VII CONGRESSO DE INICIAÇÃO CIENTÍFICA DA UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

Pró-Reitoria de Pesquisa e Extensão

PIBIC/CNPq/UFCG-2010

# *Desenvolvimento e Extensão de uma Ferramenta de Mineração de Dados para Descoberta de Padrões em Meta-Modelos MOF*

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

## RESUMO

MDA (*Model-Driven Architecture*) tem como principal objetivo deslocar o esforço e tempo durante o ciclo de vida de desenvolvimento de um software das tarefas de implementação e testes para tarefas de modelagem, meta-modelagem e transformações de modelos. Por outro lado, um padrão no âmbito da computação visa de uma forma geral apresentar regras, diretrizes, processos, serviços e correlatos que contribuam para que haja a reutilização de soluções analisadas e aprovadas em problemas usuais no desenvolvimento de aplicações, evitando duplicação de trabalho. Padrões têm sido identificados e aplicados em diversas atividades dentro do processo de desenvolvimento de software, como, por exemplo, na fase de projeto e testes. A identificação e validação de padrões em meta-modelos MOF são promissoras no auxílio ao desenvolvimento e entendimento de meta-modelos, fundamentais para a especificação de transformações, que por sua vez, também podem ser alvos da identificação e adoção de padrões. Uma ferramenta voltada para a descoberta de padrões em MDA (especificamente em meta-modelos MOF) através de algumas técnicas de mineração de dados foi extendida durante esse ano de pesquisa. Como resultado do trabalho, foram desenvolvidos dois novos módulos, assim como a implementação da interface gráfica e respectivos testes. De maneira geral, nosso sistema realiza, dentre outros processos, a extração de dados de meta-modelos MOF e a aplicação de algoritmos de mineração de dados sobre tais dados para obter-se a descoberta de padrões e métricas para o desenvolvimento de novos meta-modelos.

**Palavras-chave:** MDA, Padrões, Mineração de dados, Desenvolvimento Dirigido por Modelos.

# Development and extension of a *Data-Mining* Tool to realize patterns discovery under MOF Meta-Models

## ABSTRACT

MDA (Model-Driven Architecture) has as the main goal to change the efforts and time in software development to the tasks of modeling, meta-modeling and model transformation instead of implementation and testing. Moreover, a pattern within computing context aims to provide general rules, guidelines, processes, and related services to help ensure that there is reuse of solutions reviewed and approved in the usual problems in application development, avoiding duplication of work. Patterns have been identified and applied in various activities within the process of software development, for example, in the design and testing phase. The identification and validation of standards in MOF meta-models are promising to help the development and understanding of how meta-models are fundamental to the specification of transformations, which in turn, may also be targets of the identification and adoption of standards. As a result of work done during this year of research, were developed two new modules, as long as the implementaion of a graphic user interface and respective tests of a tool aimed to discover patterns in MDA (specifically MOF meta-models) aided by some data mining techniques. In general, our system performs, among other processes, extraction of data from MOF meta-models and application of data mining algorithms on such data to obtain the discovery of standards and metrics for the development of new meta-models .

**Keywords:** MDA, Patterns, Data Mining, Model-Drivend Development

[1] Aluno de graduação em Ciência da Computação, Unidade Acadêmica de Sistemas e Computação, UFCG, Campina Grande , PB, E-mail: sousaleal.fabio@gmail.com

[2] Professor Doutor do curso de Ciência da Computação, Unidade Acadêmica de Sistemas e Computação, UFCG, Campina Grande , PB, E-mail: franklin@dsc.ufcg.edu.br

## INTRODUÇÃO

Várias metodologias de engenharia de software descrevem processos através de informações presentes nos requisitos, arquitetura, projeto, implementação e testes. Manter estas informações pode ser um requisito do cliente ou uma garantia de certificação de qualidade do software. Adicionalmente, ela torna-se imprescindível em vários cenários, como quando da extensão do time de desenvolvimento ou da migração de tecnologias envolvidas. Em suma, a documentação do software torna-se vital para garantir uma eficiente comunicação entre os times de desenvolvimento, servindotambém como contratos entre eles.

(DDM, 2008) é uma técnica de engenharia de software consistindo da aplicação de modelos e tecnologias de modelos com o intuito de simplificar, formalizar e automatizar as várias atividades compreendidas durante o ciclo de vida do software.

Mineração de dados é a atividade de descoberta de padrões interessantes a partir de uma grande quantidade de dados armazenados em diferentes repositórios de dados (HAN, KANBER, 2001). 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 WWW.

O objetivo principal desse trabalho foi continuar o desenvolvimento de uma ferramenta voltada para a descoberta de padrões em MDA (especificamente em meta-modelos MOF) através de algumas técnicas de mineração de dados. Alcançamos, ao longo desse ano de pesquisa, o desenvolvimento de dois novos módulos: o primeiro, voltado à aplicação de algoritmos de mineração de dados, e o segundo, voltado à identificação dos padrões identificados pela ferramenta em meta-modelos genéricos. Desenvolveu-se, também, uma interface gráfica para integrar todos os módulos desenvolvidos.

### MDA (*Model-Driven Architecture*)

(OMG, 2008) tem desempenhado um papel fundamental dentro do contexto de DDM. Ela é responsável pela definição e especificação de uma série de padrões e toda uma infra-estrutura chamada de MDA (*Model-Driven Architecture*), segundo KLEPPE e BLANC.

MDA é uma realização de DDM que tem como principal objetivo 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. Para alcançar tal objetivo, MDA determina a elaboração de um conjunto de modelos padrões:

- CIM (*Computational Independent Model*): Captura a ontologia da organização e suas atividades independentemente dos requisitos computacionais envolvidos;

- PIM (*Platform Independent Model*): Captura os requisitos e projeto de software independentemente de qualquer plataforma de implementação;

- PSM (*Platform Specific Model*): Captura todos os detalhes de uma determinada plataforma na qual o software será implementado;

- Código: realização executável do software em desenvolvimento.

O CIM serve de entrada para a elaboração do PIM, que por sua vez serve de entrada para a elaboração do PSM, que, finalmente, serve de entrada para elaboração do código de implementação.

Outro importante elemento que desempenha papel fundamental dentro de MDA são as definições de transformações, que ditam como cada modelo de entrada (CIM, PIM, PSM ou código) pode ser transformado em um modelo de saída (CIM, PIM, PSM ou código). Estas definições são expressas através de uma linguagem de transformação que deve ser entendida por ferramentas nomeadas engenhos de transformação, capazes de total e automaticamente executá-las. Transformações entre modelos envolvem necessariamente transformações entre conceitos. Dentro da técnica MDA, tais conceitos são especificados através de meta-modelos, que são modelos que descrevem modelos. Assim, transformações entre modelos são especificadas através de transformações sobre meta-modelos.

Estes artefatos estão dispostos na Figura 1, que ilustra também os principais padrões de MDA. A aplicação é descrita por modelos em níveis de abstrações diferentes, ou seja, códigos, PIMs, PSMs ou CIMs. Os modelos que especificam esses modelos são chamados meta-modelos, e representam as regras para que modelos em uma certa linguagem sejam bem formados. Estes meta-modelos são descritos em uma linguagem provida pela OMG chamada MOF (*Meta Object Facility*) (OMG,2008). MOF é um padrão da OMG para MDD que foi originado na UML (OMG, 2008) (*Unified Modeling Language*) devido à necessidade de uma arquitetura de meta-modelagem para defini-la.

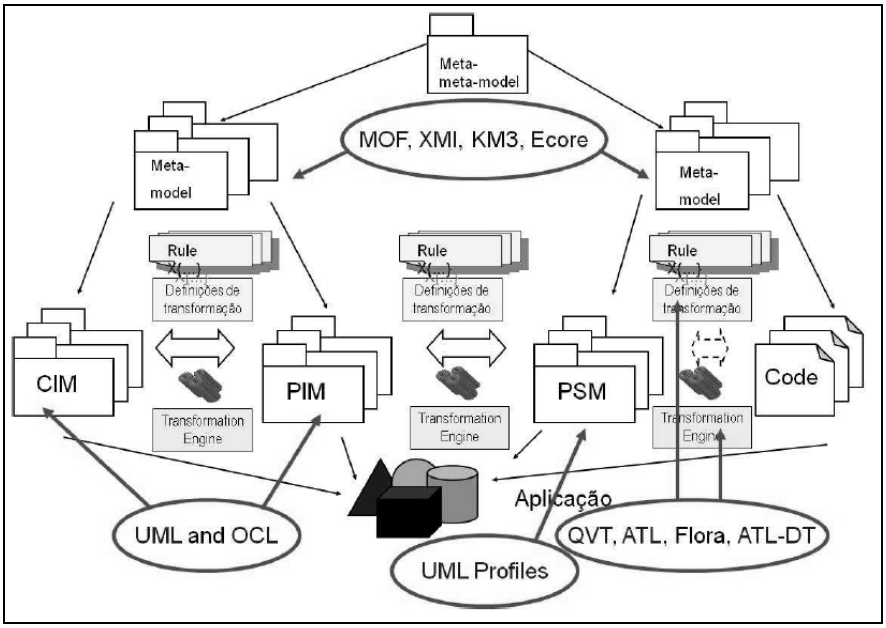



Figura 1 – Arquitetura MDA

Um dos pivores de MDA é UML (*Unified Modeling Language*) (OMG, 2008), proposta pela OMG e amplamente utilizada pela academia e mercado. Originalmente, UML surgiu como uma linguagem para escrita de artefatos de software no intuito de facilitar a sua visualização, especificação, construção e documentação. Por ser uma linguagem simples, expressiva, extensível, visual, e, principalmente, por ser provida de um vasto leque de ferramentas CASE, UML rapidamente se tornou um padrão de fato, tornando-se ainda mais difundida e popular. Em sua versão corrente, 2.0, UML torna-se ainda mais completa, expandindo seus recursos para viabilizar a especificação de sistemas em larga escala, sistemas de tempo real e DSLs (*Domain Specific Languages*) (STAHL et al., 2006). Adicionalmente, UML 2.0 tem se tornado uma linguagem mais precisa e completa com o surgimento de novos diagramas estruturais e comportamentais que viabilizam não apenas a especificação de artefatos de projeto de software, mas também a sua completa especificação, provida de detalhes rasteiros de implementação, que viabilizam a geração automática de código.

UML 2.0 contempla 13 (treze) diagramas estruturais e comportamentais, que, juntos, oferecem recursos para a especificação de requisitos do software (p.e., diagramas de casos de uso), projeto do software (p.e., diagramas de classes) e, até, implementação visto que são diagramas precisos e com recursos bem mais sofisticados para especificação de código em baixo nível (p.e., diagramas de seqüência e de atividades). Adicionalmente, UML oferece recursos que permitem estender a própria linguagem, como os perfis UML e a linguagem OCL (*Object Constraint Language*) (WARMER, KLEPPE, 2003). OCL foi inicialmente criada para especificar restrições sobre modelos UML, mas atualmente permite especificar qualquer computação dentro do contexto de UML. Isto tem tornado UML ainda mais poderosa. Juntas, UML e OCL formam o coração de MDA, pois servem de base para muitos outros padrões desta infra-estrutura, como MOF (*Meta Object Facility*) - padrão para especificar meta-modelos que reusa UML e OCL - e QVT (*Query, Views and Transformations*) - padrão para especificação de transformações, que reusa OCL.

**Padrões**

Um padrão no âmbito da computação visa de uma forma geral apresentar regras, diretrizes, processos, serviços e correlatos que contribuam para que haja a reutilização de soluções analisadas e aprovadas em problemas usuais no desenvolvimento de aplicações, evitando duplicação de trabalho. Em princípio, padrões buscam isolar no desenvolvimento as partes que são estáveis daquelas que são alteradas com freqüência. O resultado que se espera com isso é ter aplicações que são mais fáceis de manter, compreender e reusar.

Os padrões nascem a partir da necessidade de se identificar e registrar soluções para problemas que podem ocorrer diversas vezes em escopos diferentes, de forma que as soluções desenvolvidas e

conhecidas por especialistas sejam bem definidas, testadas e documentadas, tornando-se padrões por serem reutilizadas várias vezes em projetos diferentes e por terem eficácia comprovada. Um exemplo de padrão são os Padrões de Projeto, criados por Erich Gamma, John Vlissides, Ralph Jonhson e Richard Helm, os quais ficaram conhecidos como "*The Gang of Four*" (GAMMA,HELM, JONHSON, VLISSIDES, 1995). Estes padrões descrevem soluções para problemas recorrentes no desenvolvimento de sistemas orientados a objetos, de maneira que uma mesma solução possa ser usada para resolver problemas ocorridos em âmbitos diferentes.

Além do aspecto reutilizável dos padrões, temos ainda outras motivações para a sua utilização:

- Aprendizagem com a experiência de outras pessoas. Uma vez que problemas comuns em engenharia de software são identificados e catalogados, outras pessoas podem utilizar estas soluções testadas e bem documentadas;
- Vocabulário comum. Definição de um vocabulário comum para a discussão de problemas e soluções de projeto torna o sistema menos complexo, uma vez que será adotado um nível mais alto de abstração;
- Facilidade na documentação e compreensão. A compreensão de sistemas existentes fica mais fácil quando os padrões adotados são conhecidos, facilitando na manutenção da arquitetura do software;
- Refatoramento. Com o uso de padrões desde o início do projeto a necessidade de refatoramento pode diminuir.

Dentro da infra-estrutura de MDA, em seus diferentes níveis - seja meta-modelagem, modelagem ou transformação de modelos, a identificação de padrões também é promissora, uma vez que possui potencial para viabilizar: (1) a automatização de tarefas, diminuindo assim o tempo destinado para a mesma; (2) a melhor compreensão das atividades em questão, uma vez que lida com a abstração dos conceitos envolvidos na mesma; e (3) a diminuição da ocorrência de erros nos artefatos produzidos durante o processo de desenvolvimento de software. Por exemplo, a identificação de padrões em meta-modelos MOF (OMG,2008) é promissora no auxílio ao desenvolvimento e entendimento de meta-modelos, fundamentais para a especificação de transformações, que, por sua vez, também podem ser alvos da identificação e adoção de padrões.

**Mineração de Dados**

Mineração de dados é a atividade de descoberta de padrões interessantes a partir de uma grande quantidade de dados armazenados em diferentes repositórios de dados (HAN, KAMBER, 2001). Trata-se de uma área multi-disciplinar envolvendo não apenas bancos de dados e *data warehouse*, mas também estatística, aprendizagem de máquina, visualização de dados, recuperação de informação e computação de alta performance, dentre outras. 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 WWW.

O processo para mineração de dados é denominado como interativo, exploratório e cognitivo, o qual é formado pelas seguintes fases:

(1) Definição dos objetivos - Inicialmente, é necessário definir qual o tipo de conhecimento que se deseja descobrir e compreender o domínio da aplicação, bem como, escolher tarefas, técnicas e algoritmos para mineração dos dados;

(2) Seleção dos dados (*Selection*) - Um conjunto de dados relevantes ao usuário deve ser selecionado, o qual funcionará como alvo para a realização das descobertas;

(3) Limpeza de dados e pré-processamento (*Preprocessing*) - Esta fase consiste na eliminação de ruídos, dados estranhos ou inconsistentes, como também, na preparação dos dados, atuando na limpeza, transformação, integração e formatação;

(4) Transformação (*Transformation*) - Uma transformação ocorre nos dados pré- processados para que eles sejam armazenados de maneira adequada, isto é, no formato apropriado para aplicação de algoritmos de mineração;

(5) Mineração de dados (*Data mining*) - Nesta fase do processo alguns métodos são aplicados com o intuito de capturar padrões de interesse existentes nos dados;

(6) Interpretação e avaliação (*Interpretation/Evaluation*) - Com a finalidade de verificar as novas descobertas, os padrões identificados na fase de Mineração de dados passarão por uma interpretação e avaliação de acordo com algum critério do usuário;

(7) Implantação do conhecimento (*Knowledge*) - Por fim, o conhecimento adquirido no decorrer das etapas anteriores deverá ser documentado e encaminhado aos interessados ou implantado na performance do sistema.

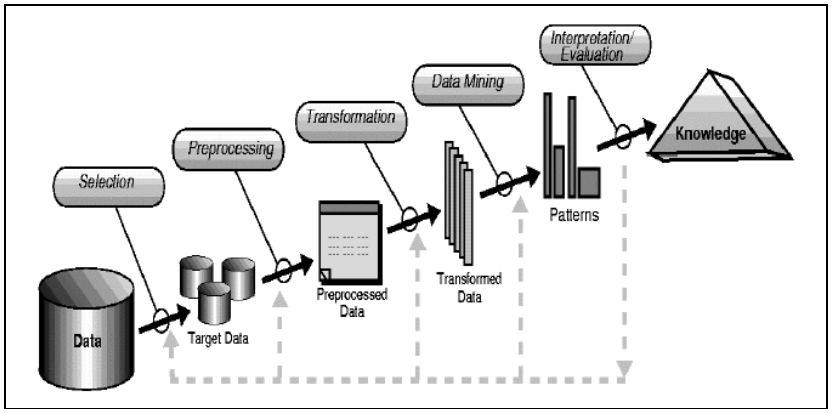


Figura 2 – Fases do processo para Mineração de dados

A mineração de dados é seguida de várias técnicas, metodologias e ferramentas que atuam em conjunto para nos permitir analisar um grande número de dados e decidir qual informação é mais relevante, nos favorecendo na tomada de decisões com maior rapidez e precisão. Uma ferramenta bem conceituada para auxiliar nesta abordagem é a *Waikato Environment for Knowledge Analysis* (WEKA), que consiste em uma coleção algoritmos de aprendizagem de máquina para tarefas de mineração de dados. A WEKA tem como objetivo principal prover facilidades para desenvolver técnicas de aprendizagem de máquina e aplicá-las em problemas reais de mineração de dados.

Assim, mineração de dados, acompanhada de suas técnicas, metodologias e ferramentas, se apresentaram como potenciais instrumentos para tarefa de descoberta de padrões dentro da infra-estrutura MDA, em seus diferentes níveis.

**Mineração de Meta-Modelos**

Para realizarmos a mineração de meta-modelos, é preciso, antes de tudo, que façamos com que o conteúdo do mesmo esteja acessível através de uma base de dados. Por esse motivo, durante um ano de pesquisa, foi feito o desenvolvimento de um protótipo de ferramenta que auxiliasse a mineração de meta-modelos. O objetivo da ferramenta desenvolvida ao longo do projeto de pesquisa é realizar a descoberta automática de padrões em meta-modelos MOF. O sistema dispõe de quatro módulos principais:

*Extractor*

Trata-se do módulo do sistema responsável pela leitura do XMI referente ao meta-modelo e extração das informações necessárias para a mineração. Estas informações extraídas são utilizadas para a geração dos arquivos para a mineração de dados através do módulo *Generator*.

Os arquivos utilizados na mineração de dados são, basicamente, arquivos texto constituídos por duas partes: a primeira, que contém meta-dados informando osatributos e seus respectivos tipos, e a segunda , que contém todos os valores para estes atributos, isto é, são os dados em si.

O módulo *Extractor* funciona de acordo com as seguintes etapas:

- Leitura do Metamodelo - XMI

  O arquivo XMI informado pelo usuário é lido logo após identificarmos os nomes dos atributos que deverão ser procurados e, na etapa a seguir, seus respectivos valores são obtidos.

- Obtenção dos valores dos atributos

  Esta etapa ocorre em paralelo com a etapa anterior, pois ao passo que os nomes dos atributos são lidos do arquivo XMI, seus respectivos valores são obtidos e armazenados em objetos Java. Nos casos em que não for encontrado nenhum valor para um determinado atributo, o valor deste atributo recebe seu *default*. Para que isto seja possível, foi criado no projeto um arquivo texto chamado .defaultValuesConfig.txt., o qual possui o conjunto de valores *default* para todos os atributos que os possuem.

*Generator*

Este módulo é responsável pela manipulação dos objetos de Java gerados no módulo descrito anteriormente para que torne os seus conteúdos disponíveis em um arquivo de texto. Este módulo do sistema realiza as etapas iniciais do processo de mineração de dados: seleção, pré-processamento e transformação dos dados. Uma vez que os dados são extraídos automaticamente, a partir do arquivo XMI, a etapa de pré-processamento da mineração de dados ocorre implicitamente, pois como os dados são gerados automaticamente, subentende-se que neles não há sujeira ou ruídos. Da mesma forma ocorre com a etapa de transformação dos dados, pois os dados já serão gerados no formato reconhecido pelo sistema.

*AlgorithmsManipulator*

Módulo responsável pela aplicação dos algoritmos de mineração de dados na base de dados obtida pelo módulo extractor. Os algoritmos a serem disponibilizados no sistema foram reutilizados da ferramenta Weka, que é uma ferramenta sob licença GPL, o que permite a cópia ou modificação legal do seu código-fonte. A decisão de reutilizar os algoritmos desta ferramenta no software ao invés de executá-los nela mesma foi no intuito de desenvolvermos um sistema auto-suficiente, que seja capaz de realizar as atividades necessárias sem a dependência de uma outra ferramenta. O módulo *AlgorithmsManipulator* funciona de acordo com as seguintes etapas:

- Executar algoritmos

  Uma vez que já temos os arquivos ARFF, o usuário pode executar os algoritmos disponíveis. Estes algoritmos são agrupados de acordo com a tarefa a qual eles pertencem. Para a descoberta de padrões exclusivamente no escopo de meta-modelos, são adotados apenas os algoritmos de associação.

- Gerar relatório com as regras encontradas

  Ao final da execução de qualquer um dos algoritmos, o sistema apresenta ao usuário um relatório com todas as regras induzidas a partir da base de dados. O usuário tem a opção de salvar este relatório em seu computador para uma análise futura dos resultados, se assim preferir. Vale destacar que o relatório gerado deve ser analisado pelo usuário de forma manual, ou seja, ele deve analisar as regras induzidas pelo algoritmo e interpretá-las. Cabe à interpretação do usuário determinar quais regras são desnecessárias e quais representam um padrão, de fato. É a partir dos padrões descobertos com a interpretação deste relatório que o módulo *Patterns*Applicator será implementado.

- Incluir novo algoritmo

  Além dos algoritmos já disponíveis no sistema, este módulo permite que outros desenvolvedores incluam os seus próprios algoritmos se achar que os existentes são insuficientes. Todo o código-fonte do sistema está estruturado de forma a facilitar esta tarefa.

*PatternsApplicator*

Conforme foi exposto no relatório parcial desse projeto, o módulo *PatternsApplicator* apresentava algumas dificuldades em sua implementação, pois o tempo para a implementação do mesmo foi subestimado. Assim como é explicado na sessão de Materiais e Métodos, esse módulo evoluiu para a criação de um sistema maior, desenvolvido conjuntamente com uma aluna de mestrado e que pode ser encontrado também em (TOOLSUPPORT, 2010).

Os padrões identificados no módulo anterior, uma vez que já forem catalogados, poderão ser adotados por desenvolvedores de meta-modelos em seus novos projetos. Por outro lado, caso os desenvolvedores desejem adotar estes padrões em projetos já existentes, eles podem utilizar o módulo *PatternsApplicator* para fazer uma análise prévia dos meta-modelos em busca de possíveis aplicações dos padrões e, em seguida, aplicar os padrões.Vale salientar que o sistema já realiza a etapa final do processo de mineração de dados, chamada de implantação do conhecimento. Isto ocorre no momento em que o padrão é aplicado no XMI referente ao meta-modelo, ou seja, o conhecimento (que neste caso são os padrões) é implantado. O sistema derivado do módulo *PatternsApplicator* funciona de acordo com as seguintes etapas:

- Ler XMI

  Inicialmente, o usuário deverá informar o arquivo XMI referente ao meta-modelo que se deseja analisar. Em seguida, será iniciada a sua leitura.

- Checar a aplicabilidade dos padrões ao meta-modelo

  Ao passo que o arquivo XMI está sendo lido, o sistema deve verificar se os padrões identificados podem ser aplicados.

- Gerar relatório com possíveis aplicações de padrão

  Ao final da checagem, o sistema deverá gerar um relatório detalhado informando quais são as partes do meta-modelo e seus respectivos padrões aplicáveis. O usuário terá a opção de salvar este relatório em seu computador para uma análise futura dos resultados em formato de texto plano (.txt), se assim preferir.

Uma visão geral da arquitetura do sistema é exibida na Figura 3. As partes do sistema que foram desenvolvidos durante esse ano de pesquisa são os blocos "Interface", "*Algorithms Manipulator*" e "*Patterns Applicator*".

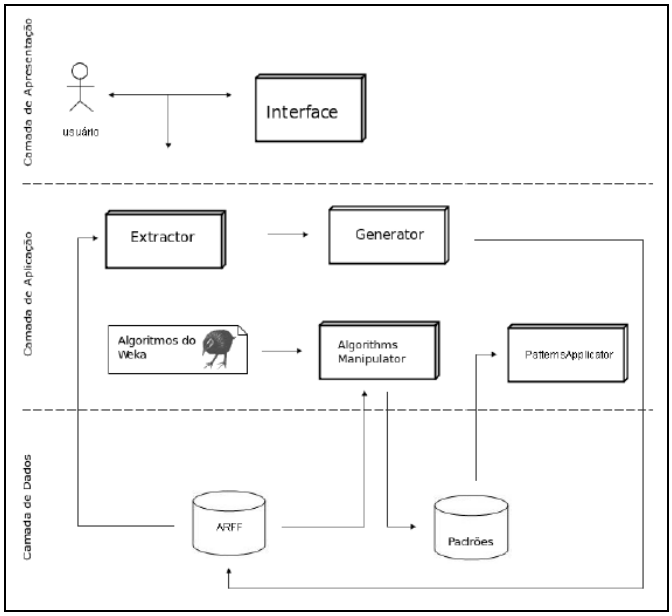



Figura 3 – Arquitetura do Sistema

## MATERIAL E MÉTODOS

Este trabalho foi desenvolvido no Grupo de Métodos Formais – (GMF, 2010) 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.

As atividades desempenhadas durante o projeto proposto foram:
1- Revisão bibliográfica sobre API's gráficas em Java - 1 mês;
2 - Estudo dos softwares de mineração de dados - 2 meses;
3 - Desenvolvimento do *AlgorithmsManipulator* - 2 meses;
4 - Desenvolvimento do módulo *PatternsApplicator* - 2 meses;
5 - Implementação da interface gráfica do software . 2 meses;
6 - Testes na ferramenta - 2 meses;
7 - Escrita de Relatórios técnicos e artigos - 2 meses.

| Atividade | Ago | Set | Out | Nov | Dez | Jan | Fev | Mar | Abr | Mai | Jun | Jul |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | x | | | | | | | | | | | |
| 2 | | x | x | | | | | | | | | |
| 3 | | | | x | x | | | | | | | |
| 4 | | | | | x | x | | | | | | |
| 5 | | | | | | | x | x | | | | |
| 6 | | | | | | | | | x | x | | |
| 7 | | | | | | | | | | | x | x |

Tabela 1 – Cronograma Proposto

Todas as sete atividades que foram listadas serão detalhadas devidamente nos próximos tópicos.

**Revisão bibliográfica sobre API's gráficas em Java**

Em Java temos três principais *frameworks* para construção de interfaces gráficas, que são, (AWT, 2010) , (Java Swing, 2010) e (SWT, 2010).

A seguir há o detalhamento técnico de cada tecnologia resultante da pesquisa realizada:

1) *Abstract Windows Toolkit (AWT)*

AWT é o *framework* para aplicações gráficas original de Java, que já vem com o *core* da linguagem. A vantagem de se trabalhar com AWT é que independentemente da versão de Java que se esteja rodando, é certo de haver uma compatibilidade do código escrito. Pelo fato de AWT ser um pacote nativo da linguagem, não há necessidade de instalar softwares adicionais no computador cliente.

AWT dá suporte ao tratamento de eventos, entretanto é relativamente limitado quanto ao número de componentes gráficos que dispõe para o programador. Componentes como tabelas, árvores e barras de progresso, por exemplo, não são suportados e esse é um dos pontos fracos do framework.

2) *Swing*

Java *Swing*, também conhecido como uma parte das *Java Foundation Classes* (JFC) é uma tentativa de solucionar algumas das deficiências de AWT. Com o *Swing*, a Sun Microsystems criou um *toolkit* bem estruturado, flexível e relativamente poderoso. Como resultado dessas vantagens, para compreender e adaptar-se bem com o *framework* o programador despende um bom tempo de desenvolvimento.

*Swing* foi construído baseado em AWT e usa o modelo baseado em eventos do mesmo. Todas as partes de *Swing* são também partes de AWT. As classes de suporte de *AWT* como *Colors*, *Images* e *Graphics* também foram reutilizadas pelo Swing.

3) *The Standard Widget Toolkit (SWT)*

SWT é um *toolkit* de baixo nível para a construção de aplicações gráficas de Java e é comparável a AWT. Há também o projeto *JFace*, que distribui um conjuntos de componentes de *SWT* de forma a tornar o desenvolvimento de GUI 's com SWT mais fácil.

Os desenvolvedores de *SWT* aprenderam dos projetos *AWT* e *Swing* e tentaram construir um *toolkit* que realizasse as mesmas tarefas que esses outros dois, mas sem as suas desvantagens.

Ao contrário de *Swing* e *AWT*, *SWT* não vem com o *core* de Java. Os pacotes desse *toolkit* devem ser instalados via bibliotecas separadas (JAR's) ou podem ser integrados à IDE Eclipse.

Como resultado da revisão bibliográfica à respeito das API's gráficas em Java, decidiu-se utilizar *SWT* e *SWING* para a construção da interface gráfica pelo fato de as mesmas apresentarem boa documentação e aprendizado relativamente rápido.

**Estudo dos softwares de mineração de dados**

A escolha do Weka como software de mineração de dados deu-se por ter sido o melhor avaliado em um estudo comparativo com o TANAGRA, KDB2000 e ORANGE, que são softwares de código aberto. Tal trabalho está disponível em (LEAL, 2008).

Durante os meses de Setembro e Outubro de 2009 foi analisado o código fonte do WEKA, que é o sistema de mineração de dados escolhido para ser integrado à ferramenta .

O código da ferramenta Weka pode ser dividido em quatro Módulos principais: *Explorer*, *Experimenter*, *Knowledge Flow* e *Simple CLI*. A ferramenta proposta na pesquisa atual faz uso apenas do módulo *Explorer*, e por esse motivo só serão apresentados os detalhes técnicos referentes a esse módulo.

O *Explorer* é a parte do Weka responsável pelas aplicações dos algoritmos de mineração de dados. É através dele que pode-se fazer o pré-processamento dos dados, aplicação algoritmos de classificação, clusterização, associação, seleção de atributos e visualização gráfica desses algoritmos.

O código do *Explorer* é baseado na criação de painéis (abas) que são habilitadas depois que o usuário entra com o arquivo ARFF para a mineração de seus atributos. O painel principal, chamado de *PreprocessPanel* é o responsável por efetuar as primeiras atividades de mineração, tais como a limpeza de dados e visualização dos atributos.

Os painéis laterais são os responsáveis pelas execuções dos algoritmos de mineração. Com isso, a integração do módulo *AlgorithmsManipulator* com a GUI em nossa ferramenta é realizada baseada na criação de um novo painel lateral para o módulo *Explorer* do Weka. Detalhes técnicos da implementação dos módulos serão mostrados detalhadamente nas próximas seções.

De início, a implementação do código parecia bastante complexa, pois o Weka é um sistema que abrange uma vasta gama de atividades relacionadas à mineração de dados e aprendizado de máquina.

No entanto, a comunidade de desenvolvedores do Weka apresentou-se bastante prestativa quando foi necessário ajuda. O uso de *Design Patterns* mostrados em (GAMMA et al., 1995) também fez com que o entendimento do código do Weka se desse de maneira mais fácil.

Como todo o código do projeto Weka é licenciado pela *GNU General Public License* (GPL), todo o trabalho resultante da integração com o mesmo será disponibilizado livremente, também sobre licença GPL para a comunidade.

**Desenvolvimento do *AlgorithmsManipulator***

O módulo *AlgorithmsManipulator* foi implementado durante os meses de Novembro e Dezembro de 2009. Um grande desafio para a implementação do *AlgorithmsManipulator* foi alcançar o entendimento dos algoritmos de mineração que estão presentes no módulo. Entender o funcionamento de tais algoritmos é essencial para que novos algoritmos possam surgir com o tempo. Juntamente com o progresso do desenvolvimento do módulo surgiram também algumas dúvidas que também foram rapidamente sanadas pela comunidade do software.

O módulo foi construído a partir de 3 propósitos principais, que são a execução de algoritmos, a geração de relatórios com as regras encontradas e a inclusão de novos algoritmos ao sistema. A seguir são dados os detalhes de como cada etapa desse módulo foi construída:

Execução de Algoritmos

Os algoritmos de associação utilizados encontram-se no pacote weka.associations. Todos eles foram desenvolvidos em Java e mais detalhes da implementação de cada um deles pode ser encontrado em (WEKA ASSOCIATIONS, 2010). Cada algoritmo de associação está de acordo com determinados parâmetros que devem ser passados na inicialização do algoritmo. Caso não seja passado nenhum argumento, a ferramenta inicia o processo com os valores default de cada algoritmo, que podem ser vistos nas documentações dos mesmos. Os algoritmos usados na ferramenta foram implementações em código Java dos algoritmos presentes em (HAN, KAMBER, 2001).

Geração do relatório com as regras encontradas

Ao final da execução de qualquer um dos algoritmos, a ferramenta desenvolvida ao longo desse ano de pesquisa apresenta ao usuário um relatório com todas as regras induzidas a partir da base de dados. O usuário tem a opção de salvar este relatório em seu computador para uma análise futura dos resultados, se assim preferir. Esses relatórios são gerados a partir das classes presentes no pacote *weka.core.logging*. As principais classes que foram usadas para gerar o relatório foram a *weka.core.logging.FileLogger* e *weka.core.logging.Logger*. A geração do relatório é feita a partir dos logs gerados pelas classes de execução dos algoritmos e poderão ser vistas na própria aba de algoritmos de associação, disponibilizada para o usuário em sua GUI.

Incluir novo Algoritmo

Além dos algoritmos já disponíveis no sistema, este módulo permite que outros desenvolvedores incluam os seus próprios algoritmos. Todo o código-fonte do sistema foi estruturado de forma a facilitar esta tarefa. Para o usuário incluir um novo algoritmo de associação o mesmo deve implementar a interface *Associator*, presente no pacote *weka.associations* e adicionar o mesmo algoritmo no referido pacote. Após isso, é necessário que o usuário informe à ferramenta da existência do seu novo algoritmo, adicionando o seu algoritmo à classe *AssociationsPanel*, do pacote *weka.gui.explorer*, seguindo o padrão utilizado pelos outros algoritmos.

**Desenvolvimento do *PatternsApplicator***

Como resultado da análise de algumas regras mostradas no *AlgorithmsManipulator*, foram definidos alguns padrões para meta-modelagem. O papel do módulo *PatternsApplicator* é realizar a leitura de um meta-modelo (em formato XMI, fornecido pelo usuário), e mostrar quais daqueles padrões poderiam ser aplicados no mesmo. Tais atividades foram implementadas com sucesso.

No entanto, durante a proposta do módulo não levou-se em consideração alguns obstáculos que poderiam impedir que o módulo ficasse pronto no tempo previsto da maneira que foi proposto (realizando a aplicação dos padrões). Tais obstáculos estão listados a seguir:

1) Algumas regras não são simples de serem aplicadas. No âmbito de padrões há certas regras que não são fáceis de serem aplicadas nem manualmente. Fazer com que todas as regras descobertas fossem aplicadas automaticamente pelo computador é uma atividade não trivial que despende muito tempo para ser desenvolvida.

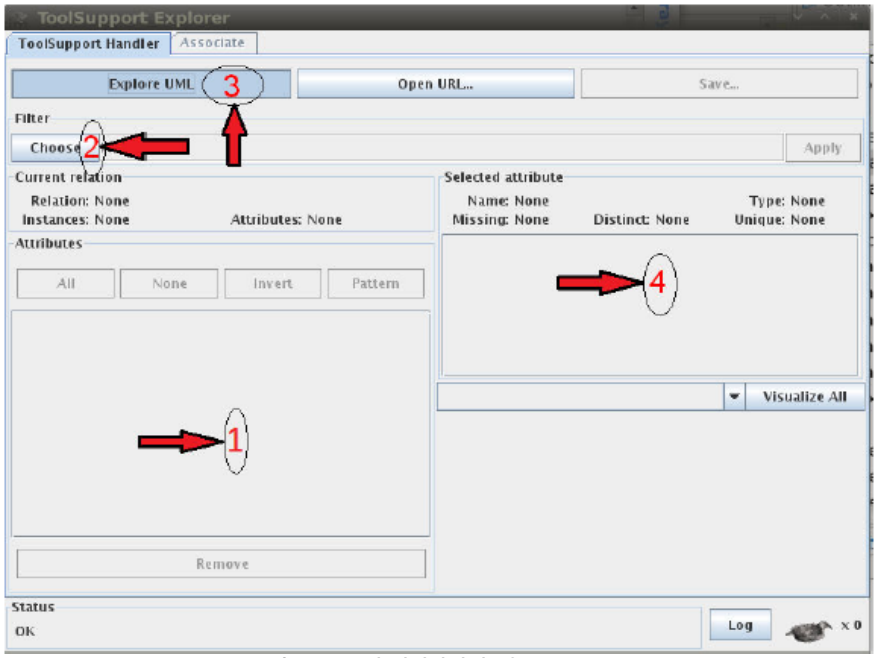
2) Tipos de XMI são diversos.

As especificações dos meta-modelos podem ser feitas em diversas ferramentas CASE, e essas ferramentas não possuem um único modelo de XMI. Fazer com que a ferramenta trabalhasse com mais de um tipo de XMI é uma atividade custosa que certamente necessitaria de mais tempo para sua conclusão.

3) Compilação das regras para as linguagens dos padrões
As aplicações das regras teriam que ser compiladas para as linguagens dos padrões, o que não é uma atividade trivial e que também despenderia muito tempo de desenvolvimento.
O desenvolvimento do *PatternsApplicator* evoluiu para outro sistema, que foi desenvolvido conjuntamente com uma aluna de mestrado do Departamento de Sistemas e Computação da UFCG e que encontra-se disponível também em (TOOLSUPPORT ,2010).

**Implementação da Interface Gráfica do Software**

Como resultado da revisão bibliográfica à respeito das API's gráficas em Java, decidiu-se utilizar *SWT* e *SWING* para a construção da interface gráfica. Nessa etapa da implementação do projeto conseguiu-se ainda fazer reuso de alguns elementos visuais da ferramenta Weka tais como o *GUIChooser* e o *FileEditor*. Alguns componentes da interface gráfica estão detalhados a seguir. Na Tela 1 é mostrada a tela inicial da ferramenta.



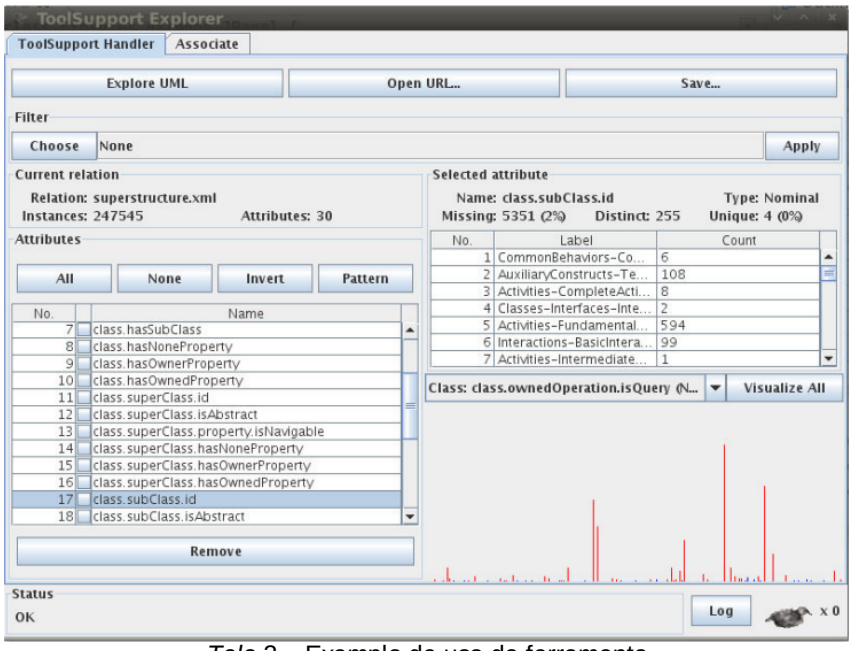

*Tela* 1 – Tela inicial da ferramenta

A Tela 1 mostra a tela com a qual o usuário terá seu primeiro contato com a ferramenta. Nela, podemos gerar o arquivo a ser minerado, como também podemos incluir qualquer .ARFF genérico, cabendo ao usuário escolher a melhor opção que enquadra-se ao seu caso.

A tela foi dividida em quatro partes para facilitar seu entendimento:

1) Visão responsável pelos atributos gerados no arquivo ARFF. Nessa parte da GUI o usuário pode escolher quais atributos serão utilizados na geração dos padrões com a utilização dos algoritmos disponibilizados na aba de associação. Além disso, o usuário pode selecionar determinados atributos e gerar um novo arquivo ARFF com os mesmos.
2) No botão indicado pelo número 2 o usuário pode aplicar alguns filtros aos atributos do ARFF. Tais filtros se classificam em *supervised* e *unsupervised*, e são úteis para fazer a "limpeza" dos dados. Como exemplo desses filtros temos o *Discretize*, que serve para discretizar atributos de valores numéricos e o *StringToNominal*, que serve para transformar atributos do tipo String em atributos nominais (enumerados no *header* do arquivo ARFF).
3) No botão indicado pelo número 3 o usuário faz a geração de um arquivo ARFF de exemplo baseado no meta-modelo da UML. Os atributos resultantes desse trabalho podem ser vistos na respectiva sessão da Tela 2.

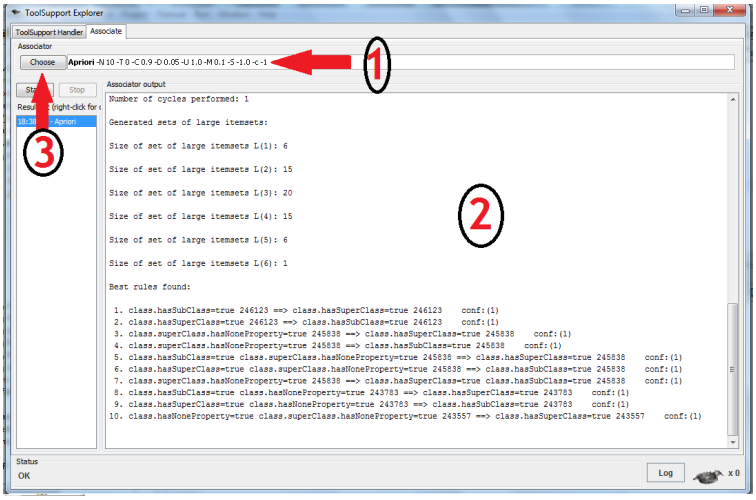
4) Na região de numero 4 são mostrados os valores dos atributos da propriedade selecionada na região 1. Na Tela 2 temos os valores do atributo *class.subclass.id* (atributo integrante do ARFF de exemplo gerado).




*Tela* 2 – Exemplo de uso da ferramenta

Na Tela 3 vemos como a GUI está integrada ao módulo *AlgorithmsManipulator*. Novamente, a imagem foi dividida nas seguintes seções:

1) Na zona de número 1 é aonde o usuário faz as configurações acerca do algoritmo de mineração que será executado. É nesse campo que configuramos parâmetros para obter um resultado mais otimizado para uma determinada base de dados. Cada algoritmo têm seus próprios parâmetros, e uma descrição detalhada de todos eles pode ser vista em (WEKA ASSOCIATIONS, 2010).
2) Na área de número 2, é aonde temos um relatório completo do algoritmo que foi executado. A saída nesse campo é feita em forma de texto plano, o que facilita seu arquivamento pelo usuário. Nele, é possível verificar aspectos como as regras encontradas, suas respectivas confianças, algoritmo aplicado e seus respectivos parâmetros e o tempo de execução do mesmo.
3) Botão de escolha do algoritmo de associação que será usado no *AlgorithmsManipulator*. Tal algoritmo será aplicado à base de dados que foi escolhida na tela da Tela *2*.




Tela 3 – GUI: *AlgorithmsManipulator*

**Testes na Ferramenta**

Foram feitos testes de unidades para cada um dos módulos desenvolvidos durante este ano de pesquisa. Tais testes usam o *framework* JUNIT e podem ser encontrados no pacote *java.test*. Apesar de não ter sido seguido à risca o processo de Test-Driven Development (BECK, 2003), aonde os testes tornam-se mais importantes do que a implementação, todos os módulos foram devidamente testados após sua implementação.

**Escrita de relatórios técnicos e artigos**

Além dos relatórios periódicos do projeto de iniciação científica, tivemos o artigo "*Applying Data Mining Techniques to Semi-Automatically Discover Guidelines for Metamodels" aceito no "Brazilian Workshop on Model-Driven Development*" - http://www.les.inf.puc-rio.br/opus/wbdsdm/, que é o primeiro evento focado em desenvolvimento dirigido a modelos do Brasil, o que atesta a relevância do trabalho realizado para a área.

## RESULTADOS E DISCUSSÃO

Como resultados da pesquisa dessa pesquisa de iniciação científica, obtiveram-se os seguintes resultados:

**Identificação de um conjunto de padrões para desenvolvimento de meta-modelos MOF;**

Com o auxílio da ferramenta desenvolvida foi possível gerar um vasto conjunto de regras para meta-modelagem. Tais regras foram analisadas manualmente e constituiram um conjunto de padrões obtidos de maneira automática. O catálogo de padrões que é citado ainda na sessão de resultados é constituído de padrões descobertos de maneira automática (através da ferramenta) e aqueles propostos por análise manual.

**Conclusão de uma ferramenta para descoberta de padrões em meta-modelos MOF;**

A ferramenta, cujos módulos estão devidamente detalhados na seção de materiais e métodos, foi concluída e encontra-se disponível para download em (TOOLSUPPORT, 2010). Toda a documentação e guia para o usuário está, também, disponível no repositório do software.

**Desenvolvimento de uma ferramenta capaz de fazer a validação de meta-modelos;**

Uma segunda ferramenta desenvolvida conjuntamente com uma aluna de mestrado, com o intuito de validar meta-modelos com relação aos padrões descobertos, também foi concluída e está disponível para download também em (TOOLSUPPORT, 2010). A construção dessa ferramenta foi importante, pois auxilia na identificação de padrões que podem ser aplicados a meta-modelos submetidos à mesma.

**Publicação em meios científicos**

Como resultado do trabalho realizado com auxílio da ferramenta desenvolvida, obteve-se a publicação intitulada "*Applying Data Mining Techniques to Semi-Automatically Discover Guidelines for Metamodels" aceito no "Brazilian Workshop on Model-Driven Development*" - http://www.les.inf.puc-rio.br/opus/wbdsdm/.

**Proposição de um catálogo de padrões formais a serem seguidos na construção e validação de novos meta-modelos;**

Com as diretrizes propostas pela ferramenta de mineração de dados desenvolvida nessa pesquisa foi viabilizada a criação de um catálogo de padrões a serem seguidos durante o desenvolvimento de novos meta-modelos MOF.

O catálogo de padrões está disponível em (VIEIRA, 2010), assim como algumas dicas quanto ao uso dos mesmos. A avaliação de tais padrões também está exposta no mesmo trabalho.

**Recursos humanos capacitados para trabalhar com pesquisas e desenvolvimento na área de Mineração de Dados, DDM, MDA e áreas afins.**

Um outro aspecto positivo desse projeto foi a formação de recursos humanos aptos a trabalhar na área de desenvolvimento dirigido a modelos e mineração de dados. A presença de pessoal capacitado a atuar nessas áreas é de fundamental importância para o avanço das mesmas, pois se tratam-se de áreas relativamente novas e que apresentam poucos pesquisadores envolvidos.

## CONCLUSÃO

Os resultados apresentados durante esse ano de pesquisa são, de fato, os que eram esperados na proposta inicial do projeto. Pode-se dizer que os objetivos do projeto foram cumpridos com sucesso e que o trabalho possui um bom grau de relevância para os envolvidos com desenvolvimento dirigido por modelos e áreas afins. A ferramenta que foi proposta inicialmente, assim como sua documentação, está disponível para download em (TOOLSUPPORT, 2010).

## AGRADECIMENTOS

Ao CNPq pelo financiamento do projeto e pela concessão da bolsa PIBIC;
Ao Grupo de Métodos formais pelo suporte físico necessário para o desenvolvimento da pesquisa.
A todos os colegas do Laboratório e ao professor orientador por viabilizar a pesquisa.

## REFERÊNCIAS BIBLIOGRÁFICAS

GOEBEL, M. ; GRUENWALD, L. ***A survey of data mining and knowledge discovery software tools*** IGKDD, San Diego, v. 1, n.1, 1999. 33p.

KOTARO, O. **Apostila de Banco de Dados.** Disponível em:< http://www.ime.usp.br/~jef/apostila.pdf > Acesso em: 18 de fevereiro de 2009.

DDM. Pastor, O.: Model-Driven Development: The OO-Method Aproach.
Presentation at UFPE, Recife, Brasil August 2008.

HAN, J.; KAMBER, M. ***Data Mining: Concepts and Techniques***. Academic Press, 2001.

OMG*.* ***Object Management Group*****.** Disponível em: http://www.omg.org/ Acesso em: 18 de fevereiro de 2008.

KLEPPE, A; WARMER, J; BAST, W. ***MDA explained: The model-driven architecture: practice and promise.*** Object-Technology Series. Addison-Wesley, 2003. 192p.

BLANC, X. ***MDA en action (Ingénérie logicielle guidée par les modèles)***. Eyrolles. 2005.

STAHL, T., VOELTER, M. and CZARNEK, K. ***Model-Driven Software Development: Technology, Engineering, Management*****.** Wiley, 2006.

GAMMA, E., HELM, R., JONHSON, R.; VLISSIDES, J.; ***Design patterns: Elements of reusable object-oriented software*****.** Addison Wesley, 1995.

WEKA. ***Weka Data Mining Software*****.** Disponível em: www.cs.waikato.ac.nz/ml/weka/. Acesso em: 17 de fevereiro de 2009.

TOOLSUPPORT. **ToolSupport Repository.** Disponível em http://code.google.com/p/toolsupport/. Acesso em: 17 de fevereiro de 2009.

AWT: ***The Abstract Windows Toolkit.*** Disponível em: http://java.sun.com/j2se/1.4.2/docs/api/java/awt/package-summary.html. Acesso em 17 de fevereiro de 2010.

JAVA SWING. ***Java Swing***. Disponível em: http://java.sun.com/docs/books/tutorial/uiswing/. Acesso em 17 de fevereiro de 2010

SWT. ***The Standard Widget Toolkit*.** Disponível em: http://www.eclipse.org/swt/. Acesso em 17 de fevereiro de 2010

LEAL, F. **Estudo comparativo de Softwares de Mineração de Dados.** Disponível em: http://gmf.ufcg.edu.br/~fabiosl/pibic2008/Analise_weka_tanagra.pdf. Acesso em: 17 de fevereiro de 2009

BECK, K. ***Test-Driven Development by Example***. Addison Wesley, 2003.

VIEIRA, A. **Dissertação de mestrado de Andreza Vieira**. Disponível em http://www.gmf.ufcg.edu.br/~andreza/. Acesso em: 17 de fevereiro de 2010.

TANAGRA – ***Data Mining Software***. Disponível em: http://eric.univ-lyon2.fr/~ricco/tanagra. Acesso em: 18 de fevereiro de 2009.

ORANGE – ***Data Mining Software***. Disponível em: www.ailab.si/orange. Acesso em: 18 de fevereiro de 2009.

KDB2000 – ***Data Mining Software***. Disponível em: http://www.di.uniba.it/~malerba/software/kdb2000. Acesso em: 18 de fevereiro de 2009.

WEKA ASSOCIATIONS. ***Weka Associations Algorithms*.** Disponível em: http://weka.sourceforge.net/doc/weka/associations/package-summary.html. Acesso em 29 de julho de 2010.

GMF. ***Grupo de Métodos formais***.Disponível em: www.gmf.ufcg.edu.br. Acesso em 30 de julho de 2010.