof. Doutor, Depto. de Sistemas e computação, UFCG, Campina Gande , PB, E-mail: franklin@dsc.ufcg.edu.br

A arquitetura MDA reconhece a importância dos modelos no processo de fabricação de um software e os torna um ponto-chave nas tarefas de desenvolvimento de um sistema. Nesse sentido, MDA sugere que os modelos de sistemas sejam especificados nas seguintes visões [21]: (i) CIM (*Computational Independent Model*) - especifica os modelos independentes de computação, representando apenas requisitos do funcionamento do sistema; (ii) PIM (*Platform Independent Model*) - especifica os requisitos e projeto de *software* independentes de qualquer plataforma de implementação; (iii) PSM (*Platform Specific Model*) – especifica os modelos com todos os detalhes da plataforma para a qual o *software* será implementado; e (iv) Código - produto final e executável.

Muitos projetos de software estão aderindo a esta nova abordagem que MDA propõe. Isto se deve, dentre outros fatores, ao grande número de vantagens que ela proporciona. A primeira delas é a produtividade, e um dos motivos para isto é a redução da quantidade de código gerado manualmente, uma vez que grande parte do código será gerada de forma automática a partir da transformação PSM ⇨ Código, acarretando na redução de tempo e custo do desenvolvimento do projeto. A portabilidade é uma outra vantagem que merece destaque, pois uma vez que o PIM é independente de plataforma, ele pode ser transformado automaticamente em vários PSMs, dessa maneira, permitindo que um sistema opere livremente em diferentes plataformas.

Um terceiro benefício claro de MDA é a facilidade na manutenção e documentação, tendo em vista que a manutenção não é mais realizada em nível de código, mas sim em um nível mais alto de abstração, no PIM, mantendo a documentação atualizada constantemente. A interoperabilidade é outra vantagem da abordagem MDA. Embora os PSM's especificados para plataformas diferentes não sejam interoperáveis diretamente, podemos utilizar o conceito de *bridges* para realizarmos a comunicação entre PSM's e também entre códigos de diferentes plataformas.

Além das vantagens apresentadas, a abordagem MDA poderia obter um desempenho ainda melhor no desenvolvimento dos seus artefatos – modelos, meta-modelos e transformações, seja reutilizando soluções de problemas que ocorrem freqüentemente em um mesmo contexto, ou definindo um vocabulário comum para discutir problemas e soluções de projetos. Esses tipos de atividades fazem com que um sistema complexo possa ser mais facilmente compreendido pelas diversas pessoas que fazem a manutenção do mesmo.

Este cenário reflete outro contexto, denominado "padrões". Segundo [22], um padrão é uma entidade que descreve um problema que ocorre repetidamente em um ambiente e então algo descreve a essência de uma solução para este problema, de tal forma que você possa usar essa solução várias vezes, sem nunca utilizá-la do mesmo modo.

O uso de padrões no desenvolvimento de *software* proporciona uma série de vantagens, dentre as quais podemos mencionar a aprendizagem com a experiência de outras pessoas, a facilidade para reestruturar um sistema, a confiabilidade no reuso de padrões cujas soluções são comprovadas, a evolução do código e a modularidade do sistema.

Dentro da infra-estrutura de MDA, em seus diferentes artefatos, a identificação de padrões é uma atividade incipiente e pouco se tem produzido. A maioria dos trabalhos presentes neste ramo ainda se concentra na identificação de como relacionar ou representar padrões de projeto [23] no escopo desta infra-estrutura, como os apresentados em [24] e [25]. Por outro lado, alguns trabalhos abordam camadas ou padrões específicos de certos contextos da arquitetura MDA, como por exemplo, padrões para especificação OCL (*Object Constraint Language*) [26] e padrões para transformação de modelos [27][28].

Já outro trabalho [29], identifica três padrões de meta-modelagem específicos para o meta-modelo de *SmallTalk* [30]. No entanto, eles não utilizam a infra-estrutura MDA, nem os seus formalismos, como MOF (*Meta Object Facility*) ou OCL.

Em suma, podemos observar que os trabalhos já existentes ou não abordam uma camada ou padrão específico de MDA, ou o fazem de forma bem restrita, muitas vezes com erros de entendimento. Adicionalmente, é importante a ressalva de que os padrões existentes foram identificados de forma manual, sem a aplicação de nenhuma técnica ou mecanismo para descoberta automática de padrões. Com isso, tal procedimento está propício a falhas e torna-se inviável quando se trata da identificação de padrões em um escopo muito grande.

A mineração de dados é um processo que se propõe a resolver este tipo de problema, descobrindo os padrões automaticamente em grandes bases de dados. De acordo com [31], mineração de dados é a atividade de descoberta de padrões interessantes a partir de uma grande quantidade de dados armazenados em diferentes repositórios. Trata-se de uma área multidisciplinar envolvendo não apenas bancos de dados e *data warehouse*, mas também estatística, aprendizagem de máquina, visualização de dados, recuperação de informação e computação de alto desempenho, dentre outros aspectos.

Estes padrões de dados podem ser minerados a partir de diferentes fontes de conhecimento, como bancos de dados relacionais, objeto-relacionais, bancos de dados espaciais, bancos de dados textuais e também a Web. Deste modo, o processo de mineração de dados, acompanhado de suas técnicas, metodologias e ferramentas, surge como potencial instrumento para tarefa de descoberta de padrões dentro

da infra-estrutura MDA, em seus diferentes artefatos. Com o exposto, concluímos que a identificação de padrões no contexto de MDA é bastante promissora.

ATIVIDADES REALIZADAS

Este trabalho foi desenvolvido no Grupo de Métodos Formais - GMF do Departamento de Sistemas e Computação no Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande – PB. É importante ressaltar que o período de realização das atividades foi condizente com as datas previstas no cronograma de execução das atividades, ilustrado na Tabela 1:

1 - Revisão bibliográfica sobre técnicas de Mineração de Dados e Formalismos de MDA, e também sobre ferramentas que dêem suporte a ambas;
2 - Proposta de técnicas para descoberta de padrões em MDA;
3 - Proposta de quais artefatos de MDA cobrir pelas técnicas propostas em (2);
4 - Implementação/Adaptação das técnicas propostas em (2) para descoberta de padrões dos artefatos identificados em (3);
5 - Identificação e validação dos Padrões em MDA;
6 - Escrita de Relatórios técnicos e artigos.

Atividades	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul
1	x	x	x									
2				x	x	x						
3				x	x	x						
4							x	x	x	x		
5										x	x	
6					x	x	x	x	x	x	x	x

Tabela 1– Cronograma de Execução.

Revisão Bibliográfica

Durante os primeiros meses do projeto foram estudados os conceitos de MDA através do livro [2], indicado pelo professor orientador. Além do livro, materiais relativos ao assunto foram consultados diretamente do site da OMG[11], grupo regulamentador do MDA.

Ainda no período de revisão bibliográfica foram estudados os conceitos iniciais de assuntos que poderiam ajudar no desenvolvimento da pesquisa, tais como Bancos de Dados e Diagramas da UML, elementos pivôs de MDA. Os estudos relativos a Bancos de dados (BD) foram de grande importância, pois quando se está lidando com *Data Mining*, os conhecimentos sobre BD's são exigidos em todo momento. Como referência para as consultas bibliográficas foram usadas a apostila do Professor Oswaldo Kotaro – IME [12], além dos materiais de aula do professor Cláudio Batista – UFCG [13].

Para a introdução à UML e seus diagramas foi usado o Guia de Consulta Rápida para UML – Novatec [14] e os materiais de aula da disciplina de Sistemas de Informação II, do Professor Franklin Ramalho – UFCG [15]. O conceito de Mineração de Dados foi abordado pelos livros [3] e [4], ambos também por indicação do orientador. Por se tratar de uma área bastante complexa, que envolve conceitos ainda não vistos, a literatura foi restringida apenas às partes que eram do interesse do foco da pesquisa:

- Introdução ao *Data Mining;*
- Seleção e estruturação dos dados para as tarefas de mineração;
- Limpeza dos dados e pré-processamento;
- Transformação dos dados;
- Estudo dos algoritmos a serem utilizados na pesquisa;

Proposta de técnicas para descoberta de padrões em MDA:

[32] tem a finalidade de mostrar como podemos usar o processo de mineração de dados para a descoberta de uma biblioteca de padrões que podem ser reutilizáveis em aplicações já existentes. Para isto, ele propõe a aplicação de regras de associação generalizadas, as quais apresentam uma melhoria em relação às regras de associação tradicionais devido à aplicação de um novo elemento denominado taxonomia. De maneira geral, a taxonomia utiliza o conceito de especialização para formular as regras, dessa forma, nos permite realizar a mineração em diferentes níveis de abstração.

Embora não seja muito recente e não utilize uma abordagem inovadora para a mineração de dados, [32] torna-se um trabalho interessante pela aplicação que ele faz da mineração de dados para descobrir uma biblioteca de padrões reutilizáveis em aplicações já existentes. Este trabalho pode servir para nos despertar outras maneiras de identificar padrões em meta-modelos, seja aplicando regras de associação generalizadas ou outras técnicas.

As regras de associação generalizadas, em sua essência, buscam fazer com que tenhamos regras do tipo: *Se A → Então B (conf. X%)*. Nesse exemplo, "A" é uma condição existente que desencadeia "B" com uma confiança de porcentagem igual a X. Associações desse tipo são as que precisamos para gerar os padrões, pois fazem com que as características dos artefatos de MDA relacionem-se entre si.

Portanto, optou-se inicialmente por utilizar principalmente as regras de associação como principal técnica para descoberta de padrões em MDA, apesar de existirem outras técnicas também úteis, tais como as regras de classificação.

Proposta de quais artefatos de MDA cobrir pelas técnicas:

Durante as reuniões semanais com o orientador foram levantados os possíveis artefatos de MDA que poderiam ser explorados. Como resultado dessas discussões, a descoberta de padrões em meta-modelos MOF apresentou-se bastante pertinente, uma vez que os mesmos auxiliam bastante no desenvolvimento e entendimento de meta-modelos, que são peças fundamentais para a especificação de transformações (Tarefas estratégicas no contexto de MDA). Como motivação auxiliar para a exploração de padrões em meta-modelos MOF, temos no laboratório em que foi realizado a pesquisa uma aluna da pós graduação que está desenvolvendo seu mestrado com foco nessa área.

Implementação/Adaptação das técnicas propostas para descoberta de padrões nos artefatos propostos

Como a atividade de implementação e adaptação das técnicas propostas para descoberta de padrões nos artefatos propostos é muito grande, subdividimos a tarefa nas seguintes rotinas: (i) Escolha da Ferramenta de Mineração, (ii) Estudo das API's de XML disponíveis em Java, (iii) Definição da arquitetura do Sistema de suporte, (iv) Implementação do Sistema de Suporte, (V) *Refactoring* do código fonte.

Escolha da Ferramenta de Mineração:

No decorrer da pesquisa foi realizada primeiramente uma análise comparativa entre os softwares de *Data Mining* que poderiam ser adotados. Como existem várias ferramentas gratuitas disponíveis, guiamos a pesquisa baseados nas seguintes métricas, retiradas de [1]:

- Habilidade de acesso a uma variedade de fontes de dados, de forma *online* e *offline*;
- Capacidade de incluir modelos de dados orientados a objetos ou modelos não padronizados (tal como multimídia, espacial ou temporal);
- Capacidade de processamento com relação ao número máximo de tabelas/tuplas/atributos;
- Capacidade de processamento com relação ao tamanho do banco de dados;
- Variedade de tipos de atributos que a ferramenta pode manipular;
- Tipo de linguagem de consulta;
- · Funcionalidade;

Foram analisadas quatro ferramentas ao todo:

- Weka[5]: É a mais utilizada em âmbito acadêmico. Como características principais podemos destacar que a mesma é feita na linguagem Java, fornece um bom suporte às tarefas de pré-processamento de dados e que apresenta muitos algoritmos já implementados. Um ponto negativo do Weka é que não possui suporte à Minerção de dados Multi-Relacional.
- Tanagra[16]: O tanagra, assim como o Weka, tem seu desenvolvimento focado com uma maior intenção para o âmbito acadêmico. O *software* preza bastante pela possibilidade de que o próprio

usuário faça e execute seus algoritmos. É bom ressaltar que o mesmo também possui uma vasta coleção de algoritmos já implementados. Um dos pontos negativos do Tanagra é que o mesmo só apresenta versões para o Sistema Operacional (S.O) Microsoft Windows.

- Orange[17]: O Orange, por ser uma ferramenta escrita na linguagem Python, possui uma pequena perda em relação aos outros dois anteriores, uma vez que a referida linguagem apresenta uma perda de eficiência em relação a linguagens de mais baixo nível como C, por exemplo. Também é importante ressaltar que o Orange, assim como o Weka e Tanagra, carrega todo o banco de dados que está sendo lido para a memória principal do computador, limitando, algumas vezes, o tamanho máximo do mesmo.
- KDB2000[18]: KDB 2000 é uma única ferramenta que integra acesso a bancos de dados, pré-processamento de dados e técnicas de transformação, além de um grande leque de algoritmos usados na mineração. A ferramenta, no entanto, não recebe atualizações desde 2004. Em seu site é informada a compatibilidade apenas com o S.O Microsoft Windows 98, o que torna seu uso inviável.

Como conclusão da análise optou-se pelo software Weka[5], por ser o mais bem avaliado nas características consideradas. Após a decisão da ferramenta que seria utilizada, foi novamente consultado o livro [3], pois o mesmo trás uma abordagem completa a respeito da ferramenta Weka. Foram realizados ainda alguns testes experimentais com as bases de dados que já vêm disponíveis no programa, além de testes com *datasets* reais, retirados de [6], com o intuito de familarizar-se com o software.

Estudo das *Application Programming Interface* (API's) de XML disponíveis em Java:

Para que se pudesse dar início à construção da ferramenta, seria necessário que fossem estudadas algumas API's para se lidar com *Extensible Markgroup Language* (XML)[19] em Java, uma vez que os meta-modelos a serem minerados estão disponíveis no formado *XML Metadata Interchange* (XMI), que é um padrão da OMG para troca de informações baseado em XML.

- *Document Object Model* (DOM)[7]: Uma API na qual o Documento inteiro é armazenado na memória principal do computador, num formato de árvore de nodos. Por esse fato, pode apresentar restrições relativas à memória caso o documento que será analisado seja muito grande.
- *Simple API for XML* (SAX)[8]: Principal alternativa ao DOM, o SAX provê acesso de maneira serial ao arquivo XML. É ideal para se lidar com grandes documentos XML, pois a leitura do documento é feita de forma seqüencial, ao contrário da API DOM.
- *JDom* [9]: Uma API simples que fornece uma representação em Java dos elementos de um documento XML. Torna fácil a manipulação dos nós do documento, além de proporcionar uma leitura fácil e dinâmica do documento. O JDom é uma alternativa ao DOM e ao SAX, que são as 2 API's mais utilizadas para manipulação de XMLs, embora integre-se bem com ambos.
- *XML Path Language* (XPath)[10]: XPath é uma API que provê uma navegação fácil e rápida em documentos XML. Com o uso da mesma, podemos percorrer os nós do documento, especificando-se através de expressões bem definidas qual(is) o(s) nó(s) que queremos acessar.

Como resultado do estudo das API's optou-se por fazer uso do XPath aliado ao JDom, por serem mais simples e preparadas para o tipo de pesquisa que será realizado com o XML.

Definição da Arquitetura do Sistema de suporte:

Após o período de revisão bibliográfica foi iniciada a construção de uma ferramenta que fosse capaz de integrar a leitura dos meta-modelos e a mineração dos mesmos de forma automatizada. A princípio, a intenção da ferramenta era povoar automaticamente a base de dados na qual seria feita a mineração, uma vez que tal tarefa é inviável e passível a erros de forma manual.

O software consiste basicamente em quatro módulos principais:

- Extrator: Parte do programa que é responsável pela leitura do meta-modelo e extração das informações necessárias do mesmo. É na etapa de codificação do leitor que serão utilizados os conhecimentos adquiridos sobre as API's de manipulação de XML.
- Gerador: É a parte da ferramenta responsável pela geração de um arquivo de extensão .ARFF, pronto para a aplicação dos algoritmos de mineração.
- Executor dos Algoritmos: É a parte do programa que trata da execução dos algoritmos de mineração de dados. Nessa parte da ferramenta o usuário tem a possibilidade de incluir seu próprio algoritmo, se achar que os que já estão presentes são insuficientes. Vale ressaltar que os

algoritmos presentes nesta parte do software serão vindos do Weka. Isso é legal, já que a licença do Weka é a *General Public License* (GPL), tal qual a da ferramenta que está sob desenvolvimento.

- Interpretador dos resultados: É a parte responsável pela interpretação dos resultados obtidos na etapa anterior, transformando-os em conhecimentos válidos e possivelmente novos padrões que serão descobertos.

A figura a seguir ilustra a arquitetura do software. Note que o Executor associa-se diretamente com o Weka, implementando os seus algoritmos.

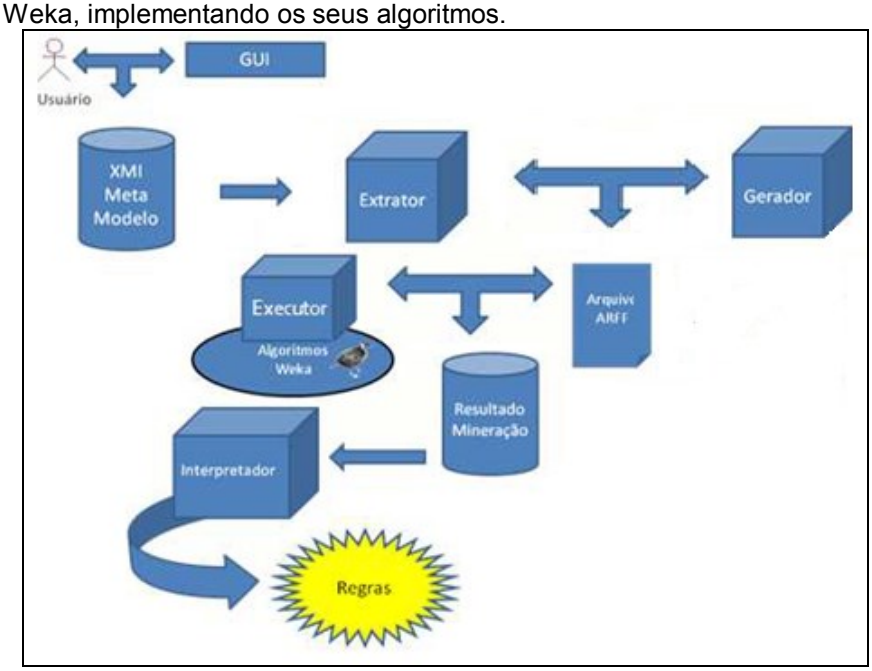

Figura 1 – Arquitetura da Ferramenta de Suporte

Implementação do sistema de Suporte

O programa consiste basicamente em:

- Ler o meta-modelo para que gere a saída com as instâncias no formato utilizado pelo Weka (.ARFF);
- Fornecer integração com os algoritmos nativos do Weka, fazendo com que os mesmos estejam disponíveis dentro do escopo do programa.
- Fornecer suporte para que o usuário possa usar seus próprios algoritmos de mineração com as bases de dados obtidas;
- Exibir os resultados de forma compreensível para o usuário;

A ferramenta facilita bastante a mineração de meta-modelos diferentes, pois trabalha manipulando meta-modelos descritos em *Meta Object Facility* (MOF), que é a linguagem padrão para meta-modelos proposta pela OMG.

***Refactoring* do Código Fonte:**

Durante o desenvolvimento da ferramenta foram realizados vários refatoramentos do código fonte do sistema. Como conseqüências desses refatoramentos conseguimos fazer com que o software lidasse de uma maneira mais simples com meta-modelos descritos em XMI, de modo a tornar mais simples a manipulação destes arquivos em nossa pesquisa. Como o sistema ainda está sob desenvolvimento, mais testes, com mais meta-modelos distintos, são necessários para que ocorra a validação do mesmo. Com o uso do refactoring, alcançamos um maior grau de reuso e modularidade do código feito.

RESULTADOS

Implementação e aplicações do protótipo:

Para tornar viável a construção da ferramenta, foi utilizado o meta-modelo da UML2.1.2 [4] como base. Como o meta-modelo é descrito em MOF, a ferramenta tornou-se compatível com qualquer outro

meta-modelo que seja descrito nessa mesma abordagem. O programa mostrou-se capaz de gerar os resultados necessários para a pesquisa

O *software*, até o presente momento, funciona da seguinte forma:

- Realiza a leitura de um meta-modelo genérico descrito em MOF através de um arquivo no formato XMI, que contém todas as informações presentes a respeito do meta-modelo com o qual estamos lidando.
- Gera um arquivo das instâncias obtidas no formato reconhecido pela Weka, de extensão .ARFF, com todos os dados extraídos.

Como passo inicial tomamos apenas as informações do meta-modelo de Máquinas de Estado da UML como válidas, pois a mineração de todo o MM de MOF apresentava-se bastante complexa. Poderíamos ainda usar classes, atributos, operações e associações, pois existem as definições desses elementos no meta-modelo da UML.

Após os primeiros testes e conseguinte validação da ferramenta, realizamos a pesquisa incluindo todo o escopo da UML, de maneira a testar a robustez da aplicação de forma geral.

A parte de aplicação dos algoritmos ainda está sendo realizada manualmente com o Weka, pois a integração do mesmo com a ferramenta desenvolvida não obteve sucesso imediato. A integração dos algoritmos elaborados pelo usuário se dará na mesma época em que for realizada a integração implementação do Executor de algoritmos, uma vez que as duas atividades apresentam-se interligadas.

Em relação aos casos de atributos com valores ausentes, o programa já faz um tratamento de forma a definir valores *default* para cada um destes. Vale destacar que este valor *default* não é aleatório, pois ele está definido na especificação da UML.

Regras de Associação

Para encontrarmos regras que apresentassem correlações e associações interessantes entre os relacionamentos do meta-modelo da UML, decidimos utilizar a funcionalidade conhecida como Análise de Associação durante a tarefa de *Mineração de Dados* do processo KDD. A ferramenta que adotamos para realizar tal tarefa, a Weka, oferece alguns algoritmos para a Análise de Associação. Dentre eles, utilizamos o *Apriori* devido a sua simplicidade e versatilidade em grandes bancos de dados. Para encontrarmos regras diversificadas, executamos o *Apriori* com diferentes configurações dos seus parâmetros, tais como o limite mínimo de confiança e suporte, e o número de regras geradas. Como resultado, obtivemos um grande número de regras de associação, algumas delas bastante interessantes, outras redundantes, óbvias ou muito específicas.

As regras foram interpretadas e avaliadas durante a tarefa de *Interpretação/Avaliação* do processo KDD. Para avaliarmos as regras selecionadas, tivemos que analisar cada uma, individualmente, comparando com a especificação do meta-modelo da UML. Dessa forma, verificamos se de fato a regra é válida. As regras mais interessantes podem nos ajudar a encontrar tendências que podem ser usadas para compreender e explorar padrões no comportamento dos dados.

A seguir, ilustramos quatro regras interpretadas como segue. R1 especifica com 100% de confiança que quando uma meta-classe representa “o todo” do relacionamento de composição, ela sempre deve possuir multiplicidade máxima igual a 1. R2 especifica com 84% de confiança que quando uma meta-classe representa “a parte” do relacionamento de composição, ela deve possuir multiplicidade mínima igual a 0. R3 especifica com 87% de confiança que quando uma meta-classe representa “a parte” do relacionamento de composição, ela deve ser concreta. R4 especifica 100% de confiança que quando uma meta-classe representa “a parte” do relacionamento de composição, ela sempre deve ser navegável.

R1: class.composition=owner → class.upper=1 conf:(100%)
R2: class.composition=owned → class.lower=0 conf:(84%)
R3: class.composition=owned → class.isAbstract=false conf:(87%)
R4: class.composition=owned → class.isNavigable=true conf:(100%)

Identificação e validação dos padrões em MDA

Durante a tarefa de *Implantação do Conhecimento* do processo KDD, as regras consideradas válidas foram analisadas em conjunto no intuito de se extrair as mais interessantes e verificar as novas descobertas. É nesta etapa que o conhecimento adquirido no decorrer das etapas anteriores deverá ser concretizado. No nosso caso, a concretização se dá em forma de padrões para meta-modelagem.

É importante ressaltar que este trabalho vai além da interpretação das regras de associação geradas. Utilizamos o processo KDD com a finalidade de obtermos, através da interpretação das regras, o conhecimento necessário para se compor padrões com base em algum domínio específico, que no nosso caso são os meta-modelos MOF.

Para a composição de padrões, primeiramente, tivemos que analisar as regras válidas e verificar quais delas possuem características em comum. A partir do conjunto de regras resultante dessa verificação, observamos o comportamento, a semântica e as influências que umas tinham sobre as outras. Dessa forma, adquirimos a essência desse conjunto de regras para a composição de padrões.

Padrão *CompositionMemberEnds*

Analisando os atributos e valores das regras R1, R2, R3 e R4, percebemos que há similaridades entre elas, como por exemplo, todas apresentam características de relacionamento de composição em seus antecedentes. Já os seus conseqüentes são formados por atributos distintos, os quais deduzem aspectos importantes de um relacionamento de composição. Considerando também a forte confiança que estas regras apresentam, agrupamos todas elas e formamos o padrão *CompositionMemberEnds.* Este padrão é ilustrado na Figura 1: a meta-classe *Owner* representa "o todo" de um relacionamento de composição e a meta-classe *Owned* representa "a parte".

O padrão especifica que em um relacionamento de composição: (R1) "o todo" deve ter a multiplicidade máxima igual a 1; (R2) "a parte" deve ter a multiplicidade mínima igual a 0; (R3) "a parte" deve ser uma meta-classe concreta; e (R4) "a parte" deve ser navegável a partir da meta-classe referente ao "todo". Nesse sentido, a Figura 1 representa todas as características que um trecho de meta-modelo deve ter depois que o padrão *CompositionMemberEnds* for aplicado.

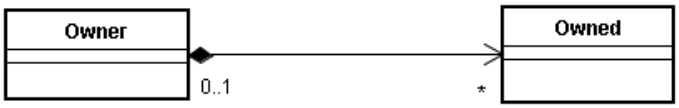

Figura 1. Definição do padrão *CompositionMemberEnds*.

O padrão *CompositionMemberEnds* deve ser aplicado quando se precisa construir meta-modelos com relacionamento de composição unidirecional entre duas meta-classes: apenas "o todo" pode navegar até o outro lado do relacionamento. O uso deste padrão também é uma maneira de evitar que pessoas inexperientes em meta-modelagem venham a construir relacionamentos de composição de forma inadequada, como por exemplo, com navegabilidade bidirecional.

Com a utilização deste padrão, podemos obter uma facilidade na documentação e entendimento dos meta-modelos. Uma vez que conhecemos o padrão, podemos facilmente identificá-lo em qualquer parte do meta-modelo. Outra vantagem é o reuso de partes ou meta-modelos completos.

Escrita de Artigos e relatórios técnicos

Como forma de publicação do conhecimento adquirido durante esse ano de iniciação científica foram escritos dois artigos a serem publicados em eventos. O primeiro deles foi intitulado "Applying Data Mining Algorithms for Automatically Discovering Patterns in MOF Metamodels" e foi encaminhado para a "Models Conference - www.modelsconference.org". O Models é uma das maiores conferências mundiais que tem área voltada ao desenvolvimento de software dirigido a modelos e áreas afins. Não tivemos o artigo aceito na conferência, no entanto, os *feedbacks* do mesmo foram muito positivos na continuidade de nossa pesquisa.

Houve também a escrita do segundo artigo, intitulado "Mineração de Regras de Associação para Descoberta Automática de Padrões em Relacionamentos dentro de Meta-modelos MOF", que foi submetido para o V Workshop em Algoritmos e aplicações de mineração de dados, evento que ocorre em junto ao SBBD, que é o simpósio mais importante da área em termos nacionais. No momento, ainda estamos aguardando o resultado final da avaliação do artigo

CONCLUSÕES

Com o término desse primeiro ano de iniciação científica o trabalho exposto mostrou-se proveitoso, pois contribui com a disponibilização de uma ferramenta de auxílio para a manipulação de meta-modelos genéricos descritos em MOF. O código oficial da ferramenta já está disponível em um repositório de SubVersion (SVN) e pode ser acessado e incrementado por qualquer membro da comunidade científica que o julgar interessante. A ferramenta, assim como sua documentação completa encontra-se disponível no repositório do Google Code. Toda a documentação necessária para o entendimento da ferramenta encontra-se junto ao código fonte da mesma, que está disponível em http://code.google.com/p/toolsupport/.

A pesquisa contribuiu ainda com a descoberta de um padrão até então desconhecido a ser seguido na construção de novos meta-modelos. A utilização do padrão *CompositionMemberEnds* é útil pois contribui com o entendimento de novos meta-modelos que são gerados por pessoas que ainda não possuem uma boa experiência nessa área.

A ferramenta ainda encontra-se em uma constante evolução, e mais um ano de iniciação científica foi proposto nesse mesmo contexto. Com esse próximo estágio da pesquisa procuramos obter mais padrões de forma a criar um de catálogo de padrões a ser seguido quando se está realizando uma atividade de meta-modelagem.

Uma outro aspecto positivo da conclusão desse projeto foi a formação de recursos humanos aptos a trabalharem na área de desenvolvimento dirigido a modelos e mineração de dados. A presença de pessoal capacitado a atuar nessa área é de fundamental importância para o avanço da mesma, pois trata-se de áreas relativamente novas e que apresentam poucos

AGRADECIMENTOS

Ao CNPq pela bolsa de Iniciação Científica.
A meus pais pelo constante apoio quando eu precisei.
A meu Professor Orientador Franklin Ramalho pelo acompanhamento das atividades e orientações no andamento do trabalho
A Andreza Vieira por sua dedicação no cumprimento de atividades e por.
A Anderson Ledo por ter me ajudado em algumas atividades e por estar presente quando eu precisei de sua ajuda.
A todos meus demais colegas de graduação e do Grupo de métodos formais.

REFERÊNCIAS BIBLIOGRÁFICAS

ABOISSA. **Girassol.** Disponível em: <http\\www.aboissa.com.br/giralssol/compq.htm> Acesso em: 12 ago. 2003.

AGRO-FAUNA. **Girassol.** Disponível em: <http\\www.cereaisdamo.com.br/culturas.php> Acesso em: 21 nov. 2003.

MACHADO, J.C.V. **Reologia, viscometria e escoamento**. SEREC/CEN/NOR, 1996. 86p.

MARKLEY, K.S.; **A indústria de óleos, ceras e gorduras vegetais no polígono das secas**. Fortaleza: Banco do Nordeste, 1995. 45p.

[1] - Goebel, M.; Gruenwald, L. *A survey of data mining and knowledge discovery software tools*. ACM IGKDD, San Diego, v. 1, n. 1, p. 20-33, 1999
[2] – Kleppe, A.; Warmer, J.; Bast, W. *MDA explained: The model-driven architecture: practice and promise.* Object-Technology Series. Addison-Wesley, 2003.
[3] – Ian H. WITTEN; EIBE FRANK; *Data Mining. Practical Machine Learning Tools and Techniques*. Morgan-Kauffman, 2005.
[4] – Metamodelo da UML 2.1.2
http://www.omg.org/spec/UML/20061001/Superstructure.cmof. Acesso em: 17 de fevereiro de 2009.
[5] – Weka – http://www.cs.waikato.ac.nz/ml/weka/. Acesso em: 17 de fevereiro de 2009.
[6]–Datasets do Weka (Bases de dados que serviram para aprendizado com a ferramenta de mineração): http://www.dsc.ufcg.edu.br/~sampaio/cursos/2008.2/PosGraduacao/MineracaoDeDados /Miscelanea/datasets/ . Acesso em: 17 de fevereiro de 2009.
[7] – *Document Object Model* (DOM):
http://java.sun.com/j2se/1.4.2/docs/api/org/w3c/dom/package-summary.html. Acesso em: 18 de fevereiro de 2009.
[8] *Simple API for XML* (SAX) – http://www.saxproject.org/ Acesso em: 18 de fevereiro de 2009.
[9] JDom: www.jdom.org/ Acesso em: 18 de fevereiro de 2009.
[10] *XPath* - http://www.w3.org/TR/xpath Acesso em: 18 de fevereiro de 2009.
[11] *Model Driven Architecture (MDA)* - http://www.omg.org/mda/
[12] Apostila de Banco de Dados do Professor Oswaldo Kotaro - http://www.ime.usp.br/~jef/apostila.pdf Acesso em: 18 de fevereiro de 2009.
[13] Materiais de Aula do professor Cláudio Batista: http://www.dsc.ufcg.edu.br/~baptista/cursos/BDadosI/index.htm. Acesso em: 18 de fevereiro de 2009.

[14] Guia de Consulta Rápida – UML – NOVATEC: http://www.novatec.com.br/guias/uml2/. Acesso em: 18 de fevereiro de 2009.
[15] Site da Disciplina de Sistemas de Informação 2 – Franklin Ramalho: http://www.dsc.ufcg.edu.br/~franklin/disciplinas/2008-2/SI2/index.php/Main/HomePage
Acesso em: 18 de fevereiro de 2009.
[16] *Tanagra – Data Mining Software*: http://eric.univ-lyon2.fr/~ricco/tanagra. Acesso em: 18 de fevereiro de 2009.
[17] *Orange – Data Mining Software*: www.ailab.si/orange Acesso em: 18 de fevereiro de 2009.
[18] KDB2000 – *Data Mining Software*: http://www.di.uniba.it/~malerba/software/kdb2000 Acesso em: 18 de fevereiro de 2009.
[19] XML – *Extensible Markgroup Language* : http://www.w3.org/XML/ Acesso em: 18 de fevereiro de 2009.
[20] Object Management Group (OMG) - http://www.omg.org/
[21] Kleppe, A.; Warmer, J.; Bast, W. *MDA explained: The model-driven architecture: practice and promise.* Object-Technology Series. Addison-Wesley, 2003.
[22] Alexander, C.; Ishikawa, S.; Silverstein, M. *A Pattern Language: Towns, Buildings, Construction*. New York, NY. Oxford University Press, 1977.
[23] Gamma, E., Helm, R., Johnson, R.; Vlissides, J. *Design patterns: Elements of reusable object-oriented software*. Addison Wesley, 1995.
[24] France, R. B.; Kim, D.-K.; Ghosh, S.; Song, E. *A UML-Based Pattern Specification Technique.* IEEE Trans. Software Eng., vol. 30, no. 3, Mar. 2004.
Disponível em: http://www.cs.colostate.edu/~georg/aspectsPub/TSE04-RBML.pdf. Acesso em: 09 de outubro de 2008.
[25] Dong, J.; Yang, S. *Visualizing Design Patterns With A UML Profile*. *In Proceedings of the IEEE Symposium on Visual/Multimedia Languages* (VL), 2003. Disponível em: http://www.utdallas.edu/~syang/vl03.pdf. Acesso em: 09 de outubro de 2008.
[26] Ackermann, J. *Formal Description of OCL Specification Patterns for Behavioral Specification of Software Components*. *In Proceedings of the Workshop on Tool Support for OCL and Related Formalisms, Technical Report.* LGL-REPORT-2005-001, 2005. Disponível em: http://lgl.epfl.ch/members/baar/oclwsAtModels05/pap5Published.pdf. Acesso em: 22 de outubro de 2008.
[27] Bézivin, J.; Jouault, F.; Paliès, J. *Towards Model Transformation Design Patterns*.
ATLAS Group (INRIA & LINA, *University of Nantes). In Proceedings of the First European Workshop on Model Transformations* (EWMT), 2005. Disponível em: http://www.sciences.univ-nantes.fr/lina/atl/www/papers/DesignPatterns05.pdf. Acesso em: 05 de novembro de 2008.
[28] Iacob, M.-E.; Steen, M. W. A.; Heerink, L. *Reusable Model Transformation Patterns*. Freeband. In Proceedings of the Workshop on Models and Model-driven Methods for Enterprise Computing (3M4EC), 2008.
[29] Mili, H.; Pachet, F. *Patterns for Metamodeling*. 1998. Disponível em: http://citeseer.nj.nec.com/44259.html. Acesso em: 09 de outubro de 2008.
[30] Goldberg, A.; Robson, D. Smalltalk-80 The Language. Addison- Wesley Publishing Company, 1989.
[31] Han, J.; Kamber, M. Data Mining: Concepts and Techniques.Academic Press, 2001.
[32] Michail, A. *Data Mining Library Reuse Patterns using Generalized* Association Rules. In Proceedings of the 22nd International Conference on Software Engineering,
pages 167- 176, 2000.