Certificação da Utilização de Padrões de Projeto no Desenvolvimento Orientado a Modelos

Maria Cristina F. Gomes[1, 2], Maria Luiza M. Campos[1], Paulo F. Pires[1, 3], Linair M. Campos[4]

[1]Programa de Pós-graduação em Informática do IM/NCE - Universidade Federal do Rio de Janeiro

[2] Petróleo Brasileiro S/A – PETROBRAS

[3] Universidade Federal do Rio Grande do Norte

[4] Núcleo de Computação Eletrônica (NCE) – Universidade Federal do Rio de Janeiro

mariacfg@posgrad.nce.ufrj.br,{mluiza,paulopires,linair}@nce.ufrj.br

Abstract. *This research presents a design patterns use certification environment at all abstraction levels of business components model driven development. The solution is based on patterns representation through specific Unified Modeling Language (UML) profiles, on Meta Object Facility (MOF) compliant patterns repository and patterns manipulation through Java Metadata Interface (JMI).*

Resumo. *Este trabalho apresenta um ambiente de certificação do uso de padrões de projeto em todos os níveis de abstração do desenvolvimento de componentes de negócio orientado a modelos. A solução se baseia na representação de padrões através de perfis específicos da Unified Modeling Language (UML), num repositório de padrões compatível com Meta Object Facility (MOF), e na manipulação desses padrões através do Java Metadata Interface (JMI).*

1. Introdução

O modelo de desenvolvimento baseado em componentes (DBC) é o resultado do investimento constante na proposição de novos métodos de trabalho e tem correspondência direta com a evolução dos níveis de abstração no desenvolvimento de software. Essa evolução passou pelas linguagens de programação estruturada com vários níveis de modulação, pela teoria e a prática da orientação a objetos, pelas tecnologias de objetos distribuídos como CORBA e a linguagem Java e pelas tecnologias específicas para o modelo DBC, como CORBA 3.0 e EJB [Herzum e Sims 1999].

Recentemente, com a abordagem de desenvolvimento orientado a modelos, pretende-se melhorar a portabilidade e interoperabilidade de sistemas, através de representações em um nível mais alto de abstração e de forma independente de soluções tecnológicas específicas. Dessas representações, através de transformações sucessivas, seria possível automatizar a geração de representações para plataformas específicas,

permitindo que o reuso seja possível nos diferentes níveis do processo [Mellor 2004]. A Arquitetura Baseada em Modelos (*Model Driven Architecture* - MDA) é uma proposta do *Object Management Group* (OMG), que enfatiza a utilização de modelos no desenvolvimento de sistemas de software.

Padrões têm um papel fundamental nos processos de desenvolvimento baseados na MDA[1]. As transformações dos modelos, nos diversos níveis de abstração até a implementação do código, necessitam que esses modelos contenham detalhe suficiente para direcionar um aplicativo de software através de todo o processo. Esses detalhes podem ser incorporados através do uso de padrões, permitindo maior automação das transformações, trazendo enorme ganho de produtividade e qualidade. Por outro lado, no modelo DBC os componentes de software possuem responsabilidades de execução de serviços de acordo com a arquitetura tecnológica utilizada. Esses diversos componentes se inter-relacionam, e para que isso aconteça de forma eficiente e coesa, é necessário seguir as melhores práticas já testadas e consolidadas, o que significa a utilização de padrões de projeto adequados. Dessa forma conclui-se que garantir a utilização correta de padrões é um aspecto do desenvolvimento de software que merece bastante atenção e destaque, tanto no modelo DBC quanto no ambiente orientado a modelos e conseqüentemente em ambientes de desenvolvimento baseado em componentes orientado a modelos.

Esse trabalho se propõe a suprir a ausência de um ambiente de certificação aberto e flexível, definindo um processo e um sistema computacional para automatizar a validação dos modelos criados quanto à utilização correta de padrões de projeto envolvidos em todas as fases do ciclo de desenvolvimento. A abordagem proposta cobre tanto os padrões de projeto independentes de plataforma empregados nas fases de requisitos e especificação, quanto os padrões dependentes de plataforma, compatíveis com a tecnologia de componentes utilizada. Na fase de implementação o sistema computacional é específico para a plataforma *Java 2 Enterprise Edition* (J2EE), podendo, no entanto, ser estendido para outros tipos de plataformas apenas com a implementação de novos serviços de reconstrução de arquitetura específicos para essas plataformas e linguagens. Foi realizada uma validação preliminar do sistema computacional desenvolvido, em ambiente controlado, que serviu para avaliar o corretismo das suas funcionalidades e pretende-se conduzir uma validação mais detalhada em ambiente real.

O trabalho encontra-se estruturado em 5 seções. A seção 2 apresenta uma introdução à abordagem da MDA e de sua relação com a utilização e certificação de padrões de projeto. Na seção 3 discutimos o ambiente de certificação proposto com as características e especificação do sistema computacional desenvolvido. Na seção 4 apresentamos um cenário de utilização desse ambiente. Na seção 5, comparamos o ambiente de certificação proposto com trabalhos relacionados e apresentamos as conclusões da pesquisa, com suas contribuições, além de possíveis trabalhos futuros.

[1] **http://www.omg.org/mda/faq_mda.htm**

2. MDA, Componentes e Certificação de Padrões de Projeto

2.1. A Abordagem da MDA

A MDA define uma abordagem para o desenvolvimento de sistemas que utiliza a especificação das funcionalidades do sistema em um modelo de alto nível, para, através de transformações sucessivas, chegar à geração da implementação dessas funcionalidades em uma plataforma tecnológica específica. Para isso a MDA propõe uma arquitetura para modelagem que inclui um conjunto de diretrizes para especificações estruturadas e formais, expressas na forma de modelos [MDA 2003].

Modelos consistem de um conjunto de elementos que descrevem algo físico, abstrato ou uma realidade hipotética. Bons modelos servem como meio de comunicação de conhecimento, projetos e idéias. O conceito central da MDA é a criação de diferentes modelos em diferentes níveis de abstração e a interligação desses modelos através de transformações até a geração do código de implementação [Mellor 2004].

Nos processos de desenvolvimento tradicionais existe uma separação bem definida nos papéis dos modelos e das linguagens de programação, onde os modelos são apenas utilizados como artefatos de projeto. Na MDA os modelos fazem parte de forma direta do processo de produção, são altamente formalizados e adquirem características de artefatos de desenvolvimento [Frankel 2003]. No processo da MDA existem três etapas principais. Na primeira etapa construímos um modelo independente de plataforma (PIM). Na segunda etapa um PIM é transformado em um ou mais modelos de plataforma específica (PSM), dependendo das plataformas tecnológicas desejadas. A etapa final do desenvolvimento é a transformação de cada PSM em código.

Nesse contexto, onde diferentes tipos de modelos são manipulados em cada etapa do processo de desenvolvimento, torna-se necessário um mecanismo comum para definição de linguagens de modelagem, chamado genericamente de metamodelagem.

As camadas de metamodelagem da estrutura da MDA são baseadas em uma arquitetura de quatro camadas [Mellor 2004], e os padrões da OMG que possuem papéis na MDA também se relacionam de acordo com essa arquitetura. O *Meta Object Facility* (MOF) reside na camada M3, a mais alta da arquitetura de metadados. É por definição, o metametamodelo comum para especificação de metamodelos da OMG. É um padrão que especifica uma linguagem abstrata para definir linguagens de modelagem, como a *Unified Modeling Language* (UML), o *Common Warehouse Metamodel* (CWM), que residem na camada M2 da arquitetura de metadados, assim como para definir o próprio MOF [Poole 2001].

Para se definir uma linguagem de modelagem específica, podemos estender uma linguagem de modelagem existente (um metamodelo existente, como por exemplo, o metamodelo da UML) ou criar uma nova (um novo metamodelo). O conceito de perfil é um mecanismo de extensão definido como parte da UML, que estabelece uma forma específica de usar a UML. A MDA se utiliza muito desse mecanismo de extensão da UML, porque necessita dar suporte a diferentes níveis de abstração [Frankel 2003]. Um perfil define uma nova linguagem simplesmente reutilizando o metamodelo da UML. Um perfil é definido por um conjunto de estereótipos, um conjunto relacionado de restrições em *Object Constraint Language* (OCL) e um conjunto de etiquetas valoradas

(*tagged values*). Outra maneira de estender o metamodelo da UML é através do MOF, aconselhável em extensões mais complexas.

2.2. A Utilização de Padrões de Projeto na MDA

A MDA unificou o paradigma do desenvolvimento baseado em padrões com a construção de modelos [Blankers 2003] e enfatizou a utilização de padrões e sua transformação através dos diferentes níveis de abstração existentes (figura 1).

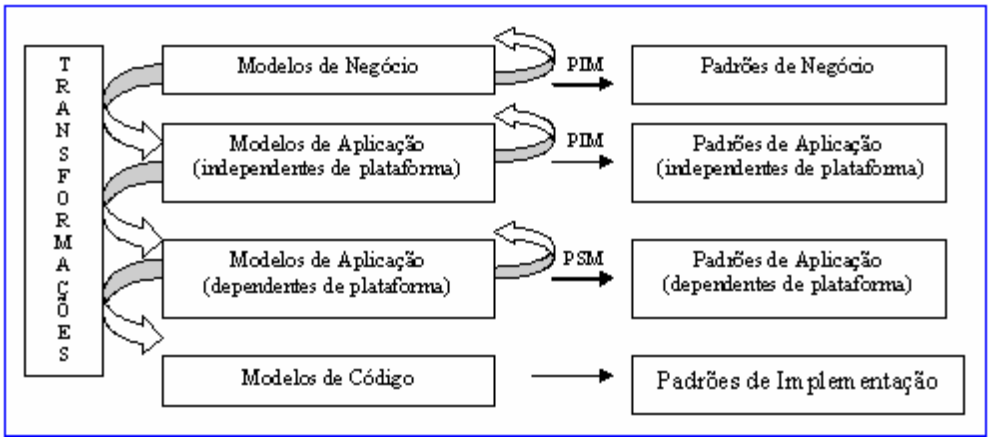

Figura 1. A utilização de padrões nos diversos níveis de abstração da MDA

Os padrões independentes de plataforma se aplicam em níveis de abstração em que ainda não estamos preocupados com a plataforma tecnológica que será utilizada. No nível dos modelos de negócio, Arlow e Neustad (2003) apresentaram uma proposta de utilização de padrões corporativos por eles chamada de padrões de arquétipos, com a descrição de alguns exemplos de padrões de arquétipos para representação de *Customer Relationship Management* (CRM), de Produto, de Inventário, de ordem (venda ou compra), de quantidade, de dinheiro, de regra, dentre outros. No nível de padrões de aplicação, utilizados no projeto lógico de uma aplicação ou componente de software, podem-se citar, entre outros, os padrões conhecidos como GoF [Gamma *et al* 1995]. Os padrões dependentes de plataforma são padrões a serem aplicados numa plataforma tecnológica específica. Cada plataforma tecnológica possui seu catálogo próprio de padrões. Esses padrões são utilizados para construções de componentes e aplicações robustas. Na plataforma J2EE temos o catálogo conhecido como Core J2EE Patterns[1].

2.3. A Certificação de Padrões de Projeto na MDA

No desenvolvimento orientado a modelos e baseado em padrões da MDA, existem padrões específicos para serem aplicados em cada nível de abstração. A formação desses níveis pode se dar de diversas formas, não existindo uma quantidade fixa de modelos de negócio e modelos de aplicação (independentes e dependentes de plataforma). A distribuição em níveis é determinada pelas características específicas de cada ambiente de desenvolvimento. A certificação da utilização de padrões nesse ambiente também necessita ser dividida em níveis, com etapas de certificação específicas para cada nível de abstração existente em um determinado processo de desenvolvimento. Cada etapa de

[1] http://java.sun.com/blueprints/corej2eepatterns/Patterns/

certificação garante a transformação correta para o próximo nível de abstração, até a implementação do código do componente que está sendo desenvolvido (figura 2).

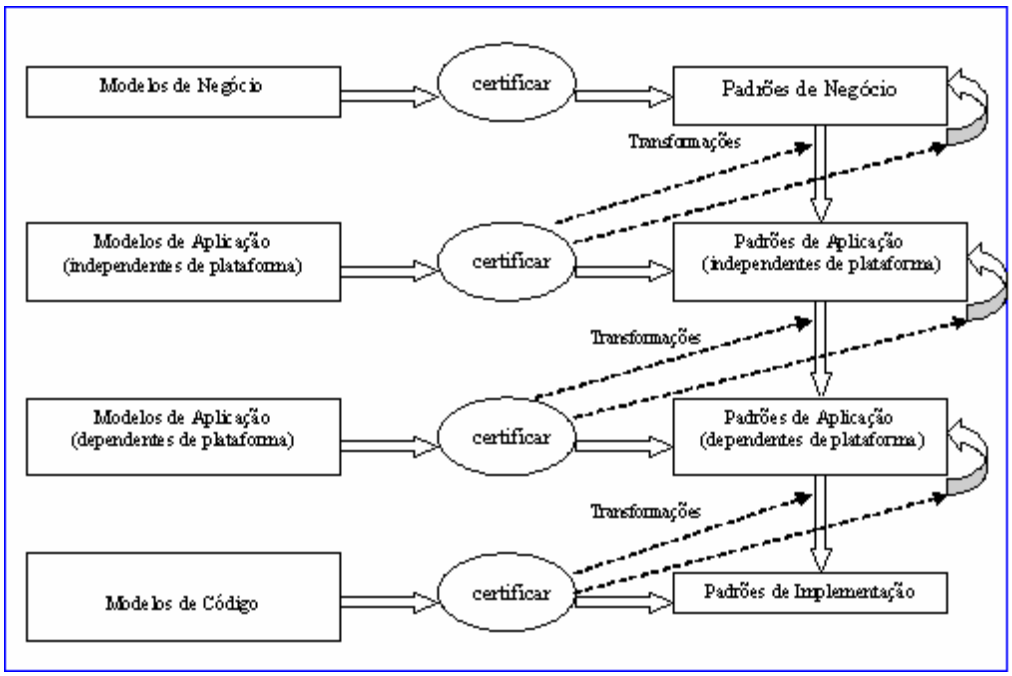

Figura 2. As certificações realizadas nos diversos níveis de abstração.

3. Ambiente de Certificação de Padrões de Projeto

Conforme descrito na seção anterior, no desenvolvimento orientado a modelos associado a uma abordagem como a da MDA, a certificação da utilização de padrões precisa ser executada em todas as etapas de transformação existentes no processo de desenvolvimento. Para atender esse requisito, esse trabalho apresenta uma proposta de um ambiente de certificação flexível, podendo ser utilizado em abordagens que contemplem diversos níveis de abstração.

3.1. Atividades e Papéis no Ambiente de Certificação

Através de pesquisas realizadas sobre certificação e sobre utilização de padrões junto a diversos projetos e na literatura, e com o levantamento das necessidades do ambiente de certificação proposto, foram identificadas as principais atividades a serem contempladas. Essas atividades foram classificadas segundo três macro-atividades, conforme representado na figura 3. Essas macro-atividades são:

- Administração de Repositório: contempla a manutenção do repositório dos padrões utilizados no ambiente de desenvolvimento;
- Reconstrução de Arquitetura: inclui a recuperação dos modelos de implementação a partir do código fonte dos componentes e aplicações;
- Análise de Modelos: envolve a comparação dos modelos dos componentes e aplicações com os modelos dos padrões utilizados e armazenados no repositório de padrões, cobrindo todos os níveis de abstração existentes no ciclo de desenvolvimento.

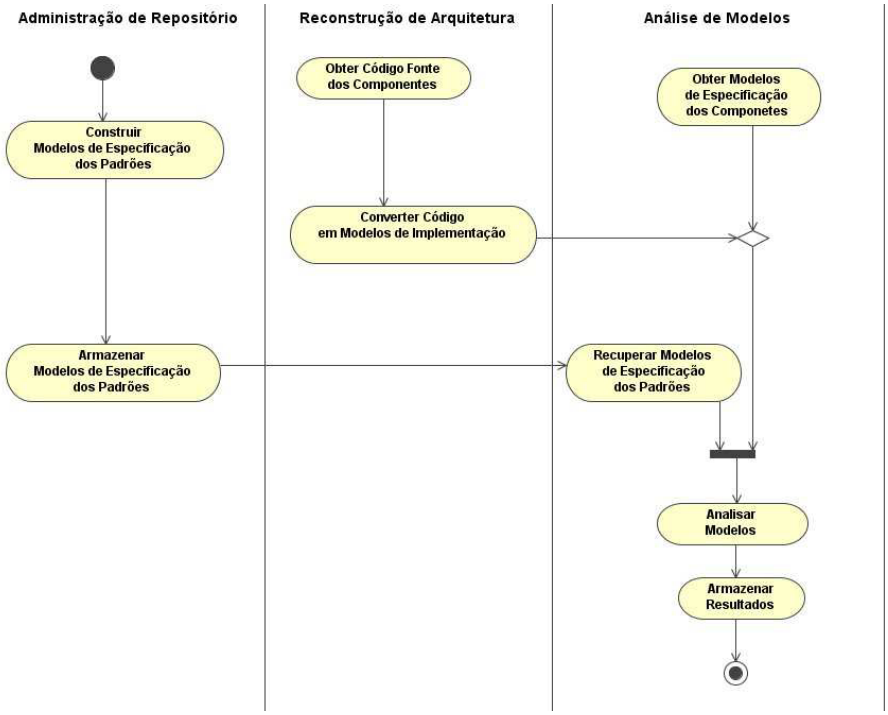

Figura 3. Diagrama de Atividades organizado por macro-atividades

Envolvidos no processo de certificação e associados a cada macro-atividade definida, foram identificados os seguintes papéis:

- Arquiteto de Negócios: responsável pela escolha dos padrões utilizados nos modelos de negócio e pela construção dos modelos das especificações desses padrões;
- Arquiteto de Aplicação: responsável pela escolha dos padrões utilizados nos modelos de aplicação dos componentes de software e pela construção dos modelos das especificações desses padrões;
- Administrador do Repositório: responsável pelo armazenamento e gerência dos modelos no repositório de padrões. É de sua responsabilidade manter atualizado o catálogo dos padrões utilizados no respectivo ambiente de desenvolvimento;
- Gerente de Reutilização: entre suas atribuições estão executar a conversão dos códigos dos componentes em modelos de implementação e analisar todos os modelos dos componentes desenvolvidos, antes da liberação desses componentes para consumo;
- Desenvolvedores: elaboram os modelos dos componentes e aplicações utilizando as especificações dos padrões existentes no repositório.

Esse modelo de atribuição de responsabilidades pode ser alterado de acordo com as características e necessidades do ambiente particular de cada organização.

3.2 A Arquitetura Conceitual do Ambiente de Certificação

A figura 4 apresenta a arquitetura do ambiente proposto, com os elementos do sistema computacional utilizado, necessário para automatizar as atividades de certificação.

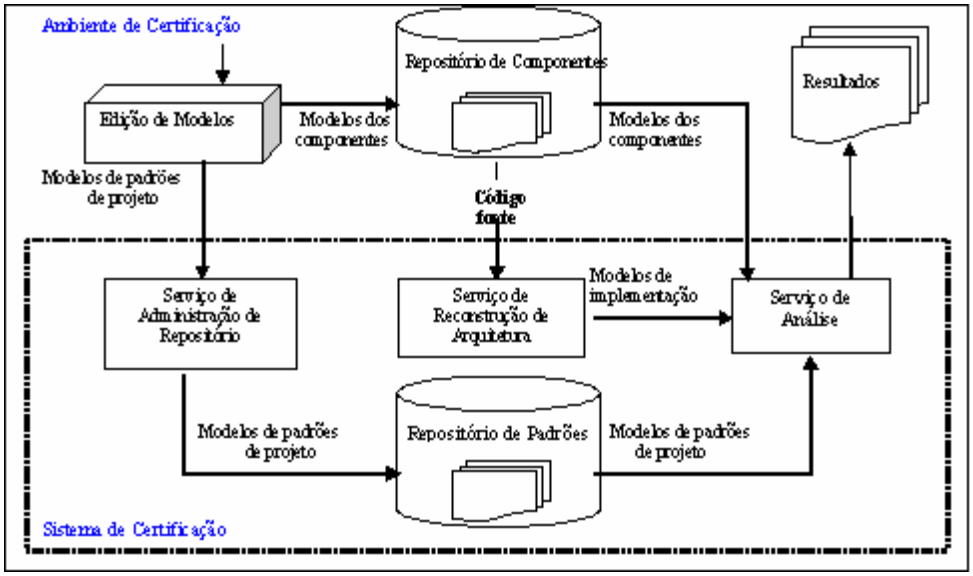

Figura 4. A Arquitetura conceitual do ambiente de certificação.

Nessa arquitetura identificamos funcionalidades existentes na abordagem da MDA que são utilizadas no ambiente de certificação proposto:

1. Edição de Modelos: utilizada na construção, através de ferramentas de modelagens específicas, dos modelos de representação dos padrões de projeto e dos modelos de desenvolvimento dos componentes e aplicações na linguagem de modelagem e formalismo de representação definido. Precisa exportar os modelos para um padrão de intercâmbio de modelos (metadados);
2. Repositório de Padrões: é utilizado para armazenar os modelos dos padrões utilizados no ambiente de desenvolvimento. Importa e exporta os modelos no padrão de intercâmbio de modelos definido. Os modelos são acessados no repositório utilizando uma forma comum de acesso a metadados;
3. Sistema de Certificação: sistema computacional de apoio, responsável pela automatização das atividades do ambiente de certificação. Sua estrutura é composta por três principais serviços. Esses serviços incorporam as características funcionais das macro-atividades identificadas no ambiente de certificação.
    a) Serviço de Administração de Repositório: responsável por manter o repositório de padrões. Os modelos de representação de padrões são importados para o repositório, e após a validação da formação correta dos modelos quanto ao formalismo utilizado. Os modelos em conformidade são aceitos e armazenados no repositório;
    b) Serviço de Reconstrução de Arquitetura: responsável pela análise e reconhecimento de código, realizando a conversão do código fonte dos componentes, obtidos do repositório de componentes, nos modelos de implementação (último nível de modelo de plataforma específica - PSM). Interpreta o código, identificando os elementos da linguagem de implementação e converte a definição desses elementos na linguagem de modelagem utilizada;

c) Serviço de Análise: responsável pela validação de modelos, analisa e certifica a correta construção dos modelos (negócio, aplicação e recuperados do código dos componentes e aplicações) com relação à utilização dos padrões de projeto. Utiliza uma forma comum de acesso a metadados para manipulação dos modelos em análise. Gera resultados com os diagnósticos da análise. Existem dois tipos de análise fornecidos pelo serviço:

- Análise dos Padrões Utilizados: Análise quanto à utilização correta dos padrões existentes no modelo. Compara a construção dos padrões identificados no modelo, com a representação desses padrões armazenados no repositório, certificando se estão em conformidade;
- Análise das Transformações de Padrões: Análise quanto às transformações de padrões existentes entre dois níveis de abstração. Nesse caso os padrões utilizados e identificados no modelo analisado, são comparados com as definições dos mapeamentos dos padrões do modelo original, certificando se as transformações entre os dois níveis estão em conformidade;

Na conexão entre os elementos da arquitetura necessitamos de mecanismos específicos de integração. Esses mecanismos estão relacionados com uma forma comum de acesso e padrão de intercâmbio de modelos. São utilizados para acesso, manipulação e intercâmbio de modelos entre as diferentes funcionalidades do ambiente.

3.3 Implementação do Ambiente de Certificação

Para a implementação da arquitetura do ambiente de certificação, foram feitas diversas escolhas tecnológicas com relação a: linguagem de modelagem e formalismo de representação, forma comum de acesso e padrão de intercâmbio de modelos, repositório de padrões e tecnologia de implementação utilizada.

Quanto à linguagem de modelagem e ao formalismo de representação, utilizados na construção dos modelos dos padrões e nos modelos de desenvolvimento dos componentes e aplicações, a escolha foi baseada nas abordagens de modelagem de papéis do *Role-Based Metamodeling Language Research Group*[1] [France *et al* 2003], e na utilização de mecanismos de extensão da UML através de perfis específicos [Dong e Yang 2003]. É necessário dispor de diversas informações sobre as características dos padrões nos seus modelos de representação. A certificação dos modelos de desenvolvimento dos componentes e aplicações que utilizam esses padrões é realizada de acordo com essas características. Da mesma forma, o sistema de certificação proposto precisa identificar os padrões existentes nos modelos analisados e todos os elementos que executam papéis nesses padrões. Assim, foi preciso acrescentar determinadas informações nos modelos de componentes e aplicações para permitir o rastreamento e identificação dos padrões presentes nesses modelos. A definição de um perfil UML específico para representação de padrões, bem como, um perfil UML específico para utilização de padrões nos modelos dos componentes e aplicações, permitem que todas as informações necessárias para o processo de certificação sejam incorporadas nos modelos através de estereótipos e etiquetas.

[1] http://www.cs.colostate.edu/~dkkim/ReuseResearch /main.html

Quanto à forma comum de acesso e padrão de intercâmbio de modelos, os padrões escolhidos foram o *Java Metadata Interface* (JMI) [JSR40 2002] e o *XML Metadata Interchange* (XMI) [XMI 2003]. Ambos são mapeamentos do MOF para a linguagem Java e XML respectivamente. Juntos, JMI e XMI fornecem uma completa estrutura de acesso e troca dinâmica de metadados independente de plataforma e de fabricante [Eckerson e Manes 1999].

O repositório de padrões escolhido foi o repositório de metamodelos MDR[1] compatível com MOF/JMI**.** Carrega qualquer metamodelo MOF e armazena suas instâncias (os metadados em conformidade com o metamodelo). Permite importação e exportação de metadados usando o padrão XMI, e manipulação desses metadados programaticamente usando interfaces Java através do JMI [Matula 2003].

A tecnologia Java [JSR244 2005] foi escolhida para a implementação do ambiente de certificação em conseqüência das escolhas do JMI e do repositório MDR. A tecnologia Java está bastante aderente à abordagem da MDA, com diversas propostas de padrões da OMG e do *Java Community Process* (JCP), relacionando o ambiente orientado a modelos da MDA com Java, a exemplo do JMI, metamodelos e perfis UML para Java e EJB [OMG 2004] [JSR026 2001].

Para o reconhecimento da linguagem Java, e conversão para UML, foram utilizadas a ferramenta ANTLR[2] e uma adaptação da classe *Modeller* da ferramenta *Java Roundtrip Engineering* (JavaRE)[3]. Essa adaptação consistiu na utilização do repositório MDR e das API's JMI do metamodelo da UML 1.4.

A utilização dessas decisões de implementação nos serviços que compõem o sistema de certificação se deu da seguinte forma:

Serviço de Administração do Repositório: Os modelos de representação de padrões são lidos pelo repositório MDR no padrão XMI. Os modelos em conformidade são armazenados no repositório pela representação de uma instância do metamodelo da UML na API JMI;

Serviço de Reconstrução de Arquitetura: Através de classes geradas pelo ANTLR, fazemos o reconhecimento da linguagem, identificamos os elementos a serem mapeados para UML e através da classe *Modeller*, adicionamos esses elementos em um atributo de representação de um modelo UML na API JMI. Após todo o código Java ser convertido, temos o modelo UML completo. Foram utilizados alguns recursos adicionados no J2SE 5.0 [JSR244 2005], para representação de determinados elementos de modelo no código implementado, como *Annotation* (para representar os estereótipos e etiquetas existentes no último nível de modelo de plataforma específica - PSM) e *Generics* (para representar as associações com multiplicidade *). O recurso de *Annotation* é resultado de uma nova proposta do JCP identificada como JSR-000175 *A Metadata Facility for the Java Programming Language* [JSR175 2004]**.**

Serviço de Análise de Modelos: Utiliza a API JMI para acessar os metadados, definições dos elementos de modelo contidos na representação JMI do modelo UML

[1] http://mdr.netbeans.org/
[2] http://www.antlr.org/
[3] http://javare.sourceforge.net/index.php

sendo analisado, identificando os padrões utilizados através dos estereótipos e etiquetas existentes. Esses padrões identificados são recuperados do repositório MDR, pela busca de uma instância do metamodelo da UML com o nome e tipo do padrão de projeto. Os modelos são comparados através das interfaces JMI, que representam os elementos existentes nos dois modelos (analisado e de representação do padrão);

4. Um Cenário de Utilização

O cenário de utilização corresponde a um exemplo de sistema de registro das vendas de uma rede de lojas de departamento. Nesse cenário temos o envolvimento dos papéis do gerente de reutilização e do desenvolvedor, descritos na seção 3.1. O desenvolver elabora manualmente todos os modelos do sistema, utilizando os estereótipos e etiquetas específicos do perfil UML para utilização de padrões, desenvolvido para o ambiente de certificação e o gerente de reutilização executa a certificação dos modelos a cada etapa de transformação no ciclo de desenvolvimento através do sistema computacional proposto.

A figura 5 apresenta o modelo de domínio correspondente ao PIM de nível mais alto de abstração, onde foi utilizado o perfil UML para utilização de padrões. Possui o estereótipo <<MNIP>> (*Modelo de Negócio Independente de Plataforma*) com a etiqueta *nivelModelo* igual a um, indicando existir mais um nível de abstração de modelo de negócio. Seus elementos de classe possuem o estereótipo <<Entidade>> indicando que representam um conceito de entidade do domínio do sistema, com a etiqueta *padroesAlvo*, indicando o padrão que será utilizado na transformação para o modelo do próximo nível de abstração.

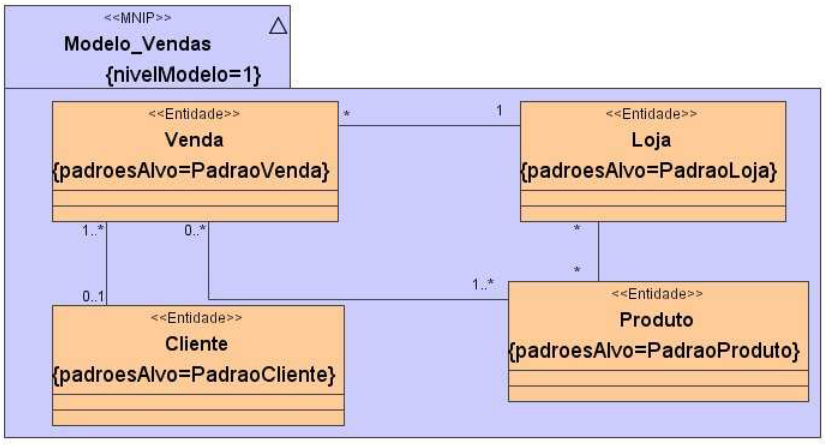

Figura 5. Modelo de Negócio Independente de Plataforma <<MNIP>> *Modelo de Vendas*

Transformamos esse modelo de negócio, aplicando exemplos simplificados de padrões corporativos de negócio, ou padrões de arquétipos de negócio, conforme denominação de Arlow e Neustadt (2003). O modelo inicial do exemplo do sistema de vendas é refinado através da substituição dos elementos Venda, Cliente, Loja e Produto, pelos padrões de negócio Venda, Cliente, Loja e Produto respectivamente.

A figura 6 apresenta o modelo do padrão de negócio Venda, que utiliza o perfil UML para representação de padrões, desenvolvido para o ambiente de certificação.

Possui o estereótipo <<PNIP>> (*Padrão de Negócio Independente de Plataforma*) com a etiqueta *nivelPadrao* igual a zero, indicando o último nível de abstração antes da transformação para um padrão <<PAIP>> (*Padrão de Aplicação Independente dePlataforma*). Todos os seus elementos com papel de classe possuem o estereótipo <<Entidade>>, indicando a representação do conceito de entidade do domínio do padrão, com a etiqueta *padroesAlvo* indicando os papéis de padrões que podem ser mapeados na transformação desses elementos para o próximo nível de abstração.

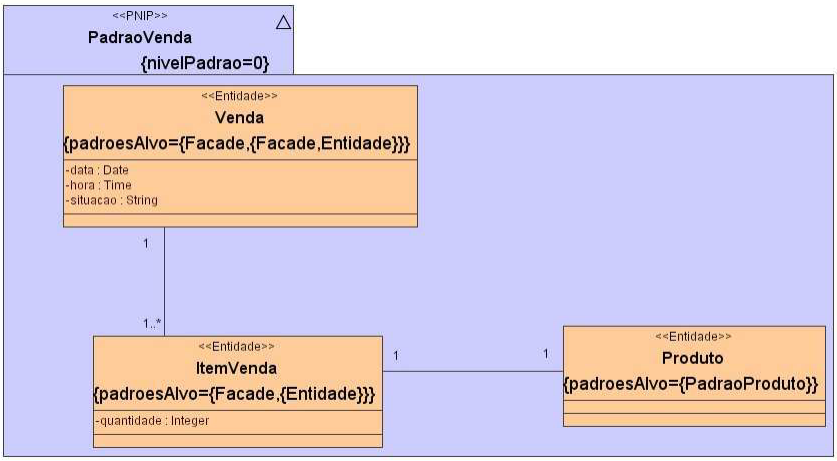

Figura 6. Padrão de Negócio Independente de Plataforma <<PNIP>> *PadrãoVenda*

No primeiro refinamento realizado no modelo de domínio do sistema, após aplicação dos padrões de arquétipos de negócio, geramos um novo modelo que não está sendo ilustrado, com estereótipo <<MNIP>>. PIM de segundo nível de abstração no ciclo de desenvolvimento corresponde a um modelo de negócio mais detalhado. Possui a etiqueta *nivelModelo* igual a zero, indicando que o próximo nível será um *Modelo de Aplicação Independente de Plataforma* <<MAIP>>, e a etiqueta *padroesAlvo* indicando o *Padrão de Aplicação Independente de Plataforma* com estereótipo <<PAIP>> que será utilizado na transformação para o próximo nível de abstração. Nesse caso o padrão utilizado será o padrão Facade do catálogo GoF.

A certificação realizada nesse modelo corresponde à certificação da primeira etapa de transformação ou refinamento. As figuras 7, 8 e 9 apresentam os resultados dessa certificação. Na figura 7, o resultado apresenta a estrutura de todos os padrões identificados no modelo analisado. Na figura 8, o resultado indica as conformidades encontradas na comparação entre as construções dos padrões existentes no modelo analisado e suas representações existentes no repositório. Outro resultado possível seria a exibição da relação de erros existentes na utilização de algum padrão no modelo. Na figura 9, o resultado relaciona as conformidades encontradas nas definições de transformação existentes no modelo original. Nesse resultado são apresentados os elementos desse modelo, os padrões de projeto utilizados na transformação de cada elemento, com os respectivos elementos e papéis gerados no modelo transformado. Ainda outro resultado possível seria a relação de erros encontrados, onde seriam apresentados os elementos dos dois modelos cujos mapeamentos de padrões não correspondem às definições existentes no modelo original. O objetivo desses resultados é auxiliar os desenvolvedores na correção das inconsistências existentes.

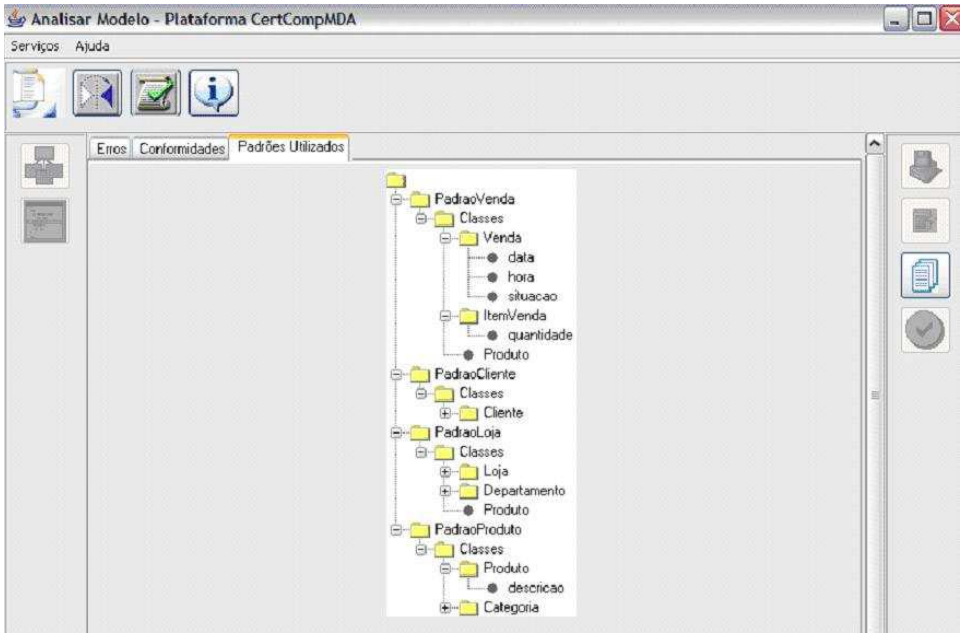

Figura 7. Padrões Utilizados

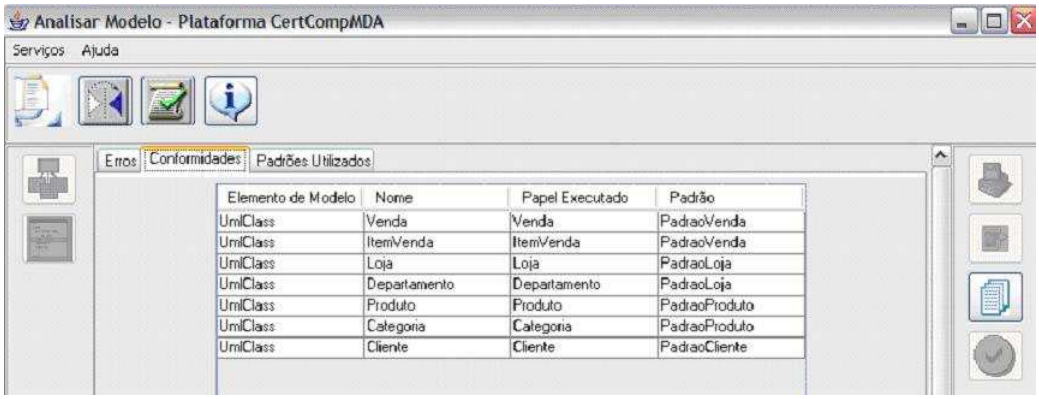

Figura 8. Padrões Utilizados – Conformidades

Figura 9. Transformação de Padrões Utilizados – Conformidades

Na segunda etapa de transformação, após a aplicação do padrão Facade, é gerado o primeiro e único modelo com estereótipo <<MAIP>> com nívelModelo igual a zero, indicando que o próximo nível será um *Modelo de Aplicação de Plataforma Específica* <<MAPE>>. A figura 10 apresenta esse modelo, onde temos a ilustração da utilização do estereótipo <<ClassePadrao>> por todos seus elementos de classe, indicando que executam um papel de padrão, especificado na etiqueta *padroesExecutados*. A etiqueta *padroesAlvo* indica a utilização do padrão

SessionFacadeValueObjet, específico da plataforma J2EE, na transformação para o próximo nível de abstração.

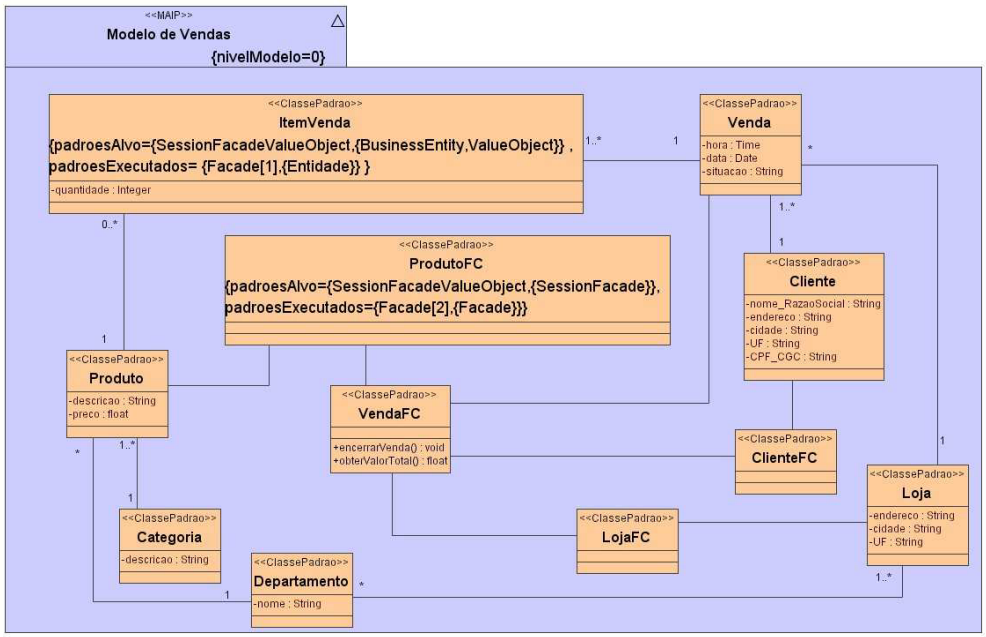

Figura10. <<MAIP>> Sistema de Vendas

A figura 11 apresenta o modelo do padrão SessionFacadeValueObject, utilizando o perfil UML para representação de padrões. Esse padrão é resultante de uma configuração nova de padrão de projeto, com a junção dos padrões SessionFacade e ValueObject do Core J2EE. Possui o estereótipo <<PAPE>> (*Padrão de Aplicação de Plataforma Específica*).

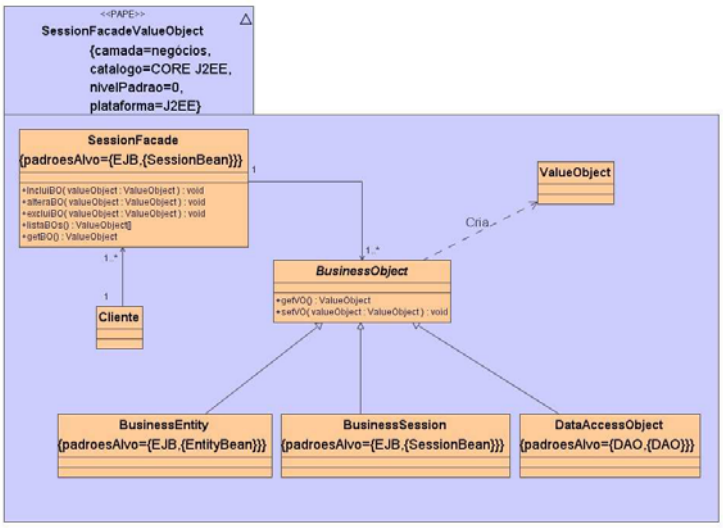

Figura 11. Padrão da Plataforma J2EE : SessionFacadeValueObject

A terceira etapa de transformação ou refinamento, após a aplicação do padrão SessionFacadeValueObject, resulta no primeiro e último modelo PSM, antes da transformação para o código J2EE. Esse modelo possui estereótipo <<MAPE>> e etiqueta *nivelModelo* igual a zero.

Além do estereótipo <<ClassePadrao>>, também podem ser utilizados os estereótipos <<OperacaoPadrao>> e <<AtributoPadrao>> nos elementos de modelo operação e atributo respectivamente, para indicar que executam papéis de padrão. Na transformação para o código, todos os estereótipos <<ClassePadrao>>, <<OperacaoPadrao>> e <<AtributoPadrao>> associados a sua respectiva etiqueta *padroesExecutados*, existentes no último nível de modelo <<MAPE>>, precisam ser transformados em anotações, para permitir a identificação dos padrões no código implementado. A figura 11 apresenta o exemplo do código gerado com a utilização dessas anotações.

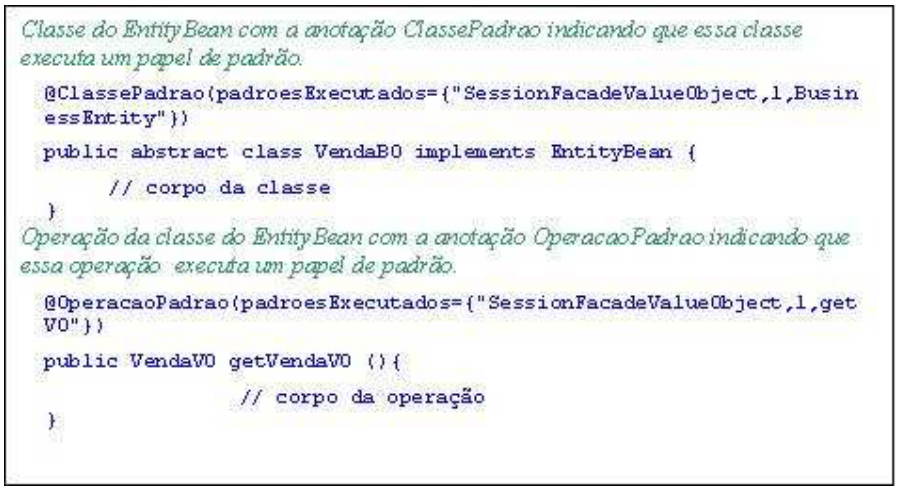

Figura 11. Estereótipos e Etiquetas como Anotações no Código

A etiqueta *padroesExecutados* é multivalorada, permitindo a definição de vários papéis de padrão para uma mesma classe, operação ou atributo, e através da informação de instância em sua lei de formação, podemos ter diversas ocorrências do mesmo padrão em um componente ou aplicação.

Em todas as etapas de transformação do ciclo de desenvolvimento do sistema, com a aplicação dos padrões de projeto específicos a cada nível de abstração, as certificações são executadas pelo gerente de reutilização, através das validações dos modelos pelo serviço de análise de modelos, conforme definido no ambiente de certificação proposto.

Os detalhes sobre a especificação dos dois perfis UML criados para o ambiente proposto, com todas as definições dos estereótipos e etiquetas utilizados, bem como a apresentação completa do cenário utilizado na validação das funcionalidades do ambiente e sistema computacional, se encontram em Gomes, M.C.F (2005).

5. Conclusão

O processo de certificação proposto inclui a definição de um formalismo de representação de padrões, com flexibilidade suficiente para representar qualquer padrão de projeto utilizado no ciclo de desenvolvimento. Define um mecanismo de mapeamento entre os padrões através dos diversos níveis de abstração, permitindo a validação das transformações de padrões na passagem de um nível para outro imediatamente adjacente. Esse sistema de certificação pode ser aplicado em qualquer

ambiente orientado a modelos, independente da quantidade de níveis de abstração e refinamentos utilizados.

Dos trabalhos e ferramentas analisados, relacionados com padrões de projeto e com o ambiente orientado a modelos da MDA, o que mais se aproxima com essa proposta é a ferramenta comercial Optimalj [Crupi e Baerveldt 2005]. Uma ferramenta que fornece um ambiente de desenvolvimento orientado a modelos baseado em padrões utilizando a plataforma tecnológica J2EE. Porém não apresenta a mesma flexibilidade para representar padrões desenvolvidos sob medida para um determinado ambiente de desenvolvimento, nem para ser utilizado em qualquer quantidade de níveis de abstração.

A utilização de perfis UML específicos para representação e utilização de padrões de projeto nos modelos das aplicações e componentes torna a solução adotada de grande alcance dentro da comunidade de desenvolvimento de software, já que nesse ambiente a UML é intensamente conhecida e adotada, sendo o padrão de fato de linguagem de modelagem de sistemas de software. Uma contribuição importante desse trabalho foi utilizar como ambiente computacional, a tecnologia Java e a plataforma J2EE, referências hoje no desenvolvimento de aplicações robustas e distribuídas. A utilização de um repositório compatível com MOF e JMI trouxe bastante flexibilidade, e a facilidade de manipulação das definições de modelos através das interfaces Java geradas tornou o desenvolvimento nesse ambiente muito mais produtivo. Como contribuições específicas podemos citar o fornecimento de uma implementação de domínio público de transformação de código Java em modelos UML (em XMI ou JMI). Esse trabalho deixa alguns problemas ainda em aberto, que poderão ser resolvidos em trabalhos futuros, como:

- Adição de restrições em OCL aos modelos de representação aumentando a precisão semântica e a precisão dos diagnósticos das avaliações realizadas;
- Desenvolvimento de extensões do sistema de certificação para outras plataformas, implementando serviços de reconstrução de arquitetura (conversão do código fonte nos modelos UML de implementação) específicos para essas plataformas;
- Incorporação dos perfis UML para Java e EJB do OMG e JCP na transformação dos modelos de plataforma específica em código;
- Implementação de um ambiente de automatização do desenvolvimento de componentes e aplicações no ambiente orientado a modelos, através da automatização das transformações de modelos utilizando os mesmos perfis UML e mecanismo de associação entre padrões definidos nesse trabalho.

Referencias

Arlow, J. e Neustadt, I. (2003) Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML. Addison-Wesley.

Blankers, T. (2003) Combining models and patterns: delivering on the promise of increased IT productivity. COMPUWARE White Paper Series. Disponível em: http://www.compuware.com/products/optimalj/1794_ENG_HTML.htm#MDA

Crupi, J. e Baerveldt, F. (2005) Implementing Sun Microsystems Core J2EE Patterns. COMPUWARE White Paper. Disponível em: http://www.bitpipe.com/detail/RES/1095961328_371.html?src=FEATURE_GLOBAL

Dong, J. e Yang, S. (2003) Visualizing Design Patterns With A UML Profile. The Proceedings of the IEEE Symposium on Visual/Multimedia Languages (VL), pag.123-125, Auckland, Nova Zelandia, outubro, 2003. Disponível em: http://www.utdallas.edu/~syang/vl03.pdf

Eckerson, W. W. e Manes, A. T. (1999) Paving the Way for Transparent Application and Data Interchange. A Java API for Metadata. SUN JMI White Papers. Disponível em: http://java.sun.com/products/jmi/pdf/JavaMetadataAPI.pdf

France, R., *et al*. (2003) A Role-Based Metamodeling Approach to Specifying Design Patterns. In Proceeding of 27th IEEE Annual International Computer Software and Applications Conference, pag. 452- 457, Dallas, Texas, novembro, 2003. Disponível em: http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=1245379

Frankel, D. S. (2003) Model Driven Architecture. Applying MDA to Enterprise Computing. John Wiley & Sons, OMG Press.

Gamma, E., *et al*. (1995) Design Patterns: Elements of reusable object-oriented software. Addison-Wesley.

Gomes, M.C.F. (2005) Certificação da Utilização de Padrões de Projeto no Desenvolvimento Orientado a Modelos. Dissertação de Mestrado. Programa de Pós-graduação em Informática do IM/NCE – Universidade Federal do Rio de Janeiro

Herzum, P. e Sims, O. (1999) Business Component Factory. John Wiley & Sons.

JSR175 (2004) A Program Annotation Facility for the Java Programming Language. JSR175 final release. JCP. Disponível em: http://www.jcp.org/en/jsr/detail?id=175

JSR026 (2001) UML/EJB Mapping Specification. JSR026 Public Draft. JCP. Disponível em: http://www.jcp.org/aboutJava/communityprocess/review/jsr026/

JSR244 (2005). Java Platform Enterprise Edition 5 Specification. JSR244 Close of Public Review. JCP. Disponível em: http://jcp.org/aboutJava/communityprocess/pr/jsr244/index.html

JSR40 (2002). Java Metadata Interface (JMI) Specification. Versão 1.0. OMG. Disponível em: http://jcp.org/aboutJava/communityprocess/pr/jsr244/index.html

Matula, M. (2003) NetBeans Metadata Repository. SUN white Paper. Disponível em: http://mdr.netbeans.org/MDR-whitepaper.pdf

MDA (2003). Guide Version 1.0.1. Document Number: omg/2003-06-01. OMG. Disponível em: http://mdr.netbeans.org/MDR-whitepaper.pdf

Mellor, S. J., *et al*. (2004) MDA Distilled: Principles of Model-Driven Architecture. Addison Wesley.

OMG (2004) Metamodel and UML Profile for Java and EJB Specification. Version 1.0. Disponível em: http://www.omg.org/docs/formal/04-02-02.pdf

Poole, J. D. (2001) Model-Driven Architecture: Vision, Standards And Emerging Technologies. Workshop on Metamodeling and Adaptive Object Models.

XMI (2003). XML Metadata Interchange. Versão 2.0. OMG. Disponível em http://www.omg.org/docs/formal/03-05-02.pdf